

Continuous Entailment Patterns for Lexical Inference in Context

Martin Schmitt and Hinrich Schütze

Center for Information and Language Processing (CIS)

LMU Munich, Germany

martin@cis.lmu.de

Abstract

Combining a pretrained language model (PLM) with textual patterns has been shown to help in both zero- and few-shot settings. For zero-shot performance, it makes sense to design patterns that closely resemble the text seen during self-supervised pretraining because the model has never seen anything else. Supervised training allows for more flexibility. If we allow for tokens outside the PLM’s vocabulary, patterns can be adapted more flexibly to a PLM’s idiosyncrasies. Contrasting patterns where a “token” can be any continuous vector vs. those where a discrete choice between vocabulary elements has to be made, we call our method *CONTinuous pAtterNs* (CONAN). We evaluate CONAN on two established benchmarks for lexical inference in context (LliC) a.k.a. predicate entailment, a challenging natural language understanding task with relatively small training sets. In a direct comparison with discrete patterns, CONAN consistently leads to improved performance, setting a new state of the art. Our experiments give valuable insights into the kind of pattern that enhances a PLM’s performance on LliC and raise important questions regarding our understanding of PLMs using text patterns.¹

1 Introduction

Lexical inference in context (LliC) – also called predicate entailment – is a variant of natural language inference (NLI) or recognizing textual entailment (Dagan et al., 2013) with focus on the lexical semantics of verbs and verbal expressions (Levy and Dagan, 2016; Schmitt and Schütze, 2019). Its goal is to detect entailment between two very similar sentences, i.e., sentences that share subject and object and only differ in the predicate, e.g., $\text{PERSON}(A) \text{ runs } \text{ORG}(B) \rightarrow \text{PERSON}(A) \text{ leads } \text{ORG}(B)$. NLI models that were not specifically

trained with lexical knowledge have been reported to struggle with this task (Glockner et al., 2018; Schmitt and Schütze, 2019), making LliC an important evaluation criterion for general language understanding. Other use cases for this kind of lexical entailment knowledge include question answering (Schoenmackers et al., 2010; McKenna et al., 2021), event coreference (Shwartz et al., 2017; Meged et al., 2020), and link prediction in knowledge graphs (Hosseini et al., 2019).

Although LliC is an inherently directional task, symmetric cosine similarity in a vector space, such as word2vec (an, 2013), has long been the state of the art for this task. Only recently transfer learning with pretrained Transformer (Vaswani et al., 2017) language models (Devlin et al., 2019), has led to large improvements for LliC. Schmitt and Schütze (2021) combine natural language (NL) patterns with a pretrained language model (PLM) and not only set a new state of the art but also beat baselines without access to such patterns.

Empirical findings suggest that a good pattern can be worth 100s of labeled training instances (Le Scao and Rush, 2021), making pattern approaches interesting for low-resource tasks such as LliC. But beyond the intuition that patterns serve as some sort of task instruction (Schick and Schütze, 2021a), little is known about the reasons for their success. Recent findings that (i) PLMs can fail to follow even simple instructions (Efrat and Levy, 2020), that (ii) PLMs can behave drastically different with paraphrases of the same pattern (Elazar et al., 2021), and that (iii) performance increases if we train a second model to rewrite an input pattern with the goal of making it more comprehensible for a target PLM (Haviv et al., 2021), strongly suggest that patterns do not make sense to PLMs in the same way as they do to humans.

Our work sheds light on the interaction of patterns and PLMs and proposes a new method of improving pattern-based models fully automatically.

¹Our code is publicly available: <https://github.com/mnschmit/conan>

On two popular LliC benchmarks, our model (i) establishes a new state of the art without the need for handcrafting patterns or automatically identifying them in a corpus and (ii) does so more efficiently thanks to shorter patterns. Our best model only uses 2 tokens per pattern.

2 The CONAN Model

Continuous patterns. LliC is a binary classification task, i.e., given a premise $\mathbf{p} = p_1 p_2 \dots p_{|\mathbf{p}|}$ and a hypothesis $\mathbf{h} = h_1 h_2 \dots h_{|\mathbf{h}|}$ a model has to decide whether \mathbf{p} entails \mathbf{h} ($y = 1$) or not ($y = 0$). A template-based approach to this task surrounds \mathbf{p} and \mathbf{h} with tokens $t_1 t_2 \dots t_m$ to bias the classifier for entailment detection, e.g., “ \mathbf{p} , which means that \mathbf{h} ”. While in most approaches that leverage a PLM \mathcal{M} these tokens come from the PLM’s vocabulary, i.e., $\forall i. t_i \in \Sigma_{\mathcal{M}}$, we propose a model based on CONtinuous pATterNs (CONAN), i.e., surround the embedding representation of \mathbf{p} and \mathbf{h} with continuous vectors that may be close to but do not have to match the embedding of any vocabulary entry.

For this, we first extend the PLM’s vocabulary by a finite set $C = \{c_1, c_2, \dots, c_{|C|}\}$ of fresh tokens, i.e., $\Sigma = \Sigma_{\mathcal{M}} \cup C$ with $C \cap \Sigma_{\mathcal{M}} = \emptyset$. Then, we distinguish two methods of surrounding \mathbf{p} and \mathbf{h} with these special tokens: α sets them both around and between \mathbf{p} and \mathbf{h} (Eq. (1)) while β only sets them between the two input sentences (Eq. (2)).

$$\alpha_k(\mathbf{p}, \mathbf{h}) = c_1 \dots c_a \mathbf{p} c_{a+1} \dots c_{a+b} \mathbf{h} c_{a+b+1} \dots c_k$$

with $a = \lfloor k/3 \rfloor$, $b = \lfloor k/3 \rfloor + (k \bmod 3)$ (1)

$$\beta_k(\mathbf{p}, \mathbf{h}) = \mathbf{p} c_1 \dots c_k \mathbf{h}$$
 (2)

Note that α divides its k tokens into three parts as equally as possible where any remaining tokens go between \mathbf{p} and \mathbf{h} if k is not a multiple of 3. In particular, this means that the same templates are produced by α and β for $k \leq 2$. We chose this behavior as a generalization of the standard approach to fine-tuning a PLM for sequence classification (such as NLI) where there is only one special token and it separates the two input sequences. The template produced by α_1 and β_1 is very similar to this. A major difference is that the embeddings for C tokens are randomly initialized whereas the standard separator token has a pretrained embedding.

Pattern-based classifier. Given $\gamma \in \{\alpha, \beta\}$, we estimate the probability distribution $P(\hat{y} | \mathbf{p}, \mathbf{h})$ with a linear classifier on top of the pooled sequence representation produced by the PLM \mathcal{M} :

$$P(\hat{y} | \mathbf{p}, \mathbf{h}) = \sigma(\mathcal{M}(\gamma(\mathbf{p}, \mathbf{h}))W + b) \quad (3)$$

where $W \in \mathbb{R}^{d \times 2}$, $b \in \mathbb{R}^2$ are learnable parameters, σ is the softmax function, and applying \mathcal{M} means encoding the whole input sequence in a single d -dimensional vector according to the specifics of the PLM. For BERT (Devlin et al., 2019) and its successor RoBERTa (Liu et al., 2019), this implies a dense pooler layer with tanh activation over the contextualized token embeddings and picking the first of these embeddings (i.e., [CLS] for BERT and $\langle \mathbf{s} \rangle$ for RoBERTa).² For training, we apply dropout with a probability of 0.1 to the output of $\mathcal{M}(\cdot)$.

Inference with multiple patterns. Previous work (Bouraoui et al., 2020; Schmitt and Schütze, 2021) combined multiple patterns with the intuition that different NL patterns can capture different aspects of the task. This intuition makes also sense for CONAN. We conjecture that an efficient use of the model parameters requires different continuous patterns to learn different representations, which can detect different types of entailment. Following the aforementioned work, we form our final score s by combining the probability estimates from different patterns Γ by comparing the maximum probability for the two classes 0, 1 over all patterns:

$$m_1(\mathbf{p}, \mathbf{h}) = \max_{\gamma \in \Gamma} P(\hat{y} = 1 | \gamma(\mathbf{p}, \mathbf{h}))$$

$$m_0(\mathbf{p}, \mathbf{h}) = \max_{\gamma \in \Gamma} P(\hat{y} = 0 | \gamma(\mathbf{p}, \mathbf{h}))$$

$$s(\mathbf{p}, \mathbf{h}) = m_1(\mathbf{p}, \mathbf{h}) - m_0(\mathbf{p}, \mathbf{h}) \quad (4)$$

In conclusion, a CONAN model γ_k^n is characterized by three factors: (i) The type of pattern $\gamma \in \{\alpha, \beta\}$, (ii) the number of patterns $n \in \mathbb{N}$, and (iii) the number of tokens $k \in \mathbb{N}$ per pattern.

Training. While multiple patterns are combined for decision finding during inference, we treat all patterns separately during training – as did previous work (Schmitt and Schütze, 2021). So, given a set of patterns Γ , we minimize the negative log-likelihood of the training data \mathcal{T} , i.e.,

$$\sum_{(\mathbf{p}, \mathbf{h}, y) \in \mathcal{T}} \sum_{\gamma \in \Gamma} -\log(P(\hat{y} = y | \gamma(\mathbf{p}, \mathbf{h}))) \quad (5)$$

In practice, we apply mini-batching to both \mathcal{T} and Γ and thus compute this loss only for a fraction of the available training data and patterns at a time. In

²Cf. Jacob Devlin’s comment on issue 43 in the official BERT repository on GitHub, <https://github.com/google-research/bert/issues/43>.

	train	dev	test	total
SherLiC	797	201	2,990	3,988
Levy/Holt	4,388	1,098	12,921	18,407

Table 1: Number of labeled instances per data split.

this case, we normalize the loss by averaging over the training samples and patterns in the mini-batch.

3 Experiments

We conduct experiments on two established LiC benchmarks, SherLiC (Schmitt and Schütze, 2019) and Levy/Holt (Levy and Dagan, 2016; Holt, 2018), using the data splits as defined in (Schmitt and Schütze, 2021) for comparison. Both benchmarks contain a majority of negative examples (SherLiC: 67%, Levy/Holt: 81%), making the detection of the entailment (i.e., the minority) class a particular challenge. See Table 1 for dataset and split sizes. Note that Levy/Holt is nearly 5 times bigger than SherLiC and still has less than 5k train samples.

Following (Schmitt and Schütze, 2021), we use RoBERTa as underlying PLM and also use the same hyperparameters whenever possible for comparison. Also following Schmitt and Schütze (2021), we instantiate the typed placeholders A , B in SherLiC with Freebase (Bollacker et al., 2008) entities, making sure that A and B are not assigned the same entity. See Appendix A for full training details.

We evaluate model performance with two metrics: (i) The area under the precision-recall curve for precision values ≥ 0.5 (AUC) as threshold-less metric using only the score s defined in the previous section and (ii) the F1 score of actual classification decisions after tuning a decision threshold ϑ on the respective dev portion of the data. Our implementation is based on (Wolf et al., 2019).

4 Results

Choosing n and k . The number n of patterns and the number k of C tokens per pattern are essential hyperparameters of a CONAN model.

Fig. 1 shows the impact on performance (measured in AUC) on SherLiC dev for different n - k -combinations. We observe that using too many, too long patterns harms performance w.r.t. the base case $n = k = 1$. Best results are obtained with either a small number of patterns or tokens or both. Comparing the α and β settings, we notice that,

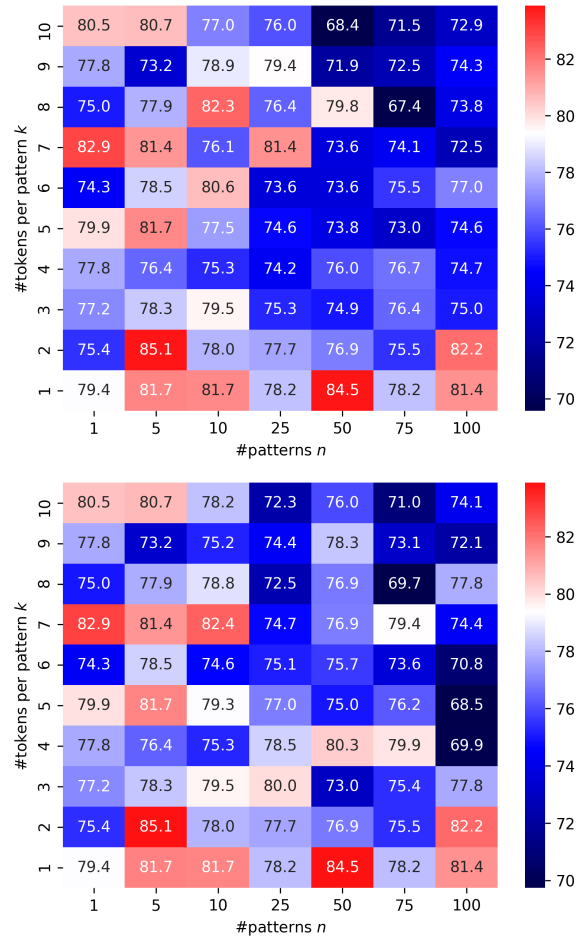


Figure 1: AUC on SherLiC dev for different CONAN models; top = α , bottom = β ; white/red/blue = similar to/better than/worse than $n = k = 1$. Note that $\alpha_k = \beta_k$ for $k \leq 2$.

rounded to one decimal, they produce identical results for both $n = 1$ and $n = 5$ patterns, suggesting that the particular position of the C tokens does not matter much in these settings for SherLiC. Even with $n = 10$ patterns, the two methods only begin to differ with $k \geq 5$ tokens per pattern. Evaluation on the Levy/Holt data (see Fig. 2 in Appendix B) shows more variation between α and β but, otherwise, confirms the trend that small n and k yield better performance.

Our results offer an explanation for the empirical finding in (Schmitt and Schütze, 2021) that patterns retrieved from a corpus lead to worse performance than handcrafted ones because the latter are generally shorter. CONAN models do not only yield better performance, they also provide an automatic way to test pattern properties, such as length, w.r.t. effect on performance for a given task.

Test performance. On both SherLiC (Table 2)

	AUC	P	R	F1
RoBERTa-base				
MANPAT $^{\Phi}$ (baseline)	69.2	62.0	81.2	70.3
AUTPAT $^{\Phi}_{15}$ (baseline)	73.1	63.0	77.4	69.4
CONAN 5_2 ($\vartheta = -0.0768$)	73.2	65.8	84.1	73.8
CONAN $^{50}_1$ ($\vartheta = -0.0108$)	74.0	70.1	78.0	73.8
CONAN- α^1_7 ($\vartheta = -0.1875$)	66.9	68.7	68.4	68.5
CONAN- β^1_7 ($\vartheta = -0.1875$)	66.9	68.7	68.4	68.5
RoBERTa-large				
NLI (baseline)	68.3	60.5	85.5	70.9
MANPAT $^{\Phi\Psi}$ (baseline)	74.4	66.0	80.8	72.6
CONAN 5_2 ($\vartheta = -0.6838$)	75.9	67.9	81.1	73.9
CONAN $^{50}_1$ ($\vartheta = -0.9999$)	73.9	64.8	81.5	72.2
CONAN- α^1_7 ($\vartheta = -0.8797$)	62.6	63.1	76.4	69.1
CONAN- β^1_7 ($\vartheta = -0.9556$)	68.6	60.8	86.0	71.2

Table 2: SherLiC test. AUC denotes the area under the precision-recall curve for precision ≥ 0.5 . All results in %. Bold means best result per column and block. All baselines from (Schmitt and Schütze, 2021). For $k \leq 2$, we simply write CONAN n_k because $\alpha = \beta$.

	AUC	P	R	F1
RoBERTa-base				
NLI (baseline)	72.6	68.7	75.3	71.9
MANPAT $^{\Phi\Psi}$ (baseline)	76.9	78.7	66.4	72.0
CONAN 5_2 ($\vartheta = -0.4809$)	77.6	79.1	67.5	72.8
CONAN 5_1 ($\vartheta = -0.9003$)	74.9	67.1	77.5	71.9
CONAN- α^5_3 ($\vartheta = -0.8985$)	73.6	75.0	66.7	70.6
CONAN- β^5_3 ($\vartheta = -0.9289$)	76.4	76.6	69.2	72.7
RoBERTa-large				
MANPAT $^{\Phi\Psi}$ (baseline)	83.9	84.8	70.1	76.7
MANPAT $^{\Phi}$ (baseline)	77.8	67.9	81.5	74.1
CONAN 5_2 ($\vartheta = -0.1315$)	85.9	81.7	78.1	79.9
CONAN 5_1 ($\vartheta = -0.9750$)	85.2	77.2	80.3	78.7
CONAN- α^5_3 ($\vartheta = -0.9212$)	84.4	82.2	74.9	78.4
CONAN- β^5_3 ($\vartheta = -0.9585$)	85.3	78.8	77.3	78.0

Table 3: Levy/Holt test. All baselines from (Schmitt and Schütze, 2021). See Table 2 for table format.

and Levy/Holt (Table 3), and across model sizes (base and large), CONAN 5_2 (using either α or β because they are identical for $k = 2$) outperforms all other models including the previous state of the art by Schmitt and Schütze (2021), who fine-tune RoBERTa both without patterns (NLI) and using handcrafted (MANPAT) or automatically retrieved corpus patterns (AUTPAT). We report their two best systems for each benchmark.

We take the performance increase with continuous patterns as a clear indicator that the flexibility offered by separating pattern tokens from the rest of the vocabulary allows RoBERTa to better adapt to the task-specific data even with only few labeled training instances in the challenging LliC task.

	AUC	P	R	F1
SherLiC train \rightarrow Levy/Holt test				
Schm&Schü (2021) (base)	38.4	52.7	57.1	54.8
Schm&Schü (2021) (large)	70.4	39.6	95.3	56.0
CONAN 5_2 (base)	54.3	38.2	89.8	53.5
CONAN 5_2 (large)	61.5	34.5	96.3	50.8
Levy/Holt train \rightarrow SherLiC test				
Schm&Schü (2021) (base)	63.3	62.8	68.4	65.5
Schm&Schü (2021) (large)	62.1	68.1	57.3	62.3
CONAN 5_2 (base)	64.9	63.4	68.9	66.1
CONAN 5_2 (large)	70.1	69.0	69.5	69.2

Table 4: Transfer experiments ($\vartheta = 0$). Best models from (Schmitt and Schütze, 2021) according to F1 score.

5 Analysis and Discussion

Nearest neighbors. To further investigate how RoBERTa makes use of the flexibility of C tokens, we compute their nearest neighbors in the space of original vocabulary tokens based on cosine similarity for our models in Tables 2 and 3. We always find the C tokens to be very dissimilar from any token in the original vocabulary, the highest cosine similarity being 0.15. And even among themselves, C tokens are very dissimilar, nearly orthogonal, with 0.08 being the highest cosine similarity here. RoBERTa seems to indeed take full advantage of the increased flexibility to put the C tokens anywhere in the embedding space. This further backs our hypothesis that the increased flexibility is beneficial for performance.

Influence of additional parameters. One might argue that the vocabulary extension and the resulting new randomly initialized token embeddings lead to an unfair advantage for CONAN models because the parameter count increases. While more parameters do generally lead to increased model capacity, the number of new parameters is so small compared to the total number of parameters in RoBERTa that we consider it improbable that the new parameters are alone responsible for the improved performance. Of all models in Tables 2 and 3, CONAN $^{50}_1$ introduces the most additional model parameters, i.e., $1 \cdot 50 \cdot 768 = 38400$ for RoBERTa-base. Given that even the smaller RoBERTa-base model still has a total of 125M parameters, the relative parameter increase is maximally 0.03%, which, we argue, is negligible.

Transfer between datasets. The experiments summarized in Table 4 investigate the hypothesis that CONAN’s better adaptation to the fine-tuning data might worsen its generalization abilities to other

LliC benchmarks. For this, we train our best model CONAN₂⁵ on SherLliC to test it on Levy/Holt and vice versa. In this scenario, we assume that the target dataset is not available at all. So there is no way to adapt to a slightly different domain other than learning general LliC reasoning. We thus set $\vartheta = 0$ in these experiments.

We find that with the very few train samples in SherLliC the risk of overfitting to SherLliC is indeed higher. When trained on Levy/Holt with around 4.4k train samples, however, CONAN clearly improves generalization to the SherLliC domain.

6 Related Work

PLMs and text patterns. GPT-2 (Radford et al., 2019) made the idea popular that a PLM can perform tasks without access to any training data when prompted with the right NL task instructions. With GPT-3, Brown et al. (2020) adapted this idea to few-shot settings where the task prompt is extended by a few training samples. While this kind of few-shot adaptation with a frozen PLM only works with very big models, Schick and Schütze (2021b) achieve similar performance with smaller models by fine-tuning the PLM on the available training data and putting them into NL templates. Recently, Schmitt and Schütze (2021) investigated the use of PLMs for LliC. Compared to a standard sequence classification fine-tuning approach, they were able to improve the PLM RoBERTa’s performance by putting an entailment candidate into textual contexts that only make sense for either a valid or invalid example. Patterns like “ y because x .” (valid) or “It does not mean that y just because x .” (invalid) make intuitive sense to humans and outperform standard RoBERTa on LliC.

A large problem with all these approaches, however, is to find well-functioning patterns, for which numerous solutions have been proposed (Shin et al., 2020; Haviv et al., 2021; Bouraoui et al., 2020; Jiang et al., 2020; Gao et al., 2021; Reynolds and McDonell, 2021). We argue that it is not optimal to constrain pattern search to the space of NL sequences if the primary goal is better task performance, and therefore abandon this constraint.

PLMs and continuous patterns. Li and Liang (2021) and Hambardzumyan et al. (2021) contemporaneously introduced the idea of mixing the input token embeddings of a PLM with other continuous vectors that do not correspond to vocabulary elements. In the spirit of GPT-2 (see above), they keep

the PLM’s parameters frozen and only fine-tune the embeddings of the “virtual tokens” to the target task. While this line of research offers certain appeals of its own, e.g., reusability of the frozen PLM weights, this is not the focus of our work. In pursuit of the best possible performance, we instead compare the use of continuous vs. NL patterns in the process of fine-tuning all PLM parameters and find that even carefully chosen NL patterns can be outperformed by our automatically learned ones.

Contemporaneously to our work, Liu et al. (2021) fine-tune entire PLMs with continuous patterns for SuperGLUE (Wang et al., 2019). Besides reformulating the SuperGLUE tasks as cloze tasks, while we keep formalizing our task as classification, Liu et al. (2021) also add more complexity by computing the continuous token representations with an LSTM (Hochreiter and Schmidhuber, 1997) and adding certain “anchor tokens”, such as a question mark, at manually chosen places. CONAN does not use any manual pattern design and embeds continuous tokens with a simple lookup table.

Another contemporaneous work by Lester et al. (2021) tests the influence of model size on the performance of a frozen PLM with trained continuous prompts. Their prompt ensembling is akin to our combining multiple patterns during inference (cf. §2). The key difference is that, instead of making predictions with different patterns and taking the majority vote, we rather compare the scores for different patterns to make our prediction.

7 Conclusion

We presented CONAN, a method that improves fine-tuning performance of a PLM with continuous patterns. CONAN does not depend on any manual pattern design and is efficient as the shortest possible patterns with good performance can be found automatically. It provides an automatic way of systematically testing structural properties of patterns, such as length, w.r.t. performance changes. In our experiments on two established LliC benchmarks, CONAN outperforms previous work using NL patterns and sets a new state of the art.

Acknowledgments

We gratefully acknowledge a Ph.D. scholarship awarded to the first author by the German Academic Scholarship Foundation (Studienstiftung des deutschen Volkes). This work was supported by the BMBF as part of the project MLWin (01IS18050).

References

- Tomas Mikolov et al. 2013. [Efficient estimation of word representations in vector space](#). *ArXiv preprint*, abs/1301.3781.
- Kurt Bollacker, Colin Evans, Praveen Paritosh, Tim Sturge, and Jamie Taylor. 2008. [Freebase: A collaboratively created graph database for structuring human knowledge](#). In *Proceedings of the 2008 ACM SIGMOD International Conference on Management of Data*, SIGMOD '08, page 1247–1250, New York, NY, USA. Association for Computing Machinery.
- Zied Bouraoui, José Camacho-Collados, and Steven Schockaert. 2020. [Inducing relational knowledge from BERT](#). 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 7456–7463. AAAI Press.
- Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. [Language models are few-shot learners](#). In *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*.
- Ido Dagan, Dan Roth, Mark Sammons, and Fabio Massimo Zanzotto. 2013. *Recognizing textual entailment: Models and applications*. Morgan & Claypool Publishers.
- 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.
- Avia Efrat and Omer Levy. 2020. [The turking test: Can language models understand instructions?](#) *ArXiv preprint*, abs/2010.11982.
- Yanai Elazar, Nora Kassner, Shauli Ravfogel, Abhilasha Ravichander, Eduard Hovy, Hinrich Schütze, and Yoav Goldberg. 2021. [Measuring and improving consistency in pretrained language models](#). *ArXiv preprint*, abs/2102.01017.
- Tianyu Gao, Adam Fisch, and Danqi Chen. 2021. [Making pre-trained language models better few-shot learners](#). In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 3816–3830, Online. Association for Computational Linguistics.
- Max Glockner, Vered Shwartz, and Yoav Goldberg. 2018. [Breaking NLI systems with sentences that require simple lexical inferences](#). In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 650–655, Melbourne, Australia. Association for Computational Linguistics.
- Karen Hambardzumyan, Hrant Khachatryan, and Jonathan May. 2021. [WARP: Word-level Adversarial ReProgramming](#). In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 4921–4933, Online. Association for Computational Linguistics.
- Adi Haviv, Jonathan Berant, and Amir Globerson. 2021. [BERTese: Learning to speak to BERT](#). In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, pages 3618–3623, Online. Association for Computational Linguistics.
- Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural Computation*, 9(8):1735–1780.
- Xavier R. Holt. 2018. Probabilistic models of relational implication. Master’s thesis, Macquarie University.
- Mohammad Javad Hosseini, Shay B. Cohen, Mark Johnson, and Mark Steedman. 2019. [Duality of link prediction and entailment graph induction](#). In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4736–4746, Florence, Italy. Association for Computational Linguistics.
- Zhengbao Jiang, Frank F. Xu, Jun Araki, and Graham Neubig. 2020. [How can we know what language models know?](#) *Transactions of the Association for Computational Linguistics*, 8:423–438.
- Diederik P. Kingma and Jimmy Ba. 2015. [Adam: A method for stochastic optimization](#). In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*.
- Teven Le Scao and Alexander Rush. 2021. [How many data points is a prompt worth?](#) In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 2627–2636, Online. Association for Computational Linguistics.

- Brian Lester, Rami Al-Rfou, and Noah Constant. 2021. [The power of scale for parameter-efficient prompt tuning](#). *ArXiv preprint*, abs/2104.08691.
- Omer Levy and Ido Dagan. 2016. [Annotating relation inference in context via question answering](#). In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 249–255, Berlin, Germany. Association for Computational Linguistics.
- Xiang Lisa Li and Percy Liang. 2021. [Prefix-tuning: Optimizing continuous prompts for generation](#). In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 4582–4597, Online. Association for Computational Linguistics.
- Xiao Liu, Yanan Zheng, Zhengxiao Du, Ming Ding, Yujie Qian, Zhilin Yang, and Jie Tang. 2021. [Gpt understands, too](#). *ArXiv preprint*, abs/2103.10385.
- 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*, abs/1907.11692.
- Nick McKenna, Liane Guillou, Mohammad Javad Hosseini, Sander Bijl de Vroe, and Mark Steedman. 2021. [Multivalent entailment graphs for question answering](#). *ArXiv preprint*, abs/2104.07846.
- Yehudit Meged, Avi Caciularu, Vered Shwartz, and Ido Dagan. 2020. [Paraphrasing vs coreferring: Two sides of the same coin](#). In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 4897–4907, Online. Association for Computational Linguistics.
- Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. [Language models are unsupervised multitask learners](#). Technical report, OpenAI, San Francisco, California, United States.
- Laria Reynolds and Kyle McDonell. 2021. [Prompt programming for large language models: Beyond the few-shot paradigm](#). *ArXiv preprint*, abs/2102.07350.
- Timo Schick and Hinrich Schütze. 2021a. [Exploiting cloze-questions for few-shot text classification and natural language inference](#). In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, pages 255–269, Online. Association for Computational Linguistics.
- Timo Schick and Hinrich Schütze. 2021b. [It’s not just size that matters: Small language models are also few-shot learners](#). In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 2339–2352, Online. Association for Computational Linguistics.
- Martin Schmitt and Hinrich Schütze. 2019. [SherLIiC: A typed event-focused lexical inference benchmark for evaluating natural language inference](#). In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 902–914, Florence, Italy. Association for Computational Linguistics.
- Martin Schmitt and Hinrich Schütze. 2021. [Language models for lexical inference in context](#). In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, pages 1267–1280, Online. Association for Computational Linguistics.
- Stefan Schoenmackers, Jesse Davis, Oren Etzioni, and Daniel Weld. 2010. [Learning first-order horn clauses from web text](#). In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, pages 1088–1098, Cambridge, MA. Association for Computational Linguistics.
- Taylor Shin, Yasaman Razeghi, Robert L. Logan IV, Eric Wallace, and Sameer Singh. 2020. [AutoPrompt: Eliciting Knowledge from Language Models with Automatically Generated Prompts](#). In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 4222–4235, Online. Association for Computational Linguistics.
- Vered Shwartz, Gabriel Stanovsky, and Ido Dagan. 2017. [Acquiring predicate paraphrases from news tweets](#). In *Proceedings of the 6th Joint Conference on Lexical and Computational Semantics (*SEM 2017)*, pages 155–160, Vancouver, Canada. Association for Computational Linguistics.
- Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. [Attention is all you need](#). 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 5998–6008.
- Alex Wang, Yada Pruksachatkun, Nikita Nangia, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R. Bowman. 2019. [Superglue: A stickier benchmark for general-purpose language understanding systems](#). In *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, pages 3261–3275.
- Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander M. Rush. 2019. [Huggingface’s transformers: State-of-the-art natural language processing](#). *ArXiv preprint*, abs/1910.03771.

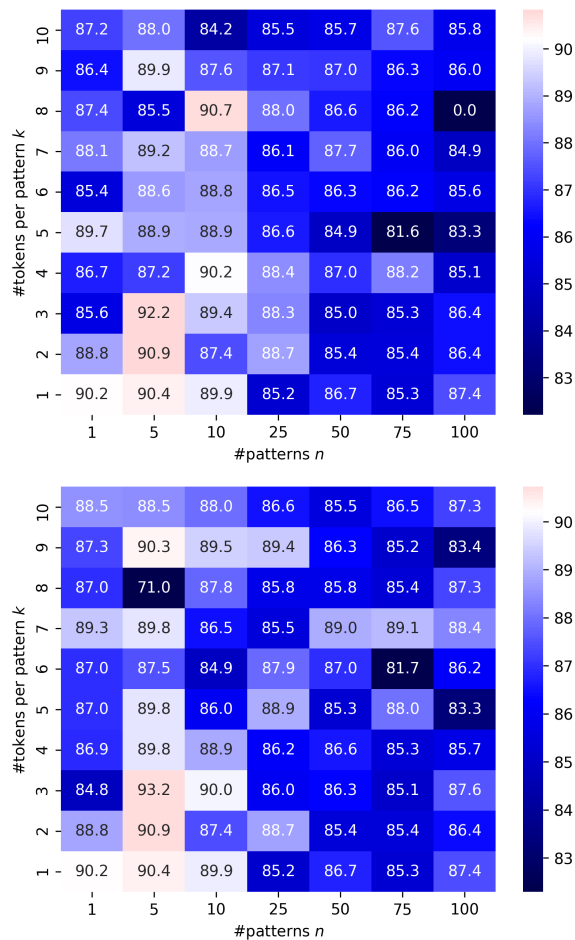


Figure 2: AUC on Levy/Holt dev for different CONAN models. Same format as Fig. 1.

A Training Details

We train our model for 5 epochs on a single GeForce RTX 2080 Ti GPU, with Adam (Kingma and Ba, 2015) and a mini-batch size of 10 (resp. 2) training instances for RoBERTa-base (resp. -large) and mini-batches of 5 patterns. We adopt well-functioning values from (Schmitt and Schütze, 2021) for all non-CONAN-specific hyperparameters, i.e., learning rate lr , weight decay λ , and accumulated batches c before a gradient update: Concretely, we set $lr = 2.28 \cdot 10^{-5}$, $\lambda = 6.52 \cdot 10^{-2}$, $c = 2$ for evaluating RoBERTa-base on SherLiC, $lr = 1.29 \cdot 10^{-5}$, $\lambda = 2.49 \cdot 10^{-4}$, $c = 3$ for RoBERTa-large on SherLiC, $lr = 2.72 \cdot 10^{-5}$, $\lambda = 1.43 \cdot 10^{-3}$, $c = 1$ for RoBERTa-base on Levy/Holt, and $lr = 4.55 \cdot 10^{-6}$, $\lambda = 3.90 \cdot 10^{-4}$, $c = 2$ for RoBERTa-large on Levy/Holt.

B More Dev Results

See Fig. 2 for evaluation results on Levy/Holt dev.