NEGATER: Unsupervised Discovery of Negatives in Commonsense Knowledge Bases

Tara Safavi, Jing Zhu, Danai Koutra
University of Michigan, Ann Arbor
{tsafavi,jingzhuu,dkoutra}@umich.edu

Abstract

Codifying commonsense knowledge in machines is a longstanding goal of artificial intelligence. Recently, much progress toward this goal has been made with automatic knowledge base (KB) construction techniques. However, such techniques focus primarily on the acquisition of positive (true) KB statements, even though *negative* (false) statements are often also important for discriminative reasoning over commonsense KBs. As a first step toward the latter, this paper proposes NegatER, a framework that ranks potential negatives in commonsense KBs using a contextual language model (LM). Importantly, as most KBs do not contain negatives, NegatER relies only on the *positive* knowledge in the LM and does not require ground-truth negative examples. Experiments demonstrate that, compared to multiple contrastive data augmentation approaches, NegatER yields negatives that are more grammatical, coherent, and informative—leading to statistically significant accuracy improvements in a challenging KB completion task and confirming that the positive knowledge in LMs can be "repurposed" to generate negative knowledge.

1 Introduction

Endowing machines with *commonsense*, which is knowledge that members of a culture usually agree upon but do not express explicitly, is a major but elusive goal of artificial intelligence (Minsky, 1974; Davis et al., 1993; Liu and Singh, 2004; Davis and Marcus, 2015). One way to capture such knowledge is with curated commonsense knowledge bases (KBs), which contain semi-structured statements of "everyday" human knowledge. As such KBs are increasingly being used to augment the capabilities of intelligent agents (Hwang et al., 2021), automatically expanding their scope has become crucial (Li et al., 2016; Davison et al., 2019; Bosselut et al., 2019; Malaviya et al., 2020).

Table 1: Out-of-KB statements are less meaningful as negative examples when sampled at random versus ranked with our NEGATER framework. The random examples are taken from the test split of the ConceptNet benchmark introduced by Li et al. (2016).

Method	Negative statement
Random sampling	("tickle", HASSUBEVENT, "supermarket") ("lawn mower", ATLOCATION, "pantry") ("closet", USEDFOR, "play baseball")
NEGATER ranking	("ride horse", HASSUBEVENT, "pedal") ("zoo keeper", ATLOCATION, "jungle") ("air ticket", USEDFOR, "get onto trolley")

Previous research in this direction focuses primarily on the acquisition of positive knowledge, or that which is true about the world. However, understanding what is true about the world often also requires gathering and reasoning over explicitly *untrue* information. Humans routinely rely on **negative knowledge**—that is, what "not to do" or what "not to believe"—in order to increase certainty in decision-making and avoid mistakes and accidents (Minsky, 1994). Similarly, discriminative models that operate over structured knowledge from KBs often require explicit negative examples in order to learn good decision boundaries (Sun et al., 2019; Ahrabian et al., 2020; Ma et al., 2021).

The main challenge with machine acquisition of structured negative knowledge, commonsense or otherwise, is that most KBs do not contain negatives at all (Safavi and Koutra, 2020; Arnaout et al., 2020). Therefore, for KB-related tasks that require both positive and negative statements, negatives must either be gathered via human annotation, or else generated ad-hoc. Both of these approaches entail distinct challenges. On one hand, human annotation of negatives can be cost-prohibitive at scale. On the other, automatic negative generation without good training examples can lead to uninformative, even nonsensical statements (Table 1), because the prevailing approach is to randomly

Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing, pages 5633–5646
November 7–11, 2021.

sample negatives from the large space of all out-of-KB statements (Li et al., 2016).

To strike a balance between expert annotation, which is costly but accurate, and random sampling, which is efficient but inaccurate, we propose **NEGATER**, a framework for unsupervised discovery of **Negat**ive Commonsense Knowledge in **E**ntity and **R**elation form. Rather than randomly sampling from the space of all out-of-KB statements to obtain negatives, NEGATER *ranks* a selection of these statements such that higher-ranking statements are "more likely" to be negative. Ranking is done with a fine-tuned contextual language model (LM), building upon studies showing that LMs can be trained to acquire a degree of commonsense "knowledge" (Petroni et al., 2019).

Importantly, because we do not assume the presence of gold negative examples for training the LM, we devise techniques that make use of *positive KB statements only*. This distinguishes NEGATER from supervised generative commonsense KB construction techniques that require abundant gold examples, usually obtained via human annotation, for fine-tuning (Bosselut et al., 2019; Hwang et al., 2021; Jiang et al., 2021). Our realistic assumption means that we do not have any explicit examples of true negatives and therefore cannot guarantee a minimum true negative rate; indeed, obtaining true negatives in KBs is a hard problem in general (Arnaout et al., 2020). However, we show in detailed experiments that NEGATER strikes a delicate balance between several factors that contribute to high-quality negative knowledge, including task-specific utility, coherence, and the true negative rate. Our contributions are as follows:

- **Problem definition**: We provide the first rigorous definition of negative knowledge in commonsense KBs (§ 2), which as far as we are aware has not been studied before.
- **Framework**: We introduce NEGATER (§ 3), which ranks out-of-KB potential negatives using a contextual language model (LM). As KBs typically do not contain gold negatives, we devise an approach that relies only on the LM's *positive* beliefs. Specifically, NEGATER first fine-tunes the LM to acquire high-quality positive knowledge, then ranks potential negatives by how much they "contradict" the LM's positive knowledge, as measured by its classification scores or gradients.
- **Evaluation**: In keeping with the novelty of the problem, we conduct multiple evaluations that address the fundamental research questions of negative commonsense. First, we measure the effectiveness of our LM fine-tuning approach and the utility of NEGATER-generated negatives in KB completion tasks (§ 4, 5). Next, we study the intrinsic quality of the generated negatives (§ 6). When considering all such factors, NEGATER outperforms numerous competitive baselines. Most notably, training KB completion models with highly-ranked negative examples from NEGATER results in statistically significant accuracy improvements of up to 1.90%. Code and data are available at `https://github.com/tsafavi/NegatER`.

2 Problem definition

As the problem of negative knowledge has not yet been addressed in the commonsense KB completion literature, we begin by defining meaningful negatives in commonsense KBs.

2.1 Positive knowledge

A commonsense knowledge base (KB) consists of triples $\{x^+\} = \{(X_h, r, X_t)^+\}$, where the superscript denotes that all in-KB triples are assumed to be positive or true. In each triple, the first and third terms are head and tail entities in the form of phrases $X_h = [w_1, \ldots, w_h]$ and $X_t = [w_1, \ldots, w_t]$ drawn from a potentially infinite vocabulary. The relation types r are symbolic and drawn from a finite dictionary R. Figure 1 provides examples of positive statements from the ConceptNet KB (Speer and Havasi, 2012), e.g., (X_h="horse", r=IsA, X_t="expensive pet").

2.2 Negative knowledge

We denote a negative triple as $x^- \notin \{x^+\}$. As the space of negatives is evidently much larger than the space of positives, we define negative knowledge to exclude *trivial negatives*, for example simple negations or nonsensical statements. Specifically, drawing from the literature on procedural negative expertise in humans (Minsky, 1994; Gartmeier et al., 2008), we define negative knowledge as nonviable or explicitly false knowledge that is heuristically valuable with respect to a given task, goal, or decision. In the context of KBs, we devise three requirements that, combined, satisfy this definition:

R1 Negative knowledge must resemble positive

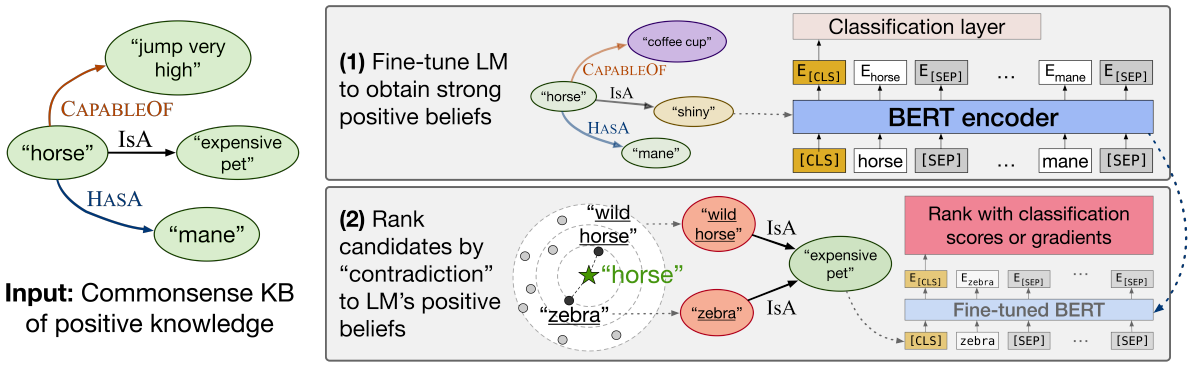

Figure 1: NEGATER consists of two steps: **(1)** Fine-tuning an LM on the input KB to obtain strong positive beliefs; and **(2)** Feeding a set of out-of-KB candidate statements to the fine-tuned LM and ranking them by the LM's classification scores or gradients. Here, the KB is a fragment of ConceptNet (Speer and Havasi, 2012).

knowledge in structure. This means that negative statements should **obey the grammatical rules** (parts of speech) of their relation types.

R2 The head and tail phrases must be **thematically or topically consistent**. For example, given the head phrase X_h="make coffee," a consistent tail phrase is one that is thematically related but still nonviable with respect to the whole statement, for example ("make coffee", HASSUBEVENT, "buy tea").

R3 Negative knowledge must be informative for a given task, goal, or decision. We consider a statement as informative if, when taken as true, it is **counterproductive or contradictory** to the goal at hand, e.g., ("make coffee", HASSUBEVENT, "drop mug").

3 Framework

We propose the NEGATER framework to match our definition of negative knowledge. As shown in Figure 1, NEGATER consists of two steps: First, a pretrained LM is fine-tuned on a given commonsense KB using a contrastive approach to acquire strong positive beliefs. Then, a set of grammatical (**R1**) and topically consistent (**R2**) out-of-KB candidate statements are fed to the LM and ranked by the degree to which they "contradict" the LM's fine-tuned positive beliefs (**R3**), such that the higher-ranking statements are more likely to be negative. We emphasize that ground-truth negative examples are *not required* at any point, which means that we trade off some accuracy (i.e., the true negative rate) for cost efficiency (i.e., the cost of gathering ground-truth negative examples for training via expert annotation).

We describe each step in detail in the remainder of this section.

3.1 Fine-tuning for positive knowledge

The first step of NEGATER is to minimally fine-tune a language model on a given commonsense KB using contrastive learning (step 1, Figure 1), such that it acquires strong positive beliefs. We focus on encoder-only BERT-based models (Devlin et al., 2019; Liu et al., 2019), as we will ultimately use their fine-tuned encodings to represent triples.

LM input and output We input a KB triple (X_h, r, X_t) to the LM by concatenating BERT's special `[CLS]` token with a linearized version of the triple, delineating the head tokens X_h, the relation r, and the tail tokens X_t with BERT's special `[SEP]` token. At the output of the encoder, we apply a semantic-level pooling operation (e.g., any of those proposed by Reimers and Gurevych (2019)) to obtain a single contextual representation of the triple, and pass it through a classification layer $W \in \mathbb{R}^H$, where H is the hidden layer dimension.

Supervision strategy Since the goal of fine-tuning is to endow the LM with strong positive beliefs, we use a common contrastive data augmentation technique for positive KB triple classification (Li et al., 2016; Malaviya et al., 2020). Specifically, for each positive x^+, we construct a contrastive corrupted version where the head, relation, or tail has been replaced by a random phrase or relation from the KB. We minimize binary cross-entropy loss between the positive training examples and their corrupted counterparts. We learn a decision threshold θ_r per relation r on the validation set to maximize validation accuracy, such that triples of relation r scored above θ_r are classified as positive.

3.2 Ranking out-of-KB statements

Now that we have an LM fine-tuned to a given commonsense KB, we feed a set of out-of-KB candidate statements to the LM in the same format as was used during fine-tuning, and rank them by the degree to which they "contradict" the LM's positive beliefs (step 2, Figure 1).

Out-of-KB candidate generation To gather out-of-KB candidate statements, we use a dense k-nearest-neighbors retrieval approach. The idea here is that the set of *all* out-of-KB statements is extremely large and most such statements are not likely to be meaningful, so we narrow the candidates down to a smaller set that is more likely to be grammatical (**R1**) and consistent (**R2**).

For each positive triple $x^+ = (X_h, r, X_t)^+$, we retrieve the k nearest-neighbor phrases to head phrase X_h using a maximum inner product search (Johnson et al., 2019) over pre-computed embeddings of the KB's entity phrases. While any choice of embedding and distance measure may be used, we use Euclidean distance between the `[CLS]` embeddings output by a separate pretrained BERT model for its empirical good performance. We then replace X_h in the head slot of the original positive x^+ by each of its neighbors $\tilde{X}_h$ in turn, yielding a set of candidates

$$\{\tilde{x}\}_{i=1}^{k}, \;\; \tilde{x} = (\tilde{X}_h, r, X_t).$$

We discard any candidates that already appear in the KB and repeat this process for the tail phrase X_t, yielding up to $2k$ candidates $\tilde{x}$ per positive x^+. We also filter the candidates to only those for which the retrieved head (tail) phrase $\tilde{X}_h$ ($\tilde{X}_t$) appears in the head (tail) slot of relation r in the KB.

Meeting R1, R2, and R3 Our filtering process discards candidate triples whose head/tail entities have not been observed to co-occur with relation r, which preserves the grammar (**R1**) of the relation. Notice that by retrieving the nearest neighbors of each head and tail phrase by semantic similarity, we also preserve the topical consistency (**R2**) of the original positive statements.

Finally, to meet requirement **R3**, we rank the remaining out-of-KB candidates by the degree to which they "contradict" the positive beliefs of the fine-tuned LM. These ranked statements can be then taken in order of rank descending as input to any discriminative KB reasoning task requiring negative examples, with the exact number of negatives being determined by the practitioner and application. We propose two independent ranking strategies:

3.2.1 NEGATER-θ_r: Ranking with scores

Our first approach, NEGATER-θ_r, relies on the decision thresholds θ_r set during the validation stage of fine-tuning. We feed the candidates $\tilde{x}$ to the LM and take only those that the LM classifies below the respective decision threshold θ_r. Per relation r, the candidates are ranked descending by their scores at the output of the classification layer, such that the higher-ranking candidates look more plausible—that is, "almost positive"—to the LM.

3.2.2 NEGATER-∇: Ranking with gradients

The premise of our second approach, NEGATER-∇, is that the candidates that most "surprise" the LM *when labeled as true* are the most likely to be negative, because they most directly contradict what the LM has observed during fine-tuning.

We quantify "surprisal" with the LM's gradients. Let $\mathcal{L}(\tilde{x}; \tilde{y})$ be the binary cross-entropy loss evaluated on candidate $\tilde{x}$ given a corresponding label $\tilde{y} \in \{-1, 1\}$. We feed each $\tilde{x}$ to the LM and compute the magnitude of the gradient of $\mathcal{L}$ with respect to the LM's parameters Θ, given a positive labeling of $\tilde{x}$:

$$\tilde{M} = \left\| \frac{\partial \mathcal{L}(\tilde{x}; \tilde{y} = 1)}{\partial \Theta} \right\|, \qquad (1)$$

and rank candidates in descending order of gradient magnitude $\tilde{M}$. Here, $\tilde{M}$ signifies the amount to which the LM's fine-tuned beliefs would need to be updated to incorporate this candidate as positive. Therefore, the higher the $\tilde{M}$, the more directly $\tilde{x}$ contradicts or negates the LM's positive beliefs.

Faster computation Because NEGATER-∇ requires a full forward and backward pass for each candidate $\tilde{x}$, it can be costly for a large number N of candidates. We therefore propose a simple (optional) trick to speed up computation. We first compute $\tilde{M}$ for an initial sample of $n \ll N$ candidates. We then use the contextual representations of these n candidates and their gradient magnitudes $\tilde{M}$ as training features and targets, respectively, to learn a regression function $f_{\tilde{M}} : \mathbb{R}^H \rightarrow \mathbb{R}$. Finally, we substitute the LM's fine-tuning layer with $f_{\tilde{M}}$, allowing us to skip the backward pass and feed batches of candidates $\tilde{x}$ to the LM in forward passes. In our experiments, we will show that this

approach is an effective and efficient alternative to full-gradient computation.

Gradients versus losses On the surface, it might seem that NEGATER-∇ could be made more efficient by ranking examples descending by their *losses*, instead of gradients. However, notice that the binary cross-entropy loss $\mathcal{L}(\tilde{x}; \tilde{y} = 1)$ is low for candidates $\tilde{x}$ that receives high scores from the LM, and high for candidates that receive low scores. Due to the contrastive approach that we used for fine-tuning, candidates with the lowest losses are mainly *true* statements, and candidates with the highest losses are mainly *nonsensical* statements. Therefore, the losses do not directly correlate with how "contradictory" the candidate statements are. By contrast, the gradient-based approach quantifies how much the LM would need to change its beliefs to incorporate the new knowledge as positive, which more directly matches requirement **R3**.

4 Fine-tuning evaluation

In this section, we evaluate the efficacy of the fine-tuning step of NEGATER (§ 3.1). In the following sections, we will evaluate the efficacy of the ranking step of NEGATER (§ 3.2) from quantitative and qualitative perspectives.

4.1 Data

The goal of this experiment is to evaluate whether our fine-tuning strategy from § 3.1 endows LMs with sufficiently accurate positive knowledge. For this, we use the English-language triple classification benchmark introduced by Li et al. (2016), which consists of 100K/2400/2400 train/validation/test triples across 34 relations and 78,334 unique entity phrases from ConceptNet 5 (Speer and Havasi, 2012). The evaluation metric is accuracy. In the evaluation splits, which are balanced positive/negative 50/50, the negatives were constructed by swapping the head, relation, or tail of each positive x^+ with that of another randomly sampled positive from the KB. Note that while the test negatives were generated randomly and are therefore mostly nonsensical (Table 1), we use this benchmark because it mainly tests models' recognition of positive knowledge, which matches the goals of our fine-tuning procedure. Ultimately, however, a more difficult dataset will be needed, which we will introduce in the next section.

Table 2: Our fine-tuned BERT reaches state-of-the-art accuracy on the ConceptNet benchmark from (Li et al., 2016). Baseline results are reported directly from the referenced papers.

	Acc.
Bilinear AVG (Li et al., 2016)	91.70
DNN AVG (Li et al., 2016)	92.00
DNN LSTM (Li et al., 2016)	89.20
DNN AVG + CKBG (Saito et al., 2018)	94.70
Factorized (Jastrzębski et al., 2018)	79.40
Prototypical (Jastrzębski et al., 2018)	89.00
Concatenation (Davison et al., 2019)	68.80
Template (Davison et al., 2019)	72.20
Template + Grammar (Davison et al., 2019)	74.40
Coherency Ranking (Davison et al., 2019)	78.80
KG-BERT$_{\text{BERT-BASE}}$ (Shen et al., 2020)	93.20
KG-BERT$_{\text{GLM(RoBERTa-LARGE)}}$ (Shen et al., 2020)	94.60
Fine-tuned BERT (ours)	**95.42**
Fine-tuned RoBERTa (ours)	94.37
Human estimate (Li et al., 2016)	95.00

4.2 Methods

We fine-tune BERT-BASE-UNCASED (Devlin et al., 2019) and RoBERTa-BASE (Liu et al., 2019). We compare our LMs to all published results on the same evaluation splits of which we are aware. Our baselines include both KB embeddings (Li et al., 2016; Jastrzębski et al., 2018) and contextual LMs (Davison et al., 2019; Shen et al., 2020). Appendix A provides implementation details.

4.3 Results

The results in Table 2 confirm the effectiveness of our fine-tuning approach, as our BERT reaches state-of-the-art accuracy on ConceptNet. It even outperforms KG-BERT$_{\text{GLM(RoBERTa-LARGE)}}$ (Shen et al., 2020), which requires an entity linking step during preprocessing and uses a RoBERTa-LARGE model pretrained with several extra tasks. In fact, we suspect that our fine-tuned BERT has saturated this benchmark, as it slightly exceeds the human accuracy estimate provided by Li et al. (2016).

5 Task-based evaluation

We next evaluate the efficacy of the ranking step in NEGATER. Specifically, we next show how the top-ranking negative examples from NEGATER can be informative (**R3**) for training KB completion models. Similar to the previous section, we fine-tune pretrained BERT and RoBERTa models for a commonsense triple classification task. However, here we use a more challenging dataset split, and

vary the ways that negatives are sampled at training time.

5.1 Data

As discussed previously, the ConceptNet split introduced by Li et al. (2016) is already saturated by BERT, likely because it contains "easy" negative test examples. We therefore construct a new, more challenging split by taking the small percentage (3%) of triples in the benchmark with negated relations (e.g., NOTISA, six total), each of which has a positive counterpart in the KB (e.g., ISA). We filter the dataset to the positive/negated relation pairs only, and take the negated triples as *true negative instances* for testing by removing the NOT- relation prefixes. Our new split, which we call ConceptNet-TN to denote True Negatives, consists of 36,210/3,278/3,278 train/validation/test triples. Again, the classes are balanced positive/negative, so accuracy is our main performance metric.

Note that because this dataset contains true (hard) negatives, we expect accuracy to be much lower than what we achieved in Table 2.

5.2 Baselines

As baselines we consider several contrastive data augmentation approaches, all of which involve corrupting positive in-KB samples.

We employ the following negative sampling baselines designed for **commonsense KBs**:

- **UNIFORM** (Li et al., 2016; Saito et al., 2018): We replace the head phrase X_h or tail phrase X_t of each positive $(X_h, r, X_t)^+$ by uniformly sampling another phrase from the KB.
- **COMET** (Bosselut et al., 2019): COMET is a version of GPT (Radford et al., 2018) that was fine-tuned to generate the tail phrase of a commonsense triple, conditioned on a head phrase and relation. To make COMET generate negatives, we prepend a "not" token to each positive head phrase X_h^+ and generate 10 tail phrases X_t^{COMET} for the modified head/relation prefix using beam search. Finally, we replace the tail phrase X_t in the positive with each X_t^{COMET} in turn, yielding negatives $(X_h^+, r, X_t^{\text{COMET}})$.

To investigate whether negative samplers tailored to encyclopedic knowledge can transfer to commonsense, we employ the following state-of-the-art baselines designed for **encyclopedic KBs**:

- **ROTATE-SA** (Sun et al., 2019): For each positive instance, a pool of candidate negatives is generated with UNIFORM. The candidates are then scored by the (shallow, but state-of-the-art) RotatE KB embedding, and a negative is sampled from the candidate pool with probability proportional to the score distribution. We take the top 50% of self-adversarially generated statements as negative examples, in order of score descending, from the last epoch of training.
- **SANS** (Ahrabian et al., 2020) is a graph-structural negative sampler that corrupts head/tail phrases of positive instances by sampling from the k-hop neighborhood of each KB entity. We set $k = 2$.

Finally, we **devise two intuitive baselines**:

- **SLOTS**: We replace the head phrase X_h (tail phrase X_t) of each positive $(X_h, r, X_t)^+$ by uniformly sampling from the set of phrases that appear in the head (tail) slot of KB triples mentioning relation r. We filter out all negative samples that appear in the KB.
- **ANTONYMS**: We tag each phrase in the KB as either a verb, noun, or adjective phrase using the SpaCy POS tagger.[1] Then, for each verb (noun, adjective) phrase, we replace the first verb (noun, adjective) token with a randomly selected antonym from either WordNet (Miller, 1998) or the gold lexical contrast dataset from (Nguyen et al., 2016).

5.3 NEGATER variants

We generate out-of-KB candidates for NEGATER with our k-NN approach using k=10, yielding around 570K candidates. We implement the NEGATER candidate ranking methods as follows:

- **NEGATER-θ_r**: We rank candidates using fine-tuned BERT's classification scores. Since the scores are scaled differently by relation type, we combine the top-ranking 50% of candidates per relation and shuffle them.
- **NEGATER-∇**: We again use BERT to rank the candidates. To choose between the full-gradient and gradient-prediction approaches (§ 3.2.2), we train an MLP to predict gradient magnitudes and plot the mean absolute

[1] https://spacy.io/usage/linguistic-features#pos-tagging

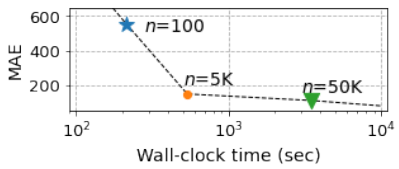

Figure 2: Lower left corner is best: Wall-clock time versus training loss (MAE) for NEGATER-∇ gradient magnitude prediction as training set size n increases.

error training loss after 100 epochs for different training set sizes n. Figure 2 shows that even for n=5K examples, the loss quickly approaches zero. Therefore, for efficiency, we use an MLP trained on n=20K examples, which takes around 1 hour to train and rank candidates on a single GPU, compared to an estimated 14 hours for the full-gradient approach. For a random sample of 100 candidates, the Pearson correlation coefficient between the true/predicted gradient magnitudes is ρ=0.982, indicating that the approximation is highly accurate.

- **No-ranking ablation**: Finally, in order to measure the importance of the LM ranking component of NEGATER, we introduce an ablation which randomly shuffles the out-of-KB candidates rather than ranking them.

After we obtain each ranked list of candidates, we feed the statements as negative training examples to BERT/RoBERTa in order of rank descending.

5.4 Results

For all performance metrics, we report averages over five trials to account for randomness in sampling and parameter initializations.

Accuracy comparison As shown in Table 3, training with the top-ranking negative examples from NEGATER always yields the best accuracy for both LMs, up to 1.90% more than the baselines. Note that this improvement is achieved with changing how only *half* of the training examples (the negatives) are sampled. Notice also that our NEGATER variants are the only samplers to offer **statistically significant improvements** over the UNIFORM baseline at $\alpha < 0.01$ for BERT and $\alpha < 0.05$ for RoBERTa (two-sided t-tests, five trials per model), signifying better-than-chance improvements.

Notice also that our most competitive baseline is SLOTS, which is a contrastive approach that samples new head/tail phrases from those appearing in the corresponding slots of the current relation r—that is, preserving the grammar (**R1**) of the relation. This confirms that grammatical negative samples are indeed more informative than nonsensical ones.

Table 3: Accuracy on ConceptNet-TN using different negative sampling approaches: Our NEGATER variants are the only negative samplers to offer statistically significant improvements over the popular UNIFORM baseline at $\alpha < 0.01$ ($^{\blacktriangle}$) for BERT and $\alpha < 0.05$ ($^{\triangle}$) for RoBERTa (two-sided t-test, five trials per model). **Bold/underline**: Best result per LM; Underline only: Second-best result per LM.

		BERT	RoBERTa
Baselines	UNIFORM	75.60 ± 0.24	75.55 ± 0.43
	COMET	76.04 ± 0.63	75.86 ± 0.75
	ROTATE-SA	75.30 ± 0.51	75.20 ± 0.37
	SANS	75.45 ± 0.38	75.17 ± 0.37
	SLOTS	$76.46 \pm 0.58^{\triangle}$	75.80 ± 0.25
	ANTONYMS	$76.06 \pm 0.30^{\triangle}$	75.58 ± 0.58
NEGATER	θ_r ranking	$\mathbf{76.95 \pm 0.28}^{\blacktriangle}$	76.29 ± 0.59
	∇ ranking	$76.53 \pm 0.22^{\blacktriangle}$	$\mathbf{76.34 \pm 0.32}^{\triangle}$
	No ranking	75.61 ± 0.29	75.29 ± 0.19

Encyclopedic versus commonsense? We hypothesize that the encyclopedic KB baselines ROTATE-SA and SANS underperform because such methods assume that the KB is a dense graph. While this is usually true for encyclopedic KBs, many commonsense KBs are highly sparse because entities are not disambiguated, which means that multiple phrases referring to the same concept may be treated as different entities in the KB.[2] For example, SANS assumes that there are plentiful entities within the k-hop neighborhood of a query entity, whereas in reality there may be very few, and these entities may not be grammatical in context of the original positive (**R1**) nor thematically relevant (**R2**) to the query entity. Therefore, encyclopedic negative samplers may not be transferrable to commonsense KBs or other highly sparse KBs.

Ablation study Table 3 also indicates that the LM ranking component of NEGATER is crucial for improving accuracy. Our no-ranking ablation leads to lower classification accuracy than both NEGATER-θ_r and NEGATER-∇. Empirically, we find that this is because the ranking step helps filter

[2] Malaviya et al. (2020) provide an illustrative example for comparison: The popular encyclopedic KB completion benchmark FB15K-237 (Toutanova and Chen, 2015) is 75× denser than the ConceptNet benchmark studied in this paper.

Table 4: NEGATER consistently yields the highest precision on ConceptNet-TN among negative samplers because it lowers the false positive rate: Performance drill-down (stdevs omitted for space). $^{\blacktriangle}$, $^{\triangle}$: Significant improvement over UNIFORM at $\alpha < 0.01$ and $\alpha < 0.05$, respectively.

		BERT		RoBERTa	
		Prec.	Rec.	Prec.	Rec.
Baselines	UNIFORM	71.29	**85.83**	73.36	80.28
	COMET	73.73	80.99	73.47	81.02
	ROTATE-SA	74.83	76.59	73.70	78.48
	SANS	72.54	82.11	73.26	79.50
	SLOTS	75.21$^{\triangle}$	79.34	73.85	80.17
	ANTONYMS	72.55	83.98	72.98	**81.62**
NEGATER	θ_r ranking	75.12$^{\blacktriangle}$	80.68	**75.92**$^{\blacktriangle}$	77.05
	∇ ranking	76.60$^{\blacktriangle}$	76.50	75.75$^{\blacktriangle}$	77.57
	No ranking	**76.81**$^{\blacktriangle}$	73.42	75.67$^{\triangle}$	74.78

out false negatives generated by our k-NN candidate construction procedure.

Performance drill-down Finally, Table 4 provides precision and recall scores to further "drill down" into NEGATER's effects. Evidently, the NEGATER variants consistently yield the best precision, whereas there is no consistent winner in terms of recall. To understand why NEGATER improves precision, we remind the reader that precision is calculated as $P = (TP)/(TP + FP)$, where TP stands for true positives and FP stands for false positives. Because training with NEGATER examples helps the LMs better recognize hard negatives—examples that "look positive" but are really negative—the LM mislabels fewer negatives, decreasing the false positive rate.

6 Human evaluation

Finally, we collect qualitative human judgments on the examples output by each negative sampler.

6.1 Data

To cover a diverse set of reasoning scenarios, we consider the HASPREREQUISITE, HASPROPERTY, HASSUBEVENT, RECEIVESACTION, and USEDFOR relations from ConceptNet. For each relation and negative sampler, we take 30 negative statements at random, yielding 1,350 statements judged in total (5 relations $\times$ 9 negative samplers $\times$ 30 statements per method/relation).

6.2 Methods

We gather judgments for (**R1**) grammar on a binary scale (incorrect/correct) and (**R2**) thematic consistency of the head/tail phrases on a 4-point scale ("not consistent at all", "a little consistent", "somewhat consistent", "highly consistent"). To estimate the true negative rate, we also obtain truthfulness judgments on a 4-point scale ("not truthful at all", "sometimes true", "mostly true", "always true"). We recruit four annotators who are fluent in English. Among 50 statements shared across the annotators, we observe an average variance of 0.058 points on the 0/1 scale for **R1**, 0.418 points on the 4-point scale for **R2**, and 0.364 points on the 4-point truthfulness scale. According to previous work in commonsense KB construction (Romero et al., 2019), these values indicate high agreement. Appendix B provides the annotation instructions.

Table 5: NEGATER best trades off grammar (**R1**), consistency (**R2**), and the true negative rate, as measured by the percentage of statements labeled "never true": Human annotation scores, normalized out of 1. Relative and average ranks are provided because not all raw metrics are directly comparable—e.g., grammar (**R1**) is judged as binary, whereas consistency (**R2**) is graded.

		R1	**R2**	% "never true"	Avg rank
Baselines	UNIFORM	0.487 (9)	0.408 (7)	0.747 (3)	6.33 (9)
	COMET	0.580 (8)	**0.703** (1)	0.407 (9)	6.00 (8)
	ROTATE-SA	0.733 (7)	0.373 (8)	0.767 (2)	5.67 (7)
	SANS	0.760 (6)	0.532 (5)	0.633 (4)	5.00 (4)
	SLOTS	0.853 (5)	0.372 (9)	**0.773** (1)	5.00 (4)
	ANTONYMS	0.860 (4)	0.495 (6)	0.613 (5)	5.00 (4)
NEGATER	θ_r ranking	0.880 (3)	0.635 (2)	0.413 (8)	4.33 (3)
	∇ ranking	**0.927** (1)	0.555 (4)	0.587 (6)	**3.67** (1)
	No ranking	0.920 (2)	0.592 (3)	0.560 (7)	4.00 (2)

6.3 Results

Table 5 compares normalized average judgment scores for **R1** and **R2**, as well as the percentage of statements labeled as "never true" (i.e., the true negative rate). Here, the takeaway is that the requirements of negative knowledge are a tradeoff, and **NEGATER-∇ best manages this tradeoff**. Indeed, the methods that yield the most true negatives (ROTATE-SA, SLOTS) perform the worst in grammar (**R1**) and consistency (**R2**), whereas methods that yield more consistent statements like COMET have a comparatively low true negative rate.

Finally, Table 6 provides examples of statements with consistency (**R2**) and truthfulness judgments. Again, it is evident that NEGATER-∇ best manages the tradeoffs of negative knowledge. In fact, it is the only negative sampler for which a majority of examples are rated both as "never true" (58.67%) and "somewhat consistent" or higher (62%).

Table 6: Our NEGATER-∇ variant best handles the tradeoff between consistency (**R2**) and truthfulness: Representative negative examples from the most competitive methods SLOTS, NEGATER-θ_r, and NEGATER-∇.

Method	Negative statement	Consistent?	True?
SLOTS	("open business", HASPREREQUISITE, "hide behind door")	A little	Never
	("go somewhere", HASSUBEVENT, "bruise appears")	Not at all	Never
	("mailbox", USEDFOR, "sleeping guests")	Not at all	Never
NEGATER-θ_r	("play baseball", HASPREREQUISITE, "join hockey team")	Somewhat	Never
	("comfort someone", HASSUBEVENT, "talk with them")	Highly	Mostly
	("having a bath", USEDFOR, "refreshing yourself")	Highly	Sometimes
NEGATER-∇	("hear news", HASPREREQUISITE, "record something")	A little	Never
	("drink water", HASSUBEVENT, "inebriation")	Highly	Never
	("luggage trolley", USEDFOR, "moving rocks")	Highly	Never

7 Related work

Commonsense KB completion Approaches to commonsense KB completion include triple classification (Li et al., 2016; Saito et al., 2018; Jastrzębski et al., 2018; Davison et al., 2019), generation of novel triples (Bosselut et al., 2019; Hwang et al., 2021), and link prediction (Malaviya et al., 2020). Such approaches either focus solely on modeling positive knowledge, or else generate negatives at random, making our work the first attempt in the direction of meaningful negative knowledge.

Knowledge in language models Several studies have shown that deep contextual language models acquire a degree of implicit commonsense knowledge during pretraining (Petroni et al., 2019; Davison et al., 2019; Roberts et al., 2020), which can be further sharpened specifically toward KB completion by targeted fine-tuning (Bosselut et al., 2019; Hwang et al., 2021). Our results in fine-tuning BERT to ConceptNet (§ 4) validate these findings. Other studies have demonstrated that pretrained LMs struggle to distinguish affirmative sentences from their negated counterparts (Kassner and Schütze, 2020; Ettinger, 2020), although this can again be addressed by fine-tuning (Kassner and Schütze, 2020; Jiang et al., 2021). Note, however, that we deal not with linguistic *negation* but with negative, or false, knowledge.

Negatives in knowledge bases We are not aware of any tailored negative samplers for commonsense knowledge. However, several negative samplers for encyclopedic KBs like Freebase exist, including self-adversarial (Cai and Wang, 2018; Sun et al., 2019), graph-structural (Ahrabian et al., 2020), and heuristic "interestingness" (Arnaout et al., 2020) approaches. While these methods share our high-level goal, we showed in § 5 that they are less effective on highly sparse commonsense KBs.

8 Conclusion

In this paper we rigorously defined negative knowledge in commonsense KBs and proposed a framework, NEGATER, to address this problem. Importantly, our framework does not require ground-truth negatives at any point, making it an effective choice when gold training examples are not available. We empirically demonstrated the strength of NEGATER over many competitive baselines in multiple evaluations, including the strength of our fine-tuning approach, the task-based utility of NEGATER statements, and the intrinsic quality of these statements. A promising future direction is to explore new reasoning tasks that can be improved with negative knowledge from NEGATER, for example multiple-choice commonsense QA (Ma et al., 2021), which often requires high-quality negative examples for training.

Acknowledgements

We thank the reviewers for their constructive feedback. This material is based upon work supported by the National Science Foundation under a Graduate Research Fellowship and CAREER Grant No. IIS 1845491, Army Young Investigator Award No. W911NF1810397, the Advanced Machine Learning Collaborative Grant from Procter & Gamble, and Amazon, Google, and Facebook faculty awards. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation or other funding parties.

References

Kian Ahrabian, Aarash Feizi, Yasmin Salehi, William L. Hamilton, and Avishek Joey Bose. 2020. Structure aware negative sampling in knowledge graphs. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 6093–6101, Online. Association for Computational Linguistics.

Hiba Arnaout, Simon Razniewski, and Gerhard Weikum. 2020. Enriching knowledge bases with interesting negative statements. In *Automated Knowledge Base Construction*.

Antoine Bosselut, Hannah Rashkin, Maarten Sap, Chaitanya Malaviya, Asli Celikyilmaz, and Yejin Choi. 2019. COMET: Commonsense transformers for automatic knowledge graph construction. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4762–4779, Florence, Italy. Association for Computational Linguistics.

Liwei Cai and William Yang Wang. 2018. KBGAN: Adversarial learning for knowledge graph embeddings. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 1470–1480, New Orleans, Louisiana. Association for Computational Linguistics.

Ernest Davis and Gary Marcus. 2015. Commonsense reasoning and commonsense knowledge in artificial intelligence. *Communications of the ACM*, 58(9):92–103.

Randall Davis, Howard Shrobe, and Peter Szolovits. 1993. What is a knowledge representation? *AI magazine*, 14(1):17–17.

Joe Davison, Joshua Feldman, and Alexander Rush. 2019. Commonsense knowledge mining from pretrained models. 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 1173–1178, Hong Kong, China. Association for Computational Linguistics.

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*, 8:34–48.

Martin Gartmeier, Johannes Bauer, Hans Gruber, and Helmut Heid. 2008. Negative knowledge: Understanding professional learning and expertise. *Vocations and Learning*, 1(2):87–103.

Suchin Gururangan, Ana Marasović, Swabha Swayamdipta, Kyle Lo, Iz Beltagy, Doug Downey, and Noah A. Smith. 2020. Don't stop pretraining: Adapt language models to domains and tasks. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 8342–8360, Online. Association for Computational Linguistics.

Jena D Hwang, Chandra Bhagavatula, Ronan Le Bras, Jeff Da, Keisuke Sakaguchi, Antoine Bosselut, and Yejin Choi. 2021. Comet-atomic 2020: On symbolic and neural commonsense knowledge graphs. In *Proceedings of the AAAI Conference on Artificial Intelligence*.

Stanislaw Jastrzębski, Dzmitry Bahdanau, Seyedarian Hosseini, Michael Noukhovitch, Yoshua Bengio, and Jackie Cheung. 2018. Commonsense mining as knowledge base completion? a study on the impact of novelty. In *Proceedings of the Workshop on Generalization in the Age of Deep Learning*, pages 8–16, New Orleans, Louisiana. Association for Computational Linguistics.

Liwei Jiang, Antoine Bosselut, Chandra Bhagavatula, and Yejin Choi. 2021. "i'm not mad": Commonsense implications of negation and contradiction. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics*, Online. Association for Computational Linguistics.

Jeff Johnson, Matthijs Douze, and Hervé Jégou. 2019. Billion-scale similarity search with gpus. *IEEE Transactions on Big Data*.

Nora Kassner and Hinrich Schütze. 2020. Negated and misprimed probes for pretrained language models: Birds can talk, but cannot fly. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7811–7818, Online. Association for Computational Linguistics.

Xiang Li, Aynaz Taheri, Lifu Tu, and Kevin Gimpel. 2016. Commonsense knowledge base completion. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1445–1455, Berlin, Germany. Association for Computational Linguistics.

Hugo Liu and Push Singh. 2004. Conceptnet—a practical commonsense reasoning tool-kit. *BT technology journal*, 22(4):211–226.

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*.

Kaixin Ma, Filip Ilievski, Jonathan Francis, Yonatan Bisk, Eric Nyberg, and Alessandro Oltramari. 2021. Knowledge-driven data construction for zero-shot evaluation in commonsense question answering. In *Proceedings of the AAAI Conference on Artificial Intelligence*.

Chaitanya Malaviya, Chandra Bhagavatula, Antoine Bosselut, and Yejin Choi. 2020. Commonsense knowledge base completion with structural and semantic context. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 2925–2933.

George A Miller. 1998. *WordNet: An electronic lexical database*. MIT press.

Marvin Minsky. 1974. A framework for representing knowledge.

Marvin Minsky. 1994. Negative expertise. *International Journal of Expert Systems*, 7(1):13–19.

Kim Anh Nguyen, Sabine Schulte im Walde, and Ngoc Thang Vu. 2016. Integrating distributional lexical contrast into word embeddings for antonym-synonym distinction. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 454–459, Berlin, Germany. Association for Computational Linguistics.

Fabio Petroni, Tim Rocktäschel, Sebastian Riedel, Patrick Lewis, Anton Bakhtin, Yuxiang Wu, and Alexander Miller. 2019. Language models as knowledge bases? 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 2463–2473, Hong Kong, China. Association for Computational Linguistics.

Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. 2018. Improving language understanding by generative pre-training.

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, Hong Kong, China. Association for Computational Linguistics.

Adam Roberts, Colin Raffel, and Noam Shazeer. 2020. How much knowledge can you pack into the parameters of a language model? In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 5418–5426, Online. Association for Computational Linguistics.

Julien Romero, Simon Razniewski, Koninika Pal, Jeff Z. Pan, Archit Sakhadeo, and Gerhard Weikum. 2019. Commonsense properties from query logs and question answering forums. In *ACM Conference on Information and Knowledge Management*, pages 1411–1420.

Tara Safavi and Danai Koutra. 2020. CoDEx: A Comprehensive Knowledge Graph Completion Benchmark. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 8328–8350, Online. Association for Computational Linguistics.

Itsumi Saito, Kyosuke Nishida, Hisako Asano, and Junji Tomita. 2018. Commonsense knowledge base completion and generation. In *Proceedings of the 22nd Conference on Computational Natural Language Learning*, pages 141–150, Brussels, Belgium. Association for Computational Linguistics.

Tao Shen, Yi Mao, Pengcheng He, Guodong Long, Adam Trischler, and Weizhu Chen. 2020. Exploiting structured knowledge in text via graph-guided representation learning. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 8980–8994, Online. Association for Computational Linguistics.

Robyn Speer and Catherine Havasi. 2012. Representing general relational knowledge in ConceptNet 5. In *Proceedings of the Eighth International Conference on Language Resources and Evaluation (LREC'12)*, pages 3679–3686, Istanbul, Turkey. European Language Resources Association (ELRA).

Zhiqing Sun, Zhi-Hong Deng, Jian-Yun Nie, and Jian Tang. 2019. Rotate: Knowledge graph embedding by relational rotation in complex space. In *International Conference on Learning Representations*.

Kristina Toutanova and Danqi Chen. 2015. Observed versus latent features for knowledge base and text inference. In *Proceedings of the 3rd Workshop on Continuous Vector Space Models and their Compositionality*, pages 57–66, Beijing, China. Association for Computational Linguistics.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Remi 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 Rush. 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, Online. Association for Computational Linguistics.

A Implementation details

Pooling To obtain a single contextual representation of a triple from a sequence of triple tokens, we experiment with three standard pooling approaches (Reimers and Gurevych, 2019): Taking the reserved `[CLS]` token embedding from the output of the encoder, and mean- and max-pooling over all output token representations. As we do not observe statistically significant differences in performance among the pooling operations, we use the `[CLS]` token as the triple embedding, since this is a very common method for encoding text sequences with BERT (Gururangan et al., 2020).

LM hyperparameters Following the recommendations given by Devlin et al. (2019), we search manually among the following hyperparameters (best configuration for BERT in **bold**, RoBERTa underlined): Batch size in $\{\underline{\mathbf{16}}, 32\}$; Learning rate in $\{10^{-4}, 10^{-5}, \underline{\mathbf{2 \times 10^{-5}}}, 3 \times 10^{-5}\}$; Number of epochs in $\{3, 5, \underline{7}, 10, \mathbf{13}\}$; Number of warmup steps in $\{0, \underline{\mathbf{10K}}, 100\text{K}\}$; Maximum sequence length in $\{16, \underline{\mathbf{32}}, 64\}$. All other hyperparameters are as reported in (Devlin et al., 2019).

Training negatives For the fine-tuning evaluation, to make our training process as similar as possible to common practice in the literature (Li et al., 2016; Saito et al., 2018; Jastrzębski et al., 2018), we corrupt each positive x^+ by replacing the head, relation, and tail of the positive in turn with a randomly sampled phrase or relation. For the task-based evaluation, we sample one negative per positive in order to make running many experiments across different negative samplers feasible.

Software and hardware We implement our LMs with the Transformers PyTorch library (Wolf et al., 2020) and run all experiments on a NVIDIA Tesla V100 GPU with 16 GB of RAM. Both BERT and RoBERTa take around 1.5 hours/epoch to train on the ConceptNet benchmark.

B Annotation instructions

In this section we provide the annotation instructions for § 6.

B.1 Task definition

In this task you will judge a set of statements based on how grammatical, truthful, and consistent they are. Each statement is given in [head phrase, relation, tail phrase] form. The criteria are as follows:

- **Grammar**: Our definition of grammar refers to whether each statement follows the grammar rules we provide for its relation type. We do not include proper use of punctuation (e.g., commas, apostrophes) or articles (e.g., "the", "a", "this") in our definition of grammar. The choices are "correct", "partially correct or unsure", and "incorrect". (*Note: in our analyses we binarize these choices, considering "partially correct" and "incorrect" as the same.*)
- **Truthfulness**: Our definition of truthfulness refers to how often you believe the whole statement holds true. The choices are: "always true", "mostly true", "sometimes true", and "never true".
- **Consistency**: We define "consistency" as the degree to which the head and tail phrases are consistent in terms of the topic, theme, or goal that they refer to. For example, the phrases "football" and "baseball" are highly consistent because they both refer to team sports, whereas the phrases "football" and "cactus" are not consistent. The choices are: "highly consistent", "somewhat consistent", "a little consistent", and "not consistent at all".

You may fill your answers in any order. For example, you might find it helpful to judge the grammar of all statements first, then the truthfulness, then the consistency. Some of the statements are subjective and there is not always a "right" answer, especially for the consistency criterion. If you are unsure of a word or reference, you may use Google or other search engines. You may also explain your reasoning/interpretation in the optional Notes box.

B.2 Examples and explanations

HasPrerequisite The HASPREREQUISITE relation describes prerequisites or pre-conditions for actions or states of being. It requires a verb phrase (an action or a state of being) in the head slot and a verb phrase or noun phrase in the tail slots. Examples:

- ("pay bill", HASPREREQUISITE, "have money")
  - Grammar: correct
  - Truthfulness: always true
  - Consistency: highly consistent

- ("purchase a cellular phone", HASPREREQUISITE, "study")
  - Grammar: correct
  - Truthfulness: never true
  - Consistency: not consistent at all
- ("paint your house", HASPREREQUISITE, "purple")
  - Grammar: incorrect
  - Truthfulness: never true
  - Consistency: a little consistent (*Our interpretation: Painting your house involves choosing a color, so the statement could be construed as a little consistent, even though it's grammatically incorrect.*)
- ("eat", HASPREREQUISITE, "send them to their room")
  - Grammar: correct
  - Truthfulness: never true
  - Consistency: not consistent at all

HasProperty The HASPROPERTY relation describes properties of actions or objects. It requires a verb phrase or noun phrase in the head slot and a description in the tail slot. Examples:

- ("school bus", HASPROPERTY, "yellow")
  - Grammar: correct
  - Truthfulness: mostly true (*Our interpretation: Yellow school buses are very common in the USA and Canada, but not all school buses are yellow.*)
  - Consistency: highly consistent
- ("basketball", HASPROPERTY, "round")
  - Grammar: correct
  - Truthfulness: always true
  - Consistency: highly consistent
- ("pilot", HasProperty, "land airplane")
  - Grammar: incorrect
  - Truthfulness: never true
  - Consistency: highly consistent (*Our interpretation: While pilots do land airplanes, the HasProperty relation requires a description in the tail slot, so it's not grammatically correct or truthful.*)
- ("gross domestic product", HASPROPERTY, "abbreviated to CTBT")
  - Grammar: correct
  - Truthfulness: never true
  - Consistency: a little consistent (*Our interpretation: The gross domestic product does have a well-known abbreviation ("GDP"), so this statement could be construed as a little consistent.*)

HasSubevent The HASSUBEVENT relation describes sub-events or components of larger events. It requires an event (verb phrase or noun phrase) in the head slot and an event in the tail slot. Examples:

- ("lying", HASSUBEVENT, "you feel guilty")
  - Grammar: correct
  - Truthfulness: mostly true (*Our interpretation: Lying often causes guilt in people, although the amount of guilt depends on the person.*)
  - Consistency: highly consistent
- ("relax", HASSUBEVENT, "vegetable")
  - Grammar: incorrect
  - Truthfulness: never true
  - Consistency: not consistent at all
- ("drink coffee", HASSUBEVENT, "water may get into your nose")
  - Grammar: correct
  - Truthfulness: never true
  - Consistency: a little consistent (*Our interpretation: Drinking coffee doesn't cause water to get into your nose, but coffee and water are both drinkable liquids, so we think this statement is a little consistent.*)

ReceivesAction The RECEIVESACTION relation describes actions that apply to objects or other actions. It requires a verb phrase or noun phrase in the head slot and an action in the tail slot. Examples:

- ("book", RECEIVESACTION, "write by person")
  - Grammar: correct
  - Truthfulness: always true
  - Consistency: highly consistent

- ("most watches", RECEIVESACTION, "rhyme with piano")
  - Grammar: correct
  - Truthfulness: never true
  - Consistency: not consistent at all
- ("oil", RECEIVESACTION, "grow in field")
  - Grammar: correct
  - Truthfulness: never true
  - Consistency: a little consistent (*Our interpretation: Since oil is a natural resource similar to other things that are grown in fields, we could see this statement being a little consistent (it's a stretch though).*)
- ("violin", RECEIVESACTION, "play with a puck")
  - Grammar: correct
  - Truthfulness: never true
  - Consistency: somewhat consistent (*Our interpretation: Violins are indeed played, but with a bow, not a puck.*)

UsedFor The USEDFOR relation describes the uses of objects or actions. It requires a verb phrase or noun phrase in the head and tail slots. Examples:

- ("shoes", USEDFOR, "protecting feet")
  - Grammar: correct
  - Truthfulness: always true
  - Consistency: highly consistent
- ("tying your shoelace", USEDFOR, "smart")
  - Grammar: incorrect
  - Truthfulness: never true
  - Consistency: not consistent at all
- ("swimming", USEDFOR, "traveling on land")
  - Grammar: correct
  - Truthfulness: never true
  - Consistency: somewhat consistent (*Our interpretation: This statement is somewhat consistent because swimming and traveling on land are both means of movement.*)
- ("bush", USEDFOR, "wrestling on")
  - Grammar: correct
  - Truthfulness: never true
  - Consistency: not consistent at all