A smashed glass cannot be full: Generation of Commonsense Explanations through Prompt-based Few-shot Learning

Andrea Zaninello
Fondazione Bruno Kessler
Free University of Bolzano (Italy)
azaninello@fbk.eu

Bernardo Magnini
Fondazione Bruno Kessler
magnini@fbk.eu

Abstract

We assume that providing explanations is a process to elicit implicit knowledge in human communication, and propose a general methodology to generate commonsense explanations from pairs of semantically related sentences. We take advantage of both prompting applied to large, encoder-decoder pre-trained language models, and few-shot learning techniques, such as pattern-exploiting training. Experiments run on the e-SNLI dataset show that the proposed method achieves state-of-the-art results on the explanation generation task, with a substantial reduction of labelled data. The obtained results open new perspective on a number of tasks involving the elicitation of implicit knowledge.

1 Introduction

When exchanging information, it is typical to exclude details that appear self-evident or insignificant, so that only part of the message is articulated verbally while other details are implied (Becker et al., 2020, 2021a). This is particularly true for information involving commonsense knowledge, which represents the backbone of everyday communication and reasoning. For example, consider the following sentences:

(1) *The glass is broken into pieces.*

(2) *The glass is full.*

We can intuitively assert that these two sentences *contradict* each other, and if we were asked to explain *why*, our answer would most probably appeal to some *implicit knowledge* about the world ("A glass broken into pieces cannot contain any liquid" and "A glass cannot be broken and full of a liquid at the same time", etc.) to various degrees of depth. This implicit, shared knowledge can easily be inferred by people, but represents a challenge for computational systems. Crucially, a request for explanation is often a request to *make explicit* something that is only implied or omitted in conversation. While this is evident in everyday communication, it is also true for more specialised domains, for example a doctor-patient scenario where the patient is given a diagnosis ("Your clinical case and tests indicate that you have type 2 diabetes") and asks for explanations to the doctor ("*Why* do you believe that?").

Therefore, the underlying assumption of this paper is that commonsense explanations are to some extent based upon the notion of *implicitness*, and that the information they rely on can not be fully derived from their textual input alone. Being able to elicit commonsense implicit knowledge is a relevant step forward not only towards providing explanations, but also towards better natural language understanding.

In this work, we state the "implicit knowledge problem" as the capacity to automatically generate explanations regarding the semantic relations between two sentences. We define a general methodology to elicit implicit knowledge from pre-trained large language models, motivated by the intuition that they contain most of the knowledge needed, and are able to generalize over unseen instances. In fact, a fully-supervised fine-tuning would require a large training set where each input-sentence pair should be labeled with one or more explanations, a setting which is unrealistic in an open-domain scenario.

To achieve this, we propose a combination of prompting and few-shot learning techniques, which are well-suited to exploit the generative capabilities of language models and, at the same time, make use of limited supervision. Similar methods have been applied to some popular NLP tasks such as text classification, inference, summarization (Schick and Schütze, 2021a,b) and, more recently, to teach language models to leverage external tools via APIs (Schick et al., 2023). However, the generation of implicit knowledge poses a further challenge

Proceedings of the 1st Workshop on Natural Language Reasoning and Structured Explanations (@ACL 2023), pages 18–29

to these techniques as it requires to generate text based on information or reasoning structures that are outside the input texts[1].

For these reasons, in our experiments, we select the e-SNLI dataset Camburu et al. (2018), where sentence pairs traditionally employed for a textual inference task (pairs are labelled with *entail*, *contradict* or *neutral* tags) were augmented with explanations for the relation tag, collected through crowd-sourcing. We compare three generation methods: *unsupervised*, *fine-tuning* and *ensembling*, showing that the *ensembling* method achieves the best results, en-par with state-of-the-art while making use of limited supervision. Moreover, we find that this method mitigates the potential negative effects of "bad" prompts, which is a desirable feature whenever prompt optimization is not possible. However, we also underline that evaluation and comparison with SOTA results is still critical, as evaluation metrics are not usually directly comparable on the explanation generation task.

The innovative contributions of this paper are the following:

- We propose a general methodology to elicit implicit knowledge from language models with very limited training data.
- We compare the effects of prompting, few-shot fine-tuning and ensembling on a set of different language models, indicating which strategy suits best for each type.
- We show that one of our proposed methods for implicit knowledge generation is able to mitigate the negative impact of badly designed prompts.

The paper is structured as follows: Section 2 provides relevant background on language modelling and elicitation of implicit knowledge. Section 3 introduces the general approach to implicit knowledge generation. Sections 4 and 5 report, respectively, the experimental setting and the results we have obtained on the e-SNLI dataset. Finally, Section 6 provides relevant context of recent approaches to implicit knowledge elicitation.

[1]The code is available at github.com/andreazaninello/explanationgeneration

2 Background

2.1 Language modelling and transfer learning

Recent advances in NLP, particularly in Natural Language Generation (NLG), have been driven by the success of transfer learning techniques applied to neural language models, pre-trained on very large textual corpora in a self-supervised fashion (Howard and Ruder, 2018; Radford et al., 2019). These general models can be trained on in-domain datasets or on specific downstream tasks with much less resource expense through fine-tuning.

Transformers (Vaswani et al., 2017), based on the *attention* mechanism, currently represent the state-of-the-art in most of NLP tasks. In our experiments, we employ Transformers' encoder-decoder (sequence-to-sequence) models, like BART (Lewis et al., 2020), T5 (Raffel et al., 2020) or PEGASUS (Zhang et al., 2020). In these models, encoder attention can access the whole initial sequence, while decoder attention can only attend to previous words; these models perform best on language generation tasks that depend on a sequential input, such as machine translation or text summarization thus we hypothesize that the encoder-decoder kind of models are particularly suited to the task of generating explanations, because the sequence to be generated (the *explanans*) has to be conditioned on a full input sequence (the *explanandum*).

2.2 Prompting

The core technique that we use to elicit implicit knowledge is prompting (see Liu et al. (2021) for an extensive survey). Prompting consists in reframing an NLP task as a language modelling task: from a practical viewpoint, it corresponds to feeding a very large language model an input prompt that describes the desired task in natural language (and/or gives some examples of the desired output), and constructing a function that maps the desired label (e.g., *positive* in a sentiment analysis task) onto a series of natural language verbalizers (e.g. *good, great, excellent*). Given this prompt as an input, the language model is let generate the output as if it were a language modelling task such as next word or masked word prediction.

This new trend, which is especially appealing as it requires much less training signal compared to regular fine-tuning, has led to a shift from *objective engineering* to *prompt engineering*: this includes both the manual design of templates (Petroni et al., 2019) and automatic prompt learning (Jiang

et al., 2020), as well as various options to ensemble (Schick and Schütze, 2021b) and compose (Han et al., 2022) multiple prompts.

When using prompts, we can simply generate with a language model with no parameter update, giving the model some answered prompts as example to direct generation, or we can update the prompts' parameters through the supervision given by training examples (Liu et al., 2021); additionally, some promising recent approaches have attempted to apply few-shot training techniques based on prompting to language generation, too. Schick and Schütze (2021c), for example, propose a method called *pattern-exploiting training* (PET) and use hand-crafted patterns as task instructions to train intermediate models, in combination with a small set of unlabeled gold training examples which they use to ensemble those models. This technique has proved successful in performing few-shot downstream NLG tasks on sequence-to-sequence models (Zhang et al., 2020) with many fewer training examples than state-of-the-art benchmarks, especially when the generated output is tightly connected to the input (e.g. text summarization). While our method is closely inspired by this line of work, to the best of our knowledge this has not been applied to tasks where models need to use implicit information outside the input text.

2.3 Evaluating generation

As we anticipated, we model the "implicit knowledge problem" as the capacity to automatically generate explanations regarding the semantic relations between two sentences, which brings up the issue of evaluating the quality of the generated predictions. Unlike classification and regression tasks, the evaluation of generation is known to be critical, and has often relied on human judgments (Wiegreffe et al., 2022), which is however expensive and difficult to scale.

On the other hand, automatic assessment is usually done by measuring the overlap between at least one reference generation and the system's output, as with popular metrics like ROUGE (Lin, 2004) and BLEU (Papineni et al., 2002). Other metrics aim to measure the semantic similarity between generations and references using contextualised embeddings, for example BERT-Score (Zhang et al., 2019) or Sentence-BERT (Reimers and Gurevych, 2019), which consider word and, respectively, sentence-embeddings.

Nevertheless, evaluating generation is a critical aspect and there is still no consensus on what metric is to be taken as a reference for system comparison, especially on explanation generation. In our experiments, we align with our selected benchmark to facilitate results' comparison, and thus report BLEU, ROUGE and BERT-Scores. However, we are aware that results may not be fully conclusive in terms of comparison with SOTA, and that metrics are rather to be taken as a relative indication of systems' performance.

3 Methodology

Our proposed methods belong to the "fixed-prompt language model tuning" types (Liu et al., 2022): the LM's parameters are updated through few-shot fine-tuning, after both training and test examples have been modified by textual templates, meaning that the prompts' parameters are fixed while the model's parameters are updated.

While some recent studies aim at discovering the best prompts or performing prompt optimization (Li et al., 2021; Shin et al., 2020), our methods aim to provide a general framework that may work well even without prompting optimization, minimizing the negative impact of "bad" prompts. Finding the best patterns and prefixes would clearly contribute to the task but falls outside the scope of this study, and would be task dependent, while we aim to provide a general framework potentially applicable to different generation tasks.

Our method is closely related to Gao et al. (2021)'s, who apply a similar procedure to classification and regression tasks, and is inspired by Schick and Schütze (2021b)'s GenPET, who apply it to a set of generation tasks close to summarization, not involving the elicitation of implicit external knowledge.

3.1 Problem formalization

We state the "implicit knowledge generation" problem as the task of automatically producing an explanation for the semantic relation between a pair of textual sentences t_a and t_b. We set M to be a language model of vocabulary V, pre-trained on a masked language modelling task. We define an input $x \in \mathcal{X}$, an output sequence $y \in \mathcal{Y}$, a label $l \in \mathcal{L}$, a label verbalizer $v \in V^*$ that maps the labels to a natural language sequences (for example mapping a `neutrality` label to the sequence "does not entail"), and the sequence result-

ing from applying a prompt $p \in P$ to input x as $z = f_{prompt}(x, p)$, with z containing one masked sequence <mask>. The f_{prompt} function then simply takes the input texts t_a, t_b, and the verbalized label l, and returns a modified, natural language version of the input.

To obtain the prompts, we need to define (1) a set of task *prefixes* $(pref_1, ..., pref_n)$ that introduce the explanation $y \in \mathcal{Y}$, and are either processed as final part of the input or pre-pended to the reference explanation at training time (or to the mask sequence at test time), and (2) a set of *patterns* $(patt_1, ..., patt_m)$, which we combine with $(pref_1, ..., pref_n)$, resulting in $n \times m$ prompts. Prompts are used to rewrite each input example, by sampling randomly across all labels. We refer to this modified training set as T, which is $n \times m$ times the size of X. We summarize and give an example of our prompting design in Section A.1.

3.2 Training objective

The models we use are pre-trained on masked language modelling task, so their objective is calculating the probability of $p_M = (\mathbf{y}|\mathbf{z})$. We have the choice of (1) processing the task prefix $pref$ using the decoder, as part of the generated sequence, or (2) with the encoder, as part of the input. Thus, assuming some model M, some task prefix $pref_i$ and some pattern $patt_j$, the model needs to compute the probability of y as follows, respectively:

$$p(\mathbf{y}|\mathbf{x}) = p_M(pref_i; y|patt_j(x)) \quad (1)$$

$$p(\mathbf{y}|\mathbf{x}) = p_M(y|patt_i(x); pref_i) \quad (2)$$

Schick and Schütze (2021b) indicate that processing it with the decoder has a stronger impact on generations, and we apply it to BART and Pegasus, as in (1). However, this is not possible with T5 as encoding it with the decoder pushes the model to produce empty strings. With T5, we therefore encode it as part of the encoder (2). Modified training and test instances are used in different zero- and few-shot training configurations, namely UNSUPERVISED (3.3), FINE-TUNING (3.4) and ENSEMBLING (3.5). We synthesize the three methods in Figure 1.

3.3 Method 1: UNSUPERVISED

In a first zero-shot configuration, which we name UNSUPERVISED and we take as our baseline, we simply evaluate the model's predictions without any training, so no parameter update is performed, nor do we use any training instances. However, to have the prompts influence the model's generations at inference, we modify each test instance with each one of the prompts, as explained in Section 3.1. We evaluate the model's predictions for each prompt separately after modifying the test instances with that prompt through function f_{prompt}, as detailed in Section 4.4. Moreover, we define a null prompt p_0 for which both $pref$ and $patt$ are an empty string, resulting in the simple concatenation of x and the <MASK> token, and evaluate its generations.

3.4 Method 2: FINE-TUNING

In a second configuration, which we call FINE-TUNING, at training time we apply all the prompts p to every input through function f_{prompt} obtaining a dataset T of all training instances X (where X can be empty). In addition, in order to avoid overfitting and ensure regularization, we also take a set of 1000 unlabeled instances U (for which the explanation y is not given), modify them with the patterns $p \in P$ and have the untrained model M generate an output for each of them. We therefore obtain a synthetically generated dataset $T_{\text{FINE-TUNED}}$, which we append to our training instances.

We use T to fine-tune M with teacher forcing, by minimizing the cross-entropy between the model's prediction and the target sentences, obtaining a fine-tuned model $M_{\text{FINE-TUNED}}$. In a zero-shot setting, only U is used for fine-tuning. In few-shot setting, we use training sets of increasing size. As in method 1 (3.3), at test time we assess the predictions of each pattern separately and of all patterns together using p_0.

3.5 Method 3: ENSEMBLING

In the previous method we were able to assess each prompt separately at inference time. However, it may not always be possible to know which pattern works better for a certain task and poor performance of one prompt could hurt the overall performance of the fine-tuned model[2]. Moreover, in few-shot scenarios, models tend to overfit the training data or copy part of the original input, resulting

[2]We are aware that several methods for prompt search and optimization have been recently proposed, and our method would certainly benefit from better quality prompts. However, our aim is to mitigate the potential impact of bad prompts, while prompt search currently falls outside the scope of this study.

in poor quality of the generations. In this configuration, we then aim to generalize over all possible patterns given at training time, without having to choose a specific pattern at test time.

Therefore, we perform a form of knowledge distillation through prompt-ensembling by taking the fine-tuned model $M_{\text{FINE-TUNED}}$ and the unlabeled instances $U = (u_1, ..., u_n)$, and use the set of patterns in $P = (p_1, ..., p_m)$ to generate a set of candidate outputs $C = (y_1, ..., y_m)$ for each $u \in U$. Then, we modify the instances in U with the null pattern p_0 and ask an untrained model M (that did not see any of the training examples) to assign a probability to each candidate generation in C given the modified inputs from U. To assign a final score to the generation, we take the exponentiated average of all the log-likelihood assigned by the untrained model across all patterns, and take the best scoring generation as our prediction.

By doing so, we obtain a new dataset T_{ENSEMBLE}, where inputs are u's from U modified by p_0 and y's are the best ranking y's from C according to M. The so obtained explanations should not be biased towards one particular pattern. Moreover, we empirically set a cutoff lower threshold at the bottom 20% of the instances ranked by their probability, so that low quality explanations are discarded.

We use this final dataset T_{ENSEMBLE} to fine-tune a final model M_{ENSEMBLE} with a procedure similar to method 2, but we only evaluate it using the null prompt as explained in Section 4.4.

4 Experiments

4.1 Pre-trained language models

We experiment on three different Transformer-based encoder-decoder language models: BART, Pegasus, and T5.

Bart (Lewis et al., 2020) uses a standard sequence to sequence architecture with a bidirectional encoder and a left-to-right decoder. It is pre-trained by firstly corrupting text with a noising function, then learning a model able to reconstruct the original text. BART was evaluated on several benchmarks and proved particularly suitable for generation tasks, a reason why we decided to employ it. In all experiments, we use the BART-large model.

Pegasus (Zhang et al., 2020) has a similar architecture to BART and trained in a self-supervised way by masking important sentences in text and have the model generate them as a single output conditioned on the remaining sentences. Thus, its training objective is similarl to a summarization task. We use the PEGASUS-large implementation in all our experiments.

T5 (Raffel et al., 2020) is also an encoder-decoder model, however unlike the previous two it was pre-trained on a mix of NLP tasks prompted in a text-to-text format, where inputs and outputs are text strings, as opposed to BERT-style models. For this reason, it is particularly suitable for methods exploiting textual verbalizations to condition generations.

4.2 Dataset

For all our experiments, we use the e-SNLI dataset (Camburu et al., 2018), an extension of the Stanford Natural Language Inference (SNLI) dataset (Bowman et al., 2015) enriched with crowd-sourced natural language explanations. The SNLI dataset is an influential dataset widely used for the task of Recognizing Textual Entailment (RTE) (Dagan et al., 2005): given two text fragments (called *premise* and *hypothesis)*, the aim of RTE is deciding whether the premise entails, contradicts or neither entails nor contradicts the hypothesis, labelling the relationship between the two texts with an *entailment*, *contradiction*, or *neutrality* label. The SNLI dataset contains 570K premise-hypothesis pairs, evenly distributed across labels. E-SNLI contains an extra layer of information, represented by a crowd-sourced natural language explanations for each instance for the training, testing and development splits. An example is given below.

```
{"guid": "test-3",
  "idx": "3",
  "label": "NEUTRALITY"
  "meta": {}
  "explanations": ["the woman
     could've been old rather
     than young"]
  "premise": "'A woman with a
     green headscarf, blue shirt
     and a very big grin.'"
  "hypothesis": 'The woman is
     young.'}
```

Few other datasets exist, such as the CoS-E dataset (Rajani et al., 2019) (an expansion of the CommonsenseQA (Talmor et al., 2019) dataset) that provide explanations based on commonsense. However, CoS-E's explanations mainly focus on one single term or phrase as the given explanation refers to

one of the five possible answers to a commonsense question. Moreover, many times the open-ended explanations in this dataset are not formed as full sentences, while we aim to generate self-contained, linguistically complete explanations, so this dataset is not suitable for our experiments.

On the other hand, the task of recognizing textual entailment involves general reasoning as well as understanding some subtle facts about the language and the referents, while heavily relying on commonsense knowledge. For this reasons, many studies which aim to investigate the ability of a system to elicit implicit knowledge have turned onto this dataset, and we choose to experiment our methods on this challenging dataset enriched with natural language explanations.

For each training instance x (and for each corresponding output y), we obtain a set of four patterns. As can be seen from table in Section A.1, odd patterns present the explanation after the two sentences, while even patterns presents the explanation before the sentences, which are then introduced by the "`Text:`" string.

4.3 Experimental setup

For the FINE-TUNING and ENSEMBLING methods, we experiment with training sizes $= (0, 10, 100, 500)$, test and unlabeled sizes $= 1000$, sampled uniformly across the original examples and labels. For UNSUPERVISED, we set $T = 0$ and do not perform any parameter update, we simply let the model generate given the modified inputs. For FINE-TUNING and ENSEMBLE we train on a single NVIDIA GeForce RTX GPU with 10GB RAM for 3 epochs, for about 15 hours. We optimize with Adafactor with square root learning rate decay, dropout rate $= 0.1$ and learning rate $= 10^4$, following (Schick and Schütze, 2021b). We train each model with 8 gradient accumulation steps using a batch size of 2 as our computing resources are limited, and generate using greedy decoding. For all models, we use the Pytorch Transformers library implementation (Wolf et al., 2019).

4.4 Evaluation

At test time, we evaluate each model on each pattern separately, by modifying each testing instance with one prompt at a time. Secondly, as in real-world scenarios it is not always possible to know in advance which patterns will perform better, we also evaluate each model on the null pattern p_0, where the input precedes the masked sequence token and we use an empty task prefix. We evaluate using common metrics for generation, comparing the predicted output with the reference explanation, and thus report BLEU-1, ROUGE-1 and BERT-Score. Kayser et al. (2021) present the most extensive, current study on the correlation between human judgments and generation, focusing in particular on the explanation generation task, and show that BERT-Score is the one that best matches human judgments, as also confirmed by other studies (Becker et al., 2021b). Therefore, we set BERT-Score as our reference metric to assess the best model and method.

5 Results and Discussion

In Table 1 we report the results of our experiments for each of the considered models. T5 represents the best scoring model, which most benefits from the proposed methods, achieving a BERT-score of 91.23 both for the P_0-FINE-TUNING method and for the ENSEMBLING method, as confirmed by both BLEU and ROUGE scores. Similarly, Pegasus benefits from the proposed methods, with a slight decrease on the ENSEMBLING method, which may be due to the error margin of the metrics.

Interestingly, both models have very low scores in both zero-shot configurations, indicating that the "fixed-prompt fine-tune" strategies may be particularly suitable for them, even without prompt optimization, as indicated by the low figures of the null prompt. On the other hand, BART displays a stronger baseline and is more sensitive to prompt design, as displayed by the decreasing values especially relevant on the p_0-FINE-TUNING method. This indicates that when using BART, which was not specifically trained to accept prompts as inputs, prompt optimization may be a better strategy. On the other hand, for all three models the ENSEMBLING method is able to mitigate the negative effects of shallow prompt design.

While a clear benchmark for the explanation generation task does not yet exist, in Table 2 we report the results of related studies on the same task and dataset. Specifically, we compare our methods with studies using the same underlying model and with comparable settings and show that our methods achieve better results but with a significant reduction in training size. In particular, Marasovic et al. (2022) use T5 with 48 training examples and achieve a significantly lower Bert-Score compared to our T5-Ensembling method with 10

training examples. Becker et al. (2021b) achieve a slightly lower BERT-Score using BART, but with 18K training examples (which we compare with our BART-Ensemble method with 500 training examples). We provide further details on the related works in Section 6.

We also manually analyzed several generations and compared the different models and methods. We notice that in the unsupervised settings, models tend to hallucinate, while with zero-shots BART and PEGASUS tend to copy (part of) the input, while T5 often returns single words. In Table 3 we report an example of the generations produced by T5 under the different configurations. Notice that generations under settings T_{10} and T_{100} are already correct. However, the bigger the train size, the closer the generation to the reference. Finally, manual inspection also highlighted that in some cases the model learns to reproduce some patterns, such as the contradiction explanation pattern "X cannot Y and Z at the same time", where X is the common referent to the sentences and Y and Z are the states described by the two sentences, respectively. However, the pattern repetition also characterises many human-generated sentences, a phenomenon that deserves further attention in future investigations if we aim at general, better natural language explanations.

6 Related work

Being a relatively recent area of interest, generation of free text explanations is not a well consolidated task. Particularly, evaluations metrics are still being discussed (Golovneva et al., 2022) (Wiegreffe et al., 2022), attempting both to capture the explanation informativeness and to improve the correlation toward human judgements. Here we report related works which are most focused on the e-SNLI dataset, being more comparable with our work (see Table 2).

Generating explanation with implicit knowledge has traditionally been addressed either by constraining generations with general knowledge paths (Ribeiro et al., 2020), by fine-tuning on specific or general knowledge datasets (Fatema Rajani et al., 2019), or with a combination of both methods (Becker et al., 2021a).

Camburu et al. (2018) train four different models with the aim to generate an explanation given only the hypothesis, generate an explanation without knowing the label, jointly predict a label and generate an explanation for the predicted label, and generate an explanation and then predict the label. Their work uses straightforward recurrent neural network architectures so it is does not achieve state-of-the-art results, but it establishes a strong baseline.

Becker et al. (2021a) generate implicit knowledge connecting sentences in text, similarly to Camburu et al. (2018). They perform fine-tuning on corpora enriched with implicit information, by constraining them with relevant concepts and connecting commonsense knowledge paths, combining data augmentation and graph-to-text methods.

Marasovic et al. (2022) both present FEB, a standardized collection of four existing English-language datasets and associated metrics, and results based on template-based prompting. In our work we show that specific prompting design for the e-SNLI task results in significant improvements with respect to more general purposes prompts.

Li et al. (2022), based on the intuition that explanation generated through single-pass prompting often lacks sufficiency and conciseness, propose a two-step approach where the first-pass output from the pretrained language model is polished, and then regenerated retaining the information that supports the contents being explained.

Ye et al. (2022) show that both the computation trace (the way the explanation is decomposed) and the natural language of the prompt, contribute to the effectiveness of explanations. According to this finding they propose automatic prompt selection that focus on prompt diversity, rather than complementarity only.

7 Conclusion

In this work, we argued that providing explanations is often a process of eliciting implicit knowledge. We proposed a general methodology to generate commonsense explanations from pairs of semantically related sentences, taking advantage of both prompting applied to large pre-trained language models and few-shot learning techniques. Experiments run on the e-SNLI dataset show that the proposed methods achieve SOTA results on the explanation generation task, with a substantial reduction of labelled data. The obtained results open new perspective for a number of tasks based on eliciting implicit knowledge.

	UNSUPERVISED (baseline)			FINE-TUNING		ENSEMBLING
	P_0	P_1 - P_4 (best)		P_0	P_1 - P_4 (best)	P_0
BART	11.10 42.30 88.61	10.86 42.05 88.53 (1)	T_0	11.09 42.3 **88.53**	10.86 42.05 88.53 (1)	11.12 42.37 88.58
			T_{10}	04.50 22.85 88.14	04.60 23.09 88.18 (1)	04.94 23.45 88.43
			T_{100}	10.86 31.87 77.09	12.27 35.82 **86.88** (4)	14.76 38.00 90.13
			T_{500}	08.09 28.23 66.57	11.88 35.68 86.64 (3)	15.06 39.45 **90.27**
PEGASUS	01.66 26.67 85.00	03.24 30.85 87.66 (3)	T_0	05.44 29.69 87.05	10.06 35.00 88.58 (1)	10.86 38.70 88.76
			T_{10}	10.58 38.06 88.38	10.50 37.93 88.42 (1)	10.59 38.66 88.52
			T_{100}	14.42 40.21 90.42	14.58 40.39 90.47 (4)	15.60 41.90 **90.69**
			T_{500}	16.87 42.30 **90.79**	16.75 42.26 **90.77** (4)	16.39 41.58 90.67
T5	05.45 27.29 85.59	04.78 25.41 84.68 (3)	T_0	05.45 27.29 85.59	04.79 24.72 84.76 (1)	06.25 11.72 84.95
			T_{10}	18.91 42.45 90.99	18.95 42.64 91.01 (2)	20.45 43.87 91.20
			T_{100}	17.62 41.96 90.91	17.63 41.67 90.91 (3)	18.37 42.06 90.96
			T_{500}	20.15 44.67 **91.23**	19.84 44.59 **91.21** (1)	20.18 44.00 **91.23**

Table 1: Results for the three sets of experiments for the considered language models in each training configuration ($T_0 - T_{500}$). $P_0 - P_{1-4}$ indicate the prompt used to modify the test input in that configuration. For prompts $P1 - 4$ we report the best scoring prompt, indicated in braket (1-4). For each experiment, we report the BLEU-1, ROUGE-1 and BERT-Scores in this order. Boldfaced, the best scoring configurations for each model, method and test prompt according to BERT-score.

Reference	BLEU	ROUGE	BERT-Score	Training size	Model
Becker et al. (2021b)	12.71	47	90	18K	BART
Our method	15.06	39.45	**90.27**	500	BART
Marasovic et al. (2022)	n/a	n/a	79.2	48	T5
Our method	20.45	43.87	**91.20**	10	T5
Li et al. (2022)	22.3	n/a	87.16	500K	GPT2
Ye et al. (2022)	n/a	n/a	83.9	500K	RoBERTa
Camburu et al. (2018)	27.58	n/a	n/a	500K	From scratch

Table 2: Benchmark for the task of generating explanations on the e-SNLI dataset.

T5-ENSEMBLING	Generated explanation on test set with P_0
Input	'A man with an afro and bandanna playing electric guitar.' contradicts 'the guy with the afro is eating spinach'
T_0	False
T_{10}	The guy is either playing electric guitar or eating spinach.
T_{100}	the man is either playing electric guitar or eating spinach.
T_{500}	A man cannot be playing electric guitar and eating spinach at the same time.
Reference	The man can not very well be playing electric guitar and eating spinach at the same time.
Baseline	: 'A man with an afro and bandanna playing electric guitar' : 'A man with an afro and bandanna playing electric guitar' contradicts 'the guy with an afro is eating spinach'. :"'A man with an

Table 3: Example generations for the best scoring model and method T5-ENSEMBLING. We report the generated explanation for each configuration, the reference explanation, and the T5-UNSUPERVISED's prediction (baseline).

8 Limitations

Although we showed significant improvements in explanation generation using prompt-based few-shot learning, our work still has some limitations. First, we experimented only on the e-SNLI dataset: although e-SNLI is a reference for the task, it would be interesting to extend the proposed methodology

to other datasets with natural language explanations (see Wiegreffe and Marasović (2021) for an extensive review).

Second, we did not attempt to automatic prompt optimization: although this may bring further minor improvements, we decided to leave optimization to a next step, as it does not change the core contribution of our work.

Third, we believe there is an intrinsic limitation in comparing our results with SOTA, as there is not a clear consensus on which metric is to be taken as the reference metric for benchmarking, along with the fact that measures sometimes disagree on scoring one system better than another. We hope that in the future this task and its evaluation will consolidate into a shared benchmark.

Finally, as for our our use of e-SNLI, we are assuming that for all sentence pairs in the dataset there is an implicit explanation of the semantic relation between the sentences. Under this assumption we always generate an explanation, even when the explanation is already explicit in one of the sentences. We think that a better capacity to detect those cases would bring relevant insight to our approach.

Acknowledgements

This work has been partially supported by the chistera ANTIDOTE Project.

References

Maria Becker, Katharina Korfhage, and Anette Frank. 2020. Implicit knowledge in argumentative texts: An annotated corpus. In *Proceedings of the 12th Language Resources and Evaluation Conference*, pages 2316–2324.

Maria Becker, Siting Liang, and Anette Frank. 2021a. Reconstructing implicit knowledge with language models. In *Proceedings of Deep Learning Inside Out (DeeLIO): The 2nd Workshop on Knowledge Extraction and Integration for Deep Learning Architectures*, pages 11–24.

Maria Becker, Siting Liang, and Anette Frank. 2021b. Reconstructing implicit knowledge with language models. In *Proceedings of Deep Learning Inside Out (DeeLIO): The 2nd Workshop on Knowledge Extraction and Integration for Deep Learning Architectures*, pages 11–24, Online. 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 (EMNLP)*. Association for Computational Linguistics.

Oana-Maria Camburu, Tim Rocktäschel, Thomas Lukasiewicz, and Phil Blunsom. 2018. e-snli: Natural language inference with natural language explanations. *Advances in Neural Information Processing Systems*, 31.

Ido Dagan, Oren Glickman, and Bernardo Magnini. 2005. The pascal recognising textual entailment challenge. In *Proceedings of the PASCAL Challenges Workshop on Recognising Textual Entailment*.

Nazneen Fatema Rajani, Bryan McCann, Caiming Xiong, and Richard Socher. 2019. Explain yourself! leveraging language models for commonsense reasoning. *arXiv e-prints*, pages arXiv–1906.

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.

Olga Golovneva, Moya Chen, Spencer Poff, Martin Corredor, Luke Zettlemoyer, Maryam Fazel-Zarandi, and Asli Celikyilmaz. 2022. Roscoe: A suite of metrics for scoring step-by-step reasoning.

Xu Han, Weilin Zhao, Ning Ding, Zhiyuan Liu, and Maosong Sun. 2022. Ptr: Prompt tuning with rules for text classification. *AI Open*, 3:182–192.

Jeremy Howard and Sebastian Ruder. 2018. Universal language model fine-tuning for text classification. *arXiv preprint arXiv:1801.06146*.

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.

Maxime Kayser, Oana-Maria Camburu, Leonard Salewski, Cornelius Emde, Virginie Do, Zeynep Akata, and Thomas Lukasiewicz. 2021. e-vil: A dataset and benchmark for natural language explanations in vision-language tasks. pages 1224–1234.

Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer. 2020. Bart: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7871–7880.

Qintong Li, Zhiyong Wu, Lingpeng Kong, and Wei Bi. 2022. Explanation regeneration via information bottleneck. *arXiv preprint arXiv:2212.09603*.

Xiaotao Li, Shujuan You, Yawen Niu, and Wai Chen. 2021. Learning embeddings for rare words leveraging Internet search engine and spatial location relationships. In *Proceedings of *SEM 2021: The Tenth Joint Conference on Lexical and Computational Semantics*, pages 278–287, Online. Association for Computational Linguistics.

Chin-Yew Lin. 2004. Rouge: A package for automatic evaluation of summaries. In *Text summarization branches out*, pages 74–81.

Fangchao Liu, Hongyu Lin, Xianpei Han, Boxi Cao, and Le Sun. 2022. Pre-training to match for unified low-shot relation extraction. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 5785–5795, Dublin, Ireland. Association for Computational Linguistics.

Pengfei Liu, Weizhe Yuan, Jinlan Fu, Zhengbao Jiang, Hiroaki Hayashi, and Graham Neubig. 2021. Pre-train, prompt, and predict: A systematic survey of prompting methods in natural language processing. *ACM Computing Surveys (CSUR)*.

Ana Marasovic, Iz Beltagy, Doug Downey, and Matthew Peters. 2022. Few-shot self-rationalization with natural language prompts. In *Findings of the Association for Computational Linguistics: NAACL 2022*, pages 410–424, Seattle, United States. Association for Computational Linguistics.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting of the Association for Computational Linguistics*, pages 311–318.

Fabio Petroni, Tim Rocktäschel, Patrick Lewis, Anton Bakhtin, Yuxiang Wu, Alexander H Miller, and Sebastian Riedel. 2019. Language models as knowledge bases? *arXiv preprint arXiv:1909.01066*.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. 2019. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research*, 21:1–67.

Nazneen Fatema Rajani, Bryan McCann, Caiming Xiong, and Richard Socher. 2019. Explain yourself! leveraging language models for commonsense reasoning. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4932–4942.

Nils Reimers and Iryna Gurevych. 2019. Sentence-bert: Sentence embeddings using siamese bert-networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3982–3992.

Leonardo FR Ribeiro, Martin Schmitt, Hinrich Schütze, and Iryna Gurevych. 2020. Investigating pretrained language models for graph-to-text generation. *arXiv preprint arXiv:2007.08426*.

Timo Schick, Jane Dwivedi-Yu, Roberto Dessì, Roberta Raileanu, Maria Lomeli, Luke Zettlemoyer, Nicola Cancedda, and Thomas Scialom. 2023. Toolformer: Language models can teach themselves to use tools.

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. Few-shot text generation with natural language instructions. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 390–402, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Timo Schick and Hinrich Schütze. 2021c. Generating datasets with pretrained language models. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 6943–6951, Online and Punta Cana, Dominican Republic. 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.

Alon Talmor, Jonathan Herzig, Nicholas Lourie, and Jonathan Berant. 2019. CommonsenseQA: A question answering challenge targeting commonsense knowledge. 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 4149–4158, Minneapolis, Minnesota. Association for Computational Linguistics.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *Advances in neural information processing systems*, 30.

Sarah Wiegreffe, Jack Hessel, Swabha Swayamdipta, Mark Riedl, and Yejin Choi. 2022. Reframing human-AI collaboration for generating free-text explanations. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language*

Technologies, pages 632–658, Seattle, United States. Association for Computational Linguistics.

Sarah Wiegreffe and Ana Marasović. 2021. Teach me to explain: A review of datasets for explainable nlp. *ArXiv*, abs/2102.12060.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, et al. 2019. Huggingface's transformers: State-of-the-art natural language processing. *arXiv preprint arXiv:1910.03771*.

Xi Ye, Srinivasan Iyer, Asli Celikyilmaz, Ves Stoyanov, Greg Durrett, and Ramakanth Pasunuru. 2022. Complementary explanations for effective in-context learning. *arXiv preprint arXiv:2211.13892*.

Jingqing Zhang, Yao Zhao, Mohammad Saleh, and Peter Liu. 2020. Pegasus: Pre-training with extracted gap-sentences for abstractive summarization. In *International Conference on Machine Learning*, pages 11328–11339. PMLR.

Tianyi Zhang, Varsha Kishore, Felix Wu, Kilian Q Weinberger, and Yoav Artzi. 2019. Bertscore: Evaluating text generation with bert. *arXiv preprint arXiv:1904.09675*.

A Appendix

A.1 Prompt design

Parameter	Values
Prefixes ($pref$)	`"Explanation:", "Rationale:"`
Task verbalizers (v)	`"entails", "contradicts", "does not entail"`
Patterns ($patt$)	$patt_1, patt_3 = t_a + v + t_b + pref +$ `<mask>`
	$patt_2, patt_4 =$ $pref +$ `<mask>` + `"Text:"` $+ t_a + v + t_b$

Table 4: Synthesis of the possible values of each of prompt parameters.

A.2 System diagrams

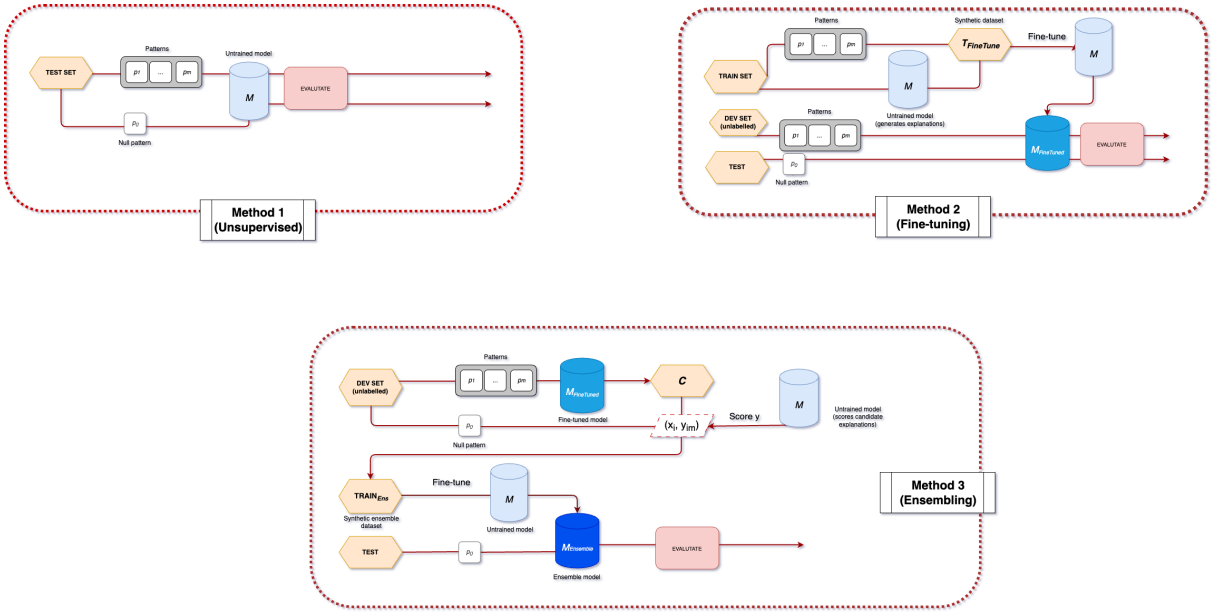

Figure 1: A diagram of our three proposed methods. Hexagons indicate datasets, cylinders indicate language models, white squares indicate prompting functions applied to inputs, and red rectangle indicates the final evaluation step.