LogiTorch: A PyTorch-based library for logical reasoning on natural language

Chadi Helwe, Chloé Clavel, Fabian Suchanek
Télécom Paris, Institut Polytechnique de Paris, France
{chadi.helwe, chloe.clavel, suchanek}@telecom-paris.fr

Abstract

Logical reasoning on natural language is one of the most challenging tasks for deep learning models. There has been an increasing interest in developing new benchmarks to evaluate the reasoning capabilities of language models such as BERT. In parallel, new models based on transformers have emerged to achieve ever better performance on these datasets. However, there is currently no library for logical reasoning that includes such benchmarks and models. This paper introduces LogiTorch, a PyTorch-based library that includes different logical reasoning benchmarks, different models, as well as utility functions such as co-reference resolution. This makes it easy to directly use the preprocessed datasets, to run the models, or to finetune them with different hyperparameters. LogiTorch is open source and can be found on GitHub[1].

1 Introduction

Machine reasoning over natural language has been an object of research since the 1950s (Newell and Simon, 1956; McCarthy et al., 1960). One prototypical task in the domain is Textual Entailment: Given a premise (such as "I ate a cake"), the goal is to determine whether a hypothesis ("I ate something sweet") is entailed or not. Other logical reasoning tasks are question answering, multiple choice question answering, and proof generation.

Lately, deep learning models have shown impressive performance on tasks such as these, in particular transformer-based models such as BERT (Devlin et al., 2019) and GPT-3 (Brown et al., 2020). However, the models can be distracted easily by trap words, syntactic variations (Kassner and Schütze, 2020), or negation (Kassner and Schütze, 2020; Ettinger, 2020; Hossain et al., 2020, 2022; Helwe et al., 2021). Hence, the question of whether these models can logically reason on text is still open (Niven and Kao, 2019; Helwe et al., 2021). New models are being created incessantly (e.g., LogiGAN (Pi et al., 2022) and Logiformer (Xu et al., 2022) in 2022), and new datasets are being created to evaluate these models, including, e.g., LogiQA (Liu et al., 2021b) and ProofWriter (Tafjord et al., 2021). The initiative of open-sourcing toolkits has accelerated the progress in the field of natural language processing, driven by projects such as Transformers (Wolf et al., 2020) from HuggingFace and Stanza (Qi et al., 2020) from Stanford. However, this progress has not yet arrived in the field of logical reasoning: researchers still have to find and download different models, parameterize them, find the corresponding datasets, bring them into suitable formats, and fine-tune the models. The datasets are maintained on different Web pages, exhibit different formats (JSON vs. full text, numerical vs. textual labels, etc.), and follow different conventions, which makes it cumbersome to apply one model across several sources. The models themselves are implemented in different frameworks, have different input and output formats, require different dependencies, and differ in the way of running them, which makes it burdensome to exchange one model for another. Some models are not even available online, but have to be re-implemented from scratch based on the diagrams in the scientific publications. All of this hinders reproducibility, re-usability, comparability, and ultimately scientific progress in the area.

In this paper, we propose to bring the benefits of open source libraries to the domain of logical reasoning: we build a Python library, LogiTorch, that includes 14 datasets and 4 implemented models for 3 different logical reasoning tasks. All models can be called in a unified way, all datasets of one task are available in the same standardized format, and all models can be run with all datasets of the same task. All models have been re-implemented from the research papers that proposed them, and they

[1] https://github.com/LogiTorch/logitorch

have been validated by subjecting them to the same experiments as the original papers, with comparable results. More models and benchmarks are in preparation. LogiTorch works on top of PyTorch (Paszke et al., 2019), and uses the Transformers library. It also includes utility functions used for preprocessing, such as coreference resolution and discourse delimitation.

The rest of the paper is organized as follows. Section 2 discusses the design and components of LogiTorch, and describes the datasets, utility functions, and models. Section 3 shows the experimental results of our implemented models on different logical reasoning tasks. We conclude in Section 4.

2 LogiTorch

LogiTorch is our Python library for logical reasoning on natural language text. Figure 1 shows the tree structure of our library. It is built on top of PyTorch and consists of 5 parts:

Datasets. We gathered different logical reasoning datasets that allow users to evaluate the reasoning capabilities of deep learning models on natural language. Once a dataset is called from LogiTorch, it is downloaded, and wrapped into an object that inherits the Dataset class of PyTorch. This means that all datasets are accessible via the same interface. We describe the datasets in detail in Section 2.1.

Data Collators. Different models require different preprocessing steps for the same data and same task: one model may work on numerical vectors, the other on textual input. Hence, we designed, for each pair of a dataset and a model, a data collator that brings the dataset into the format required by the model.

Utilities. Some models require supplementary features in addition to the input text. For example, the DAGN model (Huang et al., 2021) requires the discourse structure of the input in order to create a logical graph representation of it. For such cases, LogiTorch provides different utility functions, most notably for discourse structure analysis, coreference resolution, and logical expression extraction, which we discuss in Section 2.2.

Models. LogiTorch provides several deep learning models that have been designed to perform logical reasoning tasks such as proof generation and textual entailment. For each model, we either provide an implementation from scratch, or a wrapper over its original implementation. For the transformer-based models, we use the Transformers library from HuggingFace for the implementation of the models. We describe the models in detail in Section 2.3.

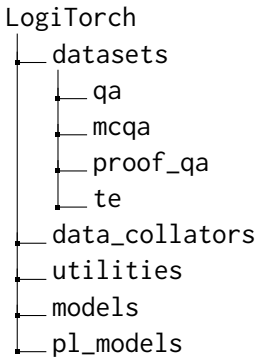

Figure 1: Tree structure of LogiTorch

PyTorch Lightning Models. For each implemented model, we also provide a PyTorch Lightning version. It includes the model, the optimizer, the training loop, and the validation evaluation. For example, the PRover model (Saha et al., 2020) has a PyTorch Lightning version called PLPRover. This allows users to play with features such as multi-GPU and fast-low precision training without modifying the training loop.

2.1 Datasets

The current implemented datasets focus on evaluating the reasoning capabilities of deep learning models. They cover four tasks: Multiple Choice Question Answering (MCQA), Question Answering (QA), Proof Generation, and Textual Entailment (TE). Table 1 shows the task and the number of instances of each dataset. Let us now describe each task and the associated datasets.

Multiple Choice Question Answering (MCQA) is the task of choosing the correct answer to a question from a list of possible answers. Here is an example taken from the LogiQA dataset (Liu et al., 2021b):

> **Context:** David knows Mr. Zhang's friend Jack, and Jack knows David's friend Ms. Lin. Everyone of them who knows Jack has a master's degree, and everyone of them who knows Ms. Lin is from Shanghai.
> **Question:** Who is from Shanghai and has a master's degree?
> **Choices:** (A) David (B) Jack (C) Mr. Zhang (D) Ms. Lin

We implement the following MCQA datasets,

Dataset	Task	Training Instances	Validation Instances	Testing Instances
AR-LSAT	MCQA	1,630	231	230
ReClor	MCQA	4,368	500	1,000
LogiQA	MCQA	7,376	651	651
RuleTaker	QA	587,922	84,030	173,496
ProofWriter	QA/Proof Generation	585,860	85,520	174,180
ParaRules Plus	QA	360,000	64,658	10,798
AbductionRules	QA	80,024	11,432	22,928
ConTRoL	TE	6,719	799	805
SNLI	TE	550,152	10,000	10,000
MNLI	TE	392,702	20,000	20,000
RTE	TE	2,490	277	3,000
Negated SNLI	TE	-	-	1,500
Negated MNLI	TE	-	-	1,500
Negated RTE	TE	-	-	1,500

Table 1: Datasets implemented in LogiTorch

which all require reasoning capabilities to choose the correct answer:

AR-LSAT (Zhong et al., 2021) is a dataset that was constructed by selecting the analytical reasoning section of 90 LSAT exams from 1991 to 2016.

LogiQA (Liu et al., 2021b) assesses the logical deductive ability of language models for the case where the correct answer to a question is not explicitly included in the question. The corpus includes paragraph-question pairs translated from the National Civil Servants Examination of China.

ReCloR (Yu et al., 2019) is a corpus consisting of questions retrieved from standardized exams such as LSAT and GMAT. To adequately evaluate a model without allowing it to take advantage of artifacts in the corpus, the testing set is split into two sets: the EASY set where the instances are biased, and the HARD set where they are not.

Question Answering (QA) is the task of answering a question given a context. Here is an example:

> **Context:** Erin is young. Erin is not kind. If someone is young and not kind then they are big.
> **Question:** Erin is big ?
> **Answer:** True

Again, we implement the QA datasets that focus on reasoning:

RuleTaker (Clark et al., 2021) is a set of many datasets to evaluate the deductive ability of language models. Each dataset consists of facts and rules and a boolean question. The model has to perform logical deductions from the rules and facts in order to answer the question. The datasets includes synthetically generated subsets that require different depths of reasoning, i.e., different numbers of deduction steps to answer a question. The dataset also includes the Bird dataset (which showcases McCarthy's problem of abnormality (McCarthy, 1986)), the Electricity dataset (which simulates the functions of an appliance), and the ParaRules corpus (where crowd workers paraphrased sentences such as "Bob is cold" to "In the snow sits Bob, crying from being cold").

ParaRules Plus (Bao, 2021) is an improved version of ParaRules (Clark et al., 2021). It has more examples for the instances with larger reasoning depths.

Abduction Rules (Young et al., 2022) is a dataset that evaluates the abductive reasoning capabilities of language models. It is generated similarly to ParaRule Plus, but in this task, the model has to generate an answer to explain an observation.

Proof Generation is an extension of the QA task, where each answer has to be accompanied by a proof. Here is an example:

> **Context:** Fact 1: Erin is young.
> Fact 2: Erin is not kind.
> Rule1: If someone is young and not kind then they are big.
> **Question:** Erin is big ?
> **Answer:** True
> **Proof:** (Fact 1 & Fact 2) $\rightarrow$ Rule 1

We have one dataset so far, **ProofWriter** (Tafjord et al., 2021), which was designed similarly to the RuleTaker datasets. However, the ProofWriter dataset contains proofs for the answer of each question. Furthermore, there is a variant of the dataset that considers the open-world assumption.

Textual Entailment (TE, also RTE) is the task of predicting whether a premise entails or contradicts

```python
import pytorch_lightning as pl
from pytorch_lightning.callbacks import ModelCheckpoint
from torch.utils.data.dataloader import DataLoader

from logitorch.data_collators.ruletaker_collator import RuleTakerCollator
from logitorch.datasets.qa.ruletaker_dataset import RuleTakerDataset
from logitorch.pl_models.ruletaker import PLRuleTaker

train_dataset = RuleTakerDataset("depth-5", "train")
val_dataset = RuleTakerDataset("depth-5", "val")

ruletaker_collate_fn = RuleTakerCollator()

train_dataloader = DataLoader(train_dataset, batch_size=32, collate_fn=ruletaker_collate_fn)
val_dataloader = DataLoader(val_dataset, batch_size=32, collate_fn=ruletaker_collate_fn)

model = PLRuleTaker(learning_rate=1e-5, weight_decay=0.1)

checkpoint_callback = ModelCheckpoint(
    save_top_k=1,
    monitor="val_loss",
    mode="min",
    dirpath="models/",
    filename="best_ruletaker.ckpt",
)
trainer = pl.Trainer(accelerator="gpu", gpus=1)
trainer.fit(model, train_dataloader, val_dataloader)
```

Listing 1: Training the RuleTaker Model

```python
from logitorch.pl_models.ruletaker import PLRuleTaker

model = PLRuleTaker.load_from_checkpoint("models/best_ruletaker.ckpt")

context = "Bob is smart. If someone is smart then he is kind."
question = "Bob is kind."

model.predict(context, question)
```

Listing 2: Predicting with the RuleTaker Model

a hypothesis. Here is an example:

> **Premise:** The two boys are in martial arts poses in an outside basketball court.
> **Hypothesis:** The two boys are not outdoors.
> **Answer:** Contradiction

SNLI (Bowman et al., 2015) is a large human-annotated corpus of premise-hypothesis pairs that are labeled with "entailment", "contradiction", or "neutral". The premises of this dataset are image captions from Flickr30k, while its hypotheses were generated by human annotators.

MNLI (Williams et al., 2018) is a large dataset that was labeled in the same way as SNLI. However, unlike SNLI, MNLI covers different text genres such as fiction, telephone speech, and letters. It also has longer instances.

RTE (Dagan et al., 2005; Haim et al., 2006; Giampiccolo et al., 2007, 2008; Bentivogli et al., 2009) is a much smaller dataset than SNLI and MNLI. It has just two classes, "entailment" and "non-entailment".

Negated TE (Hossain et al., 2020) is a testing set of benchmarks to evaluate the understanding of negation in language models. Each negated benchmark was created by randomly selecting 500 premise-hypothesis pairs from SNLI, MNLI, and RTE datasets and introducing the negation "not". For each pair, three new pairs were generated (negated premise/hypothesis, premise/negated hypothesis, and negated premise/negated hypothesis).

ConTRoL (Liu et al., 2021a) is a dataset of context-hypothesis pairs to evaluate contextual reasoning capabilities over long texts. In contrast to other TE datasets, the corpus consists of passage-long premises, and it evaluates different types of reasoning such as analytical or temporal reasoning, which makes this task more challenging.

2.2 Utilities

LogiTorch implements several utility functions that can be used for feature engineering:

Coreference Resolution is the task of finding all mentions in a text that refer to the same entity. For example, in "Zidane is one of the best footballers. He won the World Cup in 1998", the words "Zidane" and "he" refer to the same person. Coreference resolution is used by the Focal Reasoner model (Ouyang et al., 2021) to construct a graph of

fact triples, where the same mentions are connected with an undirected edge. In LogiTorch, we implemented a wrapper over a finetuned SpanBERT (Joshi et al., 2020) for coreference resolution.

Logical Expression Extraction is the task of extracting a logical representation from a text, in order to infer new logical expressions. For example, the sentence "If you have no keyboarding skills, you will not be able to use a computer" can be split into α = "you have no keyboarding skills" and β = "you are not be able to use a computer". The sentence can then be rewritten as $\alpha \rightarrow \beta$. From this, we can infer by transposition that $\neg\beta \rightarrow \neg\alpha$, which corresponds to "If you are able to use a computer, you have keyboarding skills". The LReasoner model (Wang et al., 2022) uses this utility function to extend the input with logical expressions. In LogiTorch, we developed a wrapper over the code provided by LReasoner for this purpose.

Discourse Delimitation is the task of splitting a text into elementary discourse units (EDU). It is used for the rhetorical structure theory (RST), in which it is a tree representation of a text where the leaves are EDUs, and the edges are rhetorical relations. For example, "A signal in a pure analog system can be infinitely detailed, while digital systems cannot produce signals that are more precise than their digital unit" is split into two EDUs: "A signal in a pure analog system can be infinitely detailed", and "digital systems cannot produce signals that are more precise than their digital unit". The DAGN model (Huang et al., 2021) requires EDUs to construct a graph of discourse units.

2.3 Models

LogiTorch currently implements four models:

RuleTaker (QA task) (Clark et al., 2021) is a RoBERTa-Large model (Liu et al., 2019) that has been finetuned first on the RACE dataset (Lai et al., 2017), and then finetuned again for rule-based reasoning. The model takes as input facts and rules and a boolean question. The output is either True or False. The RoBERTa model has a similar architecture to BERT, but performs better on many NLP tasks. This is because it is pretrained for a longer period, with large batches, and on a larger dataset. The pretraining task is only the Masked Language Modeling (MLM) task, but the masked tokens are changed after each training epoch.

ProofWriter (QA and proof generation) (Tafjord et al., 2021) is a T5 model (Raffel et al., 2020) finetuned to perform rule-based reasoning. It takes as input facts and rules and a question. The output is either True, False, or Unknown (if the trained dataset considers the open-world assumption). T5 is a text-to-text transfer transformer that was pretrained on a variety of NLP problems such as textual entailment, coreference resolution, linguistic acceptability, and semantic equivalence.

PRover (QA and proof generation) (Saha et al., 2020) is built on RoBERTa with three modules: the QA module, Node module, and Edge module. The QA module is responsible for answering a question as either True or False. The Node and Edge modules are responsible for generating proofs. The Node module predicts the relevant rules and facts used to generate the answer, and the Edge module predicts the link between two relevant facts and between a relevant fact and a relevant rule.

BERTNOT (TE task) (Hosseini et al., 2021) is a BERT model that is pretrained using the unlikelihood loss and knowledge distillation functions for the MLM task to model negation. Then it is finetuned on textual entailment tasks. This model is more robust on examples containing negations, and performs better on the negated NLI dataset than the original BERT.

Future releases will include newer models such as LReasoner (Huang et al., 2021), Focal Reasoner (Ouyang et al., 2021), AdaLoGN (Li et al., 2022), Logiformer (Xu et al., 2022), and LogiGAN (Pi et al., 2022).

2.4 Library Usage

Listing 1 shows a detailed example of how a model can be trained on a rule-based reasoning dataset for QA. The RuleTaker model is trained on its corresponding dataset. In Lines 9-10, we initialize the training and validation datasets with the RuleTakerDataset. We specify which sub-dataset and which split we want to use. In Line 12, we initialize the RuleTaker data collator for preprocessing the datasets. We then use the Dataloader to pre-load the datasets and use them as batches. In Line 17, we initialize the PyTorch Lightning version of RuleTaker and specify the learning rate, and the weight decay. PyTorch Lightning provides the ModelCheckpoint, which allows monitoring the validation loss and saving the best model. In Line 26, we use the PyTorch Lightning's Trainer to automate the training loop. It takes several parameters, including the accelerator, which allows training on different de-

Depth	*RuleTaker*[1]		*PRover*[2]		*ProofWriter*[2]	
	LogiTorch	*Original*	*LogiTorch*	*Original*	*LogiTorch*	*Original*[3]
0	99.9	100	100	100	99.9	100
1	98.6	98.4	99.7	99.0	98.0	99.1
2	99.1	98.4	99.5	98.8	96.7	98.6
3	99.2	98.9	99.7	99.1	97.2	98.5
4	99.7	99.2	99.7	98.8	98.1	98.7
5	99.3	99.8	99.5	99.3	99.1	99.3
All	99.3	99.2	99.7	99.3	98.4	99.2

Table 2: Accuracies of different models for the QA task at different reasoning depths. [1] Depth-5 of the testing set of RuleTaker dataset. [2] Depth-5 of the testing set of ProofWriter dataset. [3] The original implementation uses a (more powerful) T5-11B model.

vices such as CPUs, GPUs, and TPUs. Finally, we train the model with the fit function. Future releases will also provide pre-configured pipelines to train models.

Listing 2 shows the code for testing the best-saved model of Listing 1. In Line 3, we load the best model. In Line 8, we use the predict function, which takes as input a context and a question, and predicts either 0 (for False) or 1 (for True).

Dataset		*LogiTorch's BERTNOT*	*Original BERTNOT*
SNLI	Val	90.4	89.00
	Neg	47.8	45.96
MNLI	Val	83.2	84.31
	Neg	64.0	60.89
RTE	Val	65.6	69.68
	Neg	57.7	74.47

Table 3: Results of our BERTNOT implementation on different textual-entailment datasets.

3 Evaluation

We compared the performance of each model in LogiTorch to the performance of the model in the original paper, on the same datasets: we trained the Ruletaker model on the training set of RuleTaker with language reasoning paths up to depth 5 and tested it on its testing set; we trained the PRover and ProofWriter models on the training set of ProofWriter with language reasoning paths up to depth 5 and tested them on the corresponding testing set; and we trained the BERTNOT model (a pretrained BERT Base Cased model) on the MLM task, with the negated Wikipedia corpus provided by Hosseini et al. (2021) (included in LogiTorch), finetuned the model on each TE dataset (MNLI, SNLI, and RTE) and tested it on its negated counterparts (Hossain et al., 2020). All models use the the same settings as in the original papers.

Table 2 shows the results of the three different models on the QA task at different reasoning depths. Our model implementations achieve near-perfect accuracies, which are comparable to the performance in the original papers. Table 3 shows the performance on the TE task on each TE training dataset (SNLI, MNLI, and RTE). Again, our model achieves nearly the same results as reported in the original paper (Hosseini et al., 2021) on the MNLI and SNLI datasets. We are getting lower results on the RTE dataset. We assume that this is because the finetuned model has a high variance due to the small size of the training set of RTE.

4 Conclusion

We have introduced LogiTorch, a Python library for logical reasoning on natural language. It is built on top of PyTorch in combination with the Transformers and PyTorch Lightning libraries. LogiTorch includes an extensive list of textual logical reasoning datasets and utility functions, and different implemented models. The library allows researchers and developers to easily use a logical reasoning dataset and train logical reasoning models with just a few lines of code. The library is available on GitHub and is under active development.

For future work, we will add new datasets, and implement models such as DAGN, Focal Reasoner, and LogiGAN with their utility functions for feature engineering. Finally, we want to invite researchers and developers to contribute to LogiTorch. We believe that such a library will lower the hurdles to research in the area, foster re-usability, encourage comparative evaluation, strengthen reproducibility, and advance the culture of open software and data.

Acknowledgements. This work was partially funded by ANR-20-CHIA-0012-01 (“NoRDF”).

5 Ethical Considerations

Users of LogiTorch should distinguish the datasets and models of our library from the originals. They should always credit and cite both our library and the original data source, as in "We used LogiTorch's (Helwe et al., 2022) re-implementation of BERTNOT (Hosseini et al., 2021)". These conditions are mentioned on our GitHub page.

References

Qiming Bao. 2021. Pararule plus: A larger deep multi-step reasoning dataset over natural language.

Luisa Bentivogli, Peter Clark, Ido Dagan, and Danilo Giampiccolo. 2009. The fifth pascal recognizing textual entailment challenge. In *TAC*.

Samuel Bowman, Gabor Angeli, Christopher Potts, and Christopher D Manning. 2015. A large annotated corpus for learning natural language inference. In *Conference on Empirical Methods in Natural Language Processing*.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. 2020. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901.

Peter Clark, Oyvind Tafjord, and Kyle Richardson. 2021. Transformers as soft reasoners over language. In *Proceedings of the Twenty-Ninth International Conference on International Joint Conferences on Artificial Intelligence*, pages 3882–3890.

Ido Dagan, Oren Glickman, and Bernardo Magnini. 2005. The pascal recognising textual entailment challenge. In *Machine Learning Challenges Workshop*.

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.

Allyson Ettinger. 2020. What bert is not: Lessons from a new suite of psycholinguistic diagnostics for language models. *Transactions of the Association for Computational Linguistics*.

Danilo Giampiccolo, Hoa Trang Dang, Bernardo Magnini, Ido Dagan, Elena Cabrio, and Bill Dolan. 2008. The fourth pascal recognizing textual entailment challenge. In *TAC*.

Danilo Giampiccolo, Bernardo Magnini, Ido Dagan, and William B Dolan. 2007. The third pascal recognizing textual entailment challenge. In *ACL-PASCAL workshop on textual entailment and paraphrasing*.

R Bar Haim, Ido Dagan, Bill Dolan, Lisa Ferro, Danilo Giampiccolo, Bernardo Magnini, and Idan Szpektor. 2006. The second pascal recognising textual entailment challenge. In *The Second PASCAL Challenges Workshop on Recognising Textual Entailment*.

Chadi Helwe, Chloé Clavel, and Fabian Suchanek. 2022. Logitorch: A pytorch-based library for logical reasoning on natural language. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*.

Chadi Helwe, Chloé Clavel, and Fabian M Suchanek. 2021. Reasoning with transformer-based models: Deep learning, but shallow reasoning. In *3rd Conference on Automated Knowledge Base Construction*.

Md Mosharaf Hossain, Dhivya Chinnappa, and Eduardo Blanco. 2022. An analysis of negation in natural language understanding corpora. In *Annual Meeting of the Association for Computational Linguistics*.

Md Mosharaf Hossain, Venelin Kovatchev, Pranoy Dutta, Tiffany Kao, Elizabeth Wei, and Eduardo Blanco. 2020. An analysis of natural language inference benchmarks through the lens of negation. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 9106–9118.

Arian Hosseini, Siva Reddy, Dzmitry Bahdanau, R Devon Hjelm, Alessandro Sordoni, and Aaron Courville. 2021. Understanding by understanding not: Modeling negation in language models. In *North American Chapter of the Association for Computational Linguistics: Human Language Technologies*.

Yinya Huang, Meng Fang, Yu Cao, Liwei Wang, and Xiaodan Liang. 2021. Dagn: Discourse-aware graph network for logical reasoning. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 5848–5855.

Mandar Joshi, Danqi Chen, Yinhan Liu, Daniel S Weld, Luke Zettlemoyer, and Omer Levy. 2020. Spanbert: Improving pre-training by representing and predicting spans. *Transactions of the Association for Computational Linguistics*, 8:64–77.

Nora Kassner and Hinrich Schütze. 2020. Negated and misprimed probes for pretrained language models: Birds can talk, but cannot fly. In *Annual Meeting of the Association for Computational Linguistics*.

Guokun Lai, Qizhe Xie, Hanxiao Liu, Yiming Yang, and Eduard Hovy. 2017. Race: Large-scale reading comprehension dataset from examinations. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 785–794.

Xiao Li, Gong Cheng, Ziheng Chen, Yawei Sun, and Yuzhong Qu. 2022. Adalogn: Adaptive logic graph network for reasoning-based machine reading comprehension. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 7147–7161.

Hanmeng Liu, Leyang Cui, Jian Liu, and Yue Zhang. 2021a. Natural language inference in context-investigating contextual reasoning over long texts. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 13388–13396.

Jian Liu, Leyang Cui, Hanmeng Liu, Dandan Huang, Yile Wang, and Yue Zhang. 2021b. Logiqa: a challenge dataset for machine reading comprehension with logical reasoning. In *Proceedings of the Twenty-Ninth International Conference on International Joint Conferences on Artificial Intelligence*, pages 3622–3628.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*.

John McCarthy. 1986. Applications of circumscription to formalizing common-sense knowledge. *Artificial intelligence*, 28(1):89–116.

John McCarthy et al. 1960. *Programs with common sense*. RLE and MIT computation center Cambridge, MA, USA.

Allen Newell and Herbert Simon. 1956. The logic theory machine–a complex information processing system. *IRE Transactions on information theory*, 2(3):61–79.

Timothy Niven and Hung-Yu Kao. 2019. Probing neural network comprehension of natural language arguments. In *Annual Meeting of the Association for Computational Linguistics*.

Siru Ouyang, Zhuosheng Zhang, and Hai Zhao. 2021. Fact-driven logical reasoning. *arXiv preprint arXiv:2105.10334*.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. 2019. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems*, 32.

Xinyu Pi, Wanjun Zhong, Yan Gao, Nan Duan, and Jian-Guang Lou. 2022. Logigan: Learning logical reasoning via adversarial pre-training. *arXiv preprint arXiv:2205.08794*.

Peng Qi, Yuhao Zhang, Yuhui Zhang, Jason Bolton, and Christopher D. Manning. 2020. Stanza: A Python natural language processing toolkit for many human languages.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, Peter J Liu, et al. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *J. Mach. Learn. Res.*, 21(140):1–67.

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.

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.

Siyuan Wang, Wanjun Zhong, Duyu Tang, Zhongyu Wei, Zhihao Fan, Daxin Jiang, Ming Zhou, and Nan Duan. 2022. Logic-driven context extension and data augmentation for logical reasoning of text. In *Findings of the Association for Computational Linguistics: ACL 2022*, pages 1619–1629.

Adina Williams, Nikita Nangia, and Samuel Bowman. 2018. A broad-coverage challenge corpus for sentence understanding through inference. In *Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, et al. 2020. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 conference on empirical methods in natural language processing: system demonstrations*, pages 38–45.

Fangzhi Xu, Jun Liu, Qika Lin, Yudai Pan, and Lingling Zhang. 2022. Logiformer: A two-branch graph transformer network for interpretable logical reasoning. In *Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 1055–1065.

Nathan Young, Qiming Bao, Joshua Bensemann, and Michael Witbrock. 2022. Abductionrules: Training transformers to explain unexpected inputs. *arXiv preprint arXiv:2203.12186*.

Weihao Yu, Zihang Jiang, Yanfei Dong, and Jiashi Feng. 2019. Reclor: A reading comprehension dataset requiring logical reasoning. In *International Conference on Learning Representations*.

Wanjun Zhong, Siyuan Wang, Duyu Tang, Zenan Xu, Daya Guo, Jiahai Wang, Jian Yin, Ming Zhou, and Nan Duan. 2021. Ar-lsat: Investigating analytical reasoning of text. *arXiv preprint arXiv:2104.06598*.