Can Pre-trained Models Really Generate Single-Step Textual Entailment?

Anonymous ACL submission

Abstract

We investigate the task of generating textual entailment (GTE). Different from prior works on recognizing textual entailment, also known as NLI, GTE requires the models with deeper reasoning capabilities - generating entailment from premises rather than making prediction on given premises and the entailment. We argue that existing adapted datasets are limited and inadequate to train and evaluate human-like reasoning in the GTE. In this paper, we propose a new large-scale benchmark, named $\mathcal{SEG}$, targeted for learning and evaluating models' capabilities towards RTE. $\mathcal{SEG}$ consists of 15k instances with each containing a pair of premise statements and a human-annotated entailment. It is constructed by first retrieving instances from a knowledge base, and then augmenting each instance with several complementary instances by 7 manually crafted transformations. We demonstrate that even extensively fine-tuned pre-trained models perform poorly on $\mathcal{SEG}$. The best baseline can only generate valid textual entailment for 59.1% cases. Further, to motivate future advances, we provide detailed analysis to show significant gaps between baselines and human performance.

1 Introduction

Textual entailment is an important and routine part of linguistic communication, whether in our daily lives or scientific literature (Korman et al., 2018). [1] Existing efforts focus on recognizing textual entailment (`entailment`, `contradiction` and `neutral`) between the premise and the hypothesis, also known as natural language inference (NLI) (Dagan et al., 2010; MacCartney and Manning, 2008). Recent powerful pre-trained models have achieved near human-level performance on NLI task (Devlin et al., 2019; Liu et al., 2019). It is time to challenge pre-trained models with more difficult entailment tasks. Therefore, we would ask whether *pre-trained models can generate the textual entailment from given premises?*

To answer the above questions, we first adapt existing entailment datasets (e.g., multip-hop question answering datasets, explainable NLI datasets) to the generative paradigm and then fine-tuned a pre-trained model (e.g., T5 (Raffel et al., 2019)) on these datasets. [2] Note that for simplicity, we focus on *single-step* textual entailment generation when given *two* premises. The preliminary results are surprisingly good (Table 1). The accuracy of human evaluation is above 80%. However, through in-depth analysis, we observe two main limitations to these datasets.

- The given premises always have valid entailment.

Table 1: The performance of T5-large on generating textual entailment task whose examples are constructed from existing datasets. "†" denotes the datasets on multi-hop QA task, and "‡" denotes the datasets on explainable NLI task.

Dataset	BLEU	Human
RuleTaker† (Clark et al., 2020)	100.0	100%
e-SNLI‡ (Camburu et al., 2018)	47.50	86%
EntailmentBank† (Dalvi et al., 2021)	47.57	84%
QASC† (Khot et al., 2020)	38.33	82%

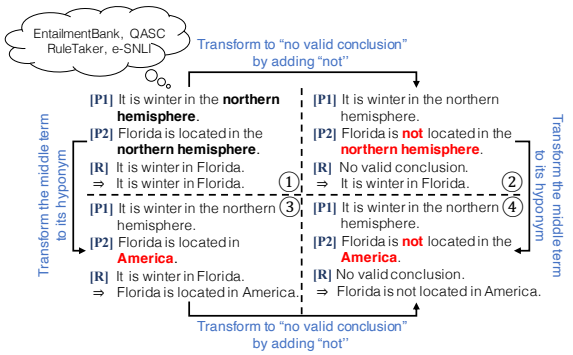

Figure 1: "P1" and "P2" denote the given two premises. "R" denotes the reference entailment. "⇒" denotes the output of finetuned T5-large model. **Black bold** font indicates middle items. **Red bold** font indicates the modifications. ① is chosen from the test data of adapted EntailmentBank (Dalvi et al., 2021). ② ③ ④ modifies the input premises by hand.

[1] In this paper, "entailment" means "textual entailment" by default.

[2] Please refer to the Appendix A.1 for details.

If we make some perturbations to the premises so that no entailment should be drawn, the model will fail and output an incorrect entailment. As shown in Figure 1 (① → ②), when we transform one premise to its negative form by adding the word "not", the model makes a mistake by still drawing the same conclusion as before.

- The two premises share the middle term with exactly the same lexical form.[3] If we modify the lexical form of the middle term in one premise, the model may fail because it does not capture the semantic relationship between the original middle term and the modified middle term. As shown in ① → ③ (Figure 1), when "northern hemisphere" is changed into "America", the model just replicates the second premise.

These limitations may reduce the difficulty for models to learn the ability of entailment generation, or enable models to learn some shortcuts, so that models can achieve a high accuracy. To push the development of models in generating textual entailment, it is necessary to introduce a more robust and powerful dataset.

In this paper, we define the task of generating textual entailment (GTE), and build $\mathcal{SEG}$ (**S**ingle-step textual **E**ntailment **G**eneration) to train and evaluate the models' ability towards this task. Overall, $\mathcal{SEG}$ contains about 15k instances, and each contains a pair of premise statements and a *human-annotated* entailment if valid, or a no valid entailment (NVE) if not. The dataset is constructed by first retrieving instances from a natural knowledge base, and then augmenting each instance with several complementary instances by 7 manually crafted transformations. These complementary instances are more helpful to train and evaluate the human-like reasoning ability of the models, i.e., to make valid entailment by distinguishing those lexically subtle but semantically important differences. In summary, our contributions include: [4]

- We formally define the task of generating textual entailment and build $\mathcal{SEG}$, a more robust and powerful dataset collected from natural corpus and complementary transformations, and checked by human annotators, which can help to train and evaluate the human-like reasoning ability of models.
- We evaluate several state-of-the-art NLP generative models on $\mathcal{SEG}$. [5] The best generator models can only generate valid textual entailment 59.1% of times. Further, to motivate future advances, we provide detailed analysis to show significant gaps between baselines and human performance.

2 Generating Textual Entailment

In this section, we first give the formal definition of textual entailment based on (Korman et al., 2018), and then describe the task of generating textual entailment (GTE).

Definition 1 *The entailing and entailed texts are premise* (P) *and hypothesis* (H)*, respectively.* P *textually entails* H *if and only if, typically, a human reading* P *would be justified in inferring the proposition expressed by* H *from the proposition expressed by* P*.*

Textual entailment in NLP is a directional relation between text fragments. The relation holds whenever the truth of one text fragment follows from another text. Textual entailment is not the same as pure logical entailment – it is a more relaxed definition. Another popular definition is that *a human reading* P *would infer that* H *is most likely true* (Dagan et al., 2010). For the detailed discussions of these two definitions please refer to (Korman et al., 2018).

In this work, we explore textual entailment in a generative paradigm. Specifically, given the premises that contains only *two premises* $P1$ and $P2$, the GTE task requires to generate the *single-step* textual entailment. Note that entailment must be based on two premises, neither of which alone can infer the entailment. For simplicity, we focus on the more basic settings: two premises and single-step. Even with this basic settings, there is still a huge gap between current models and human performance (see Section 5). More complex scenarios can be leave for future work, such as multi-premises, multi-step and multi-entailment.

3 $\mathcal{SEG}$: Data Collection and Analysis

$\mathcal{SEG}$ is a carefully designed benchmark for generating textual entailment, consisting of 15k instances in total, and each contains a pair of premise statements, and a human-annotated entailment if valid,

[3] Here, the middle term is the term appearing in both premises. It acts as an intermediary connecting given premises to draw conclusions. Formal definition can be referred in (Smiley, 1973).

[4] Data will be released upon the publication of this paper.

[5] In this paper, generative model refers to the seq2seq model (e.g., T5, BART) or auto-regressive model (e.g., GPT2).

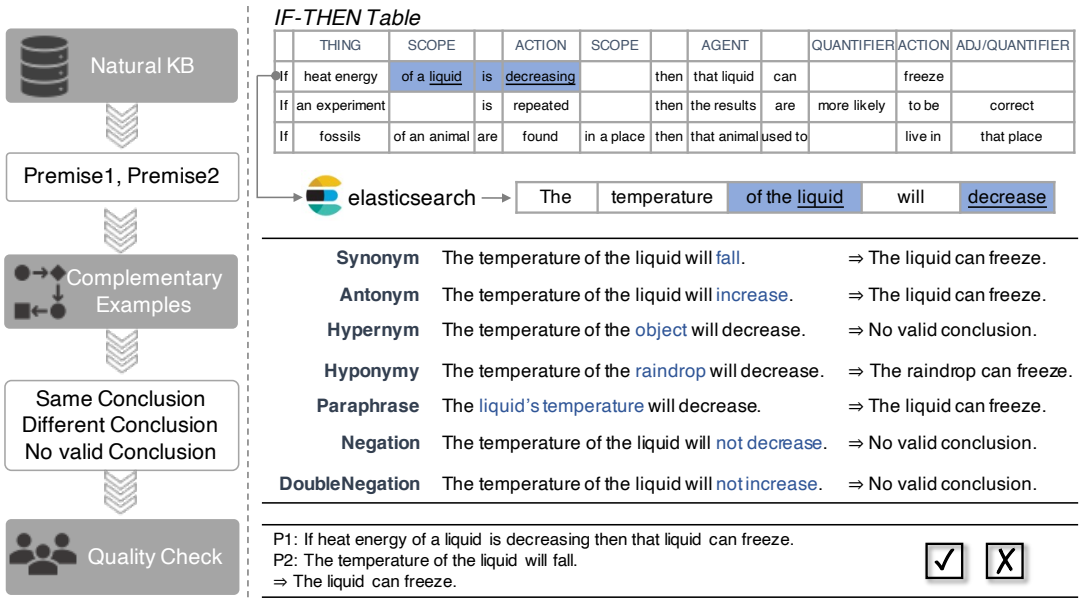

Figure 2: Dataset collection workflow, consisting of two stages: 1) collecting premises from a natural knowledge base named WorldTree, where knowledge is organized in tables, and 2) for those instances with valid conclusions, collecting from complementary transformations. During each stage, we hire workers to annotate and check the validation of entailments.

or a no valid entailment (NVE) if not. Overall, data collection procedure consists of two stages (Figure 2). To break the two limitations of adapted existing datasets mentioned in Section 1, we first retrieve two statements from knowledge base (KB) as two premises, who share the middle term with exactly the same lexical forms (Section 3.1). Then we construct several complementary instances by manually designing 7 transformations, which requires that the model can make valid entailment by distinguishing those subtle but semantically important lexical differences (Section 3.2). Furthermore, during human annotation, we design strict strategies to control the quality of annotation (Section 3.3). Finally we provide a detailed analysis of the proposed $\mathcal{SEG}$ (Section 3.4).

3.1 Stage 1: Collecting from Knowledge Base

In this section, we first describe the adopted KB. Then, we detail how to collect instances based on the KB, which follows the collection order of $P1 \rightarrow$ middle term $\rightarrow P2 \rightarrow$ entailment. All steps are automatical, except for the collection of entailment, which requires human annotation.

Knowledge Base WorldTree (Xie et al., 2020), a natural KB, is adapted as our source corpus. WorldTree provides a tablestore of sentences in terms of science and general knowledge. Each table is organized by a particular kind of relation (e.g.,"if-then" relation in Figure 2), [6] and columns of the table represent various roles or arguments to the specific relation.

Collecting P1 Given the WorldTree, we first randomly select a cerain table, and then randomly choose a sentence as $P1$ from the table.

Collecting Middle Term After obtaining a $P1$ in the table, we carefully select certain columns from a sentence in WorldTree as the middle term. The selection guideline is through human-designed templates listed in Appendix A.2.

Collecting P2 Given a $P1$ and a middle term, we use elasticsearch to retrieve $P2$ from the whole WorldTree. The retrieved candidates are then filtered based on the relevance scores (≥ 7) returned by the search engine. Finally, we keep at most 3 candidates as $P2$.

Collecting Entailment After obtaining $P1$ and $P2$, we work with an annotation service provider to annotate entailments for first-staged premises. Details are shown in Section 3.3.

3.2 Stage 2: Collecting from Complementary Transformations

After stage 1, we observe that despite the collection of instances with no valid entailment, there are middle terms with the same lexical form. To

[6] There are 81 kinds of relations in total.

Table 2: Descriptions of the functions of different transformations. "ΔLexical" and "ΔSemantic" denote the change of lexical and semantic caused by a certain transformation.

Transformation	Functions	ΔLexical	ΔSemantic
Synonym	Require to focus on semantic associations instead of shallow word overlap.	small	small
Antonym & Negation	Require to capture small perturbations, which may flip the entailments to NVE.	small	big
Hypernym & Hyponym	Require an extra monotonicity inference step to draw a new conclusion.	small	small
Paraphrase	Require to focus on semantic associations instead of sentence structure.	big	small
DoubleNegation	Require to judge whether a double negative forms a positive.	small	big

eliminate this obvious shortcut, we make careful transformations for each instance with valid entailment to obtain complementary instances. The criterion for the transformation is to perturb the original instance (only two premises) in two dimensions *lexical* and *semantic*, e.g., similar lexical and similar semantic, similar lexical and different semantic, different lexical and similar semantic. By constructing complementary instances, it can help to train and evaluate the human-like reasoning ability of the models, i.e., to generate valid entailment by distinguishing those lexically subtle but semantically important differences. Next, we describe how to collect premises from transformations automatically, and how to annotate entailment manually.

Collecting Premises from Transformations To construct complimentary instances, we design 7 transformations to cover the diversity on lexical and semantic (Table 2), including *synonym, antonym, negation, hypernym, hyponym, paraphrase, double negation*. It is possible to design more transformations or to combine them, and we leave it as future work. Details on how we apply these transformations can be referred in Appendix A.3.

Collecting Entailment After obtaining premises through transformations, We take the same procedure as in the stage 1 to collect the entailment.

3.3 Ethics and Quality Control

Before official annotation, we first conduct a trail phase for all candidate workers to fully understand the task and test their entailment ability. And we selected 80 qualified workers, both of them can achieve 80% accuracy on the trial data. Then we conduct a training session for selected workers to further enhance their science knowledge and basic inference skills needed in our data. All workers are categorized into two teams: a team of entailment constructors and a team of entailment checkers. The annotation process consists of two steps: 1) a construction step to write entailments given two premises, and 2) a double-round checking step for quality control.

Construction During construction stage, each instances with two premises are shown to three random workers. Workers need to write an entailment sentence if a valid entailment can be generated, otherwise give "NVE" label. Entailments are required to be: 1) derived from both premises instead of *none* or *only one* of them, and 2) fluent and no syntax errors.

First-Round Checking Afterwards, an instance with two premises and three candidate entailments is exposed to five checkers for first-round checking. Each checker should make an approval/disapproval decision for the annotation according to the two criterions mentioned in the construction stage. A candidate entailment is regarded as accepted only if at least 4 checkers approved the entailment annotation. Otherwise, a rejected entailment is sent back to the construction step for revision. Finally for each instances, we have three valid candidate entailments.

Second-Round Checking Instances with three candidate entailments after first-round checking are further fed into another two checkers for a second-round checking step. Here the two checkers focus on whether the two premises can necessarily draw a *one-and-only* entailment. If multiple entailments with completely different semantics can be derived from the two premises, the instance is abandoned. Finally the two checkers reach a decision by discussion, to choose an entailment from the three candidates as the gold entailment.

We paid RMB¥1 per constructor per instance for construction step, RMB¥0.96 per checker per instance for first-round checking, and RMB¥0.8 per checker per instance for second-round checking[7].

[7]Workers consist of both part-time and full-time employees, where we ensure full-time employees work at most 8 hours per day. And the local minimum hourly wage is RMB¥23 per hour.

Table 3: Dataset statistics. "#Total" denotes the number of instances constructed in each stage or under each transformations. "#Pos" and "#Neg" denote the number of instances with and without valid entailments respectively. "%Pos" denotes the ratio of instances with valid entailments.

	#Total	#Pos	#Neg	%Pos
Stage 1	6,677	3,140	3,537	0.47
Synonym	1,309	979	330	0.748
Antonym	1,181	299	882	0.253
Hypernym	999	241	758	0.241
Hyponymy	1,189	712	477	0.599
Paraphase	1,362	1,155	207	0.848
Negation	1,067	430	637	0.403
DoubleNegation	1,197	455	742	0.38
Stage 2	8,304	4,430	3,874	0.467
Overall	14,981	7,570	7,411	0.495

For the construction step, a worker can produce 3 entailments during two minutes. And averagely it costs a checker 30 seconds in the first-round checking and 20 seconds in the second-round checking. During the process of collecting $\mathcal{SEG}$, all of the natural corpus we used are sourced from publically available resources[8].

3.4 Dataset Statistics

Overall $\mathcal{SEG}$ has 14,981 instances. Summary statistics are shown in Table 3. In the first stage, the numbers of examples with and without valid entailments are comparable, validating the efficiency of our method to identify textual entailment from natural knowledge base. In the second stage, examples under different transformations are unbalanced across labels, as denoted by "%Pos". Transformations in terms of "Synonym", "Hyponymy" and "Paraphase" tend to keep the labels of examples unchanged while others flip the labels. The dataset is further split into training, validation and testing sets with 12648, 1826, 3708 examples respectively. Examples transformed from the same original instance are guaranteed in the same split.

4 Experimental Settings

We design experiments to benchmark state-of-the-art NLP generative models on the proposed generating textual entailment dataset $\mathcal{SEG}$. We introduce different tasks (Sec 4.1) to evaluate models from different perspectives under several automatic evaluation metrics (Sec 4.2), and also propose an ensemble metric for better evaluation.

[8]WorldTree corpus can be downloaded at http://cognitiveai.org/explanationbank/, and during transformation we use WordNet(database and associated tools can be downloaded at https://wordnet.princeton.edu/download)

4.1 Task Definition

Since instances in $\mathcal{SEG}$ are annotated with either "NVE" labels or valid entailments, we define three tasks to evaluate models from different perspectives. For all of the three tasks, the inputs are concatenation of two premises.

Task 1: Classification Since nearly half of samples in $\mathcal{SEG}$ are with no valid entailments, this task aims to evaluate the ability of models to distinguish whether two premises can generate valid entailments. Given two premises, the output is a binary label, "yes" for premises having valid entailments and "no" otherwise.

Task 2: Generation For samples with valid entailments, this task aims to evaluate the quality of model-generating entailments. Consider samples with valid entailments in $\mathcal{SEG}$, the output is to generate the textual entailment from given two premises.

Task 3: Multi-task This task aims to evaluate whether models can perform classification and generation tasks simultaneously. The outputs of this task can be either a short phase "no valid entailment" indicating "NVE", or any valid entailments derived from premises.

4.2 Evaluation Metrics

We adopt different evaluation metrics for classification and generation tasks.

For classification task, besides standard accuracy, we introduce two new metrics, **pairwise accuracy** and **group accuracy**. Pairwise accuracy evaluates a pair of predictions as accurate if both predictions of the original and a transformed instance are correct. Similarly, group accuracy evaluates as accurate if predictions of the original and all its transformed instances, which form a group, are correct.

For generation task, we consider the following automatic metrics: BLEU (Papineni et al., 2002), ROUGE (Lin, 2004), BERTScore (Zhang et al., 2019), SentenceBERT (Reimers and Gurevych, 2019) and BLEURT (Sellam et al., 2020). For BLEURT, we also consider a finetuned version BLEURT†[9]. We show the agreement of these metrics with human ratings in Table 4. As the table

[9]To finetune BLEURT, we annotate 1,000 pairs of reference and candidate textual entailments for their validations (valid ones score "1" and invalid ones score "−1"). Among them, 800 pairs are used for finetuning and 200 for testing correlations with human ratings.

Table 4: Pearson correlation τ with human ratings of different metrics of text generation. BEURT† denotes BLEURT finetuned on human ratings.

Metric	BLEU	ROUGE-1	ROUGE-2	BERTScore
τ	0.393	0.487	0.447	0.448
Metric	SentenceBERT	BLEURT	BLEURT†	Ensemble
τ	0.518	0.571	0.653	0.707

Table 5: Performance(%) of baseline models on classification task. "Standard" denotes standard accuracy. "Pairwise" and "Group" denote pairwise and group accuracy respectively.

Model	Standard	Pairwise	Group	Precision	Recall	F1-score
T5	72.4	55.1	35.7	69.8	79.2	74.2
BART	73.7	53.9	35.0	75.0	71.4	73.2
GPT2	70.0	47.1	27.8	70.3	69.8	70.0
LSTM	64.8	41.1	24.5	66.4	60.3	63.2

shows, most automatic metrics have weak correlations with human ratings. This may be because most metrics mainly focus on textual similarity rather than logical validation. Finetuning BLEURT on human ratings can improve correlations. Inspired by this, we further train a model-based (MLP classifier used here) metric, denoted as the ensemble metric, by combining the above metrics as input features to predict human judgments. The ensemble metric exhibits the best correlation with human judgments as shown in Table 4. Once we get the ensemble metric, we can use it to evaluate the validation of the generated entailments. We define standard validation as the fraction of valid entailments among all generated ones. Similar to pairwise/group accuracy, we also introduce **pairwise validation** and **group validation** for generation task, which evaluate as valid if all generated entailments in a pair or group are valid.

Baselines We consider the following generative models as the baselines: LSTM (Hochreiter and Schmidhuber, 1997), GPT2-large (Radford et al., 2019), BART-large (Lewis et al., 2019) and T5-large (Raffel et al., 2019). Details on training these models are listed in Appendix A.4.

5 Results

5.1 Classification Task

Table 5 shows the performance of baseline models on classification task. Overall, all models achieve mediocre performance in terms of standard accuracy while the pairwise and group accuracy is low. This indicates that the models can infer each premise independently with confidence, but on the other hand struggle to give consistent judgements for the complementary instances.

For the comparison of different models, T5 and BART achieve the highest performance of most metrics among all baselines. T5 has a higher recall rate while BART has a higher precision rate. This indicates that T5 tends to predict that the given premises have valid entailments while BART tends to be the opposite.

5.2 Generation Task

Table 6 shows the performance of baseline models on the generation task. Overall, pre-trained models achieve mediocre performance in terms of the validation metric, while LSTM can barely generate valid entailments. On the whole, T5 model outperforms others models. In 66% cases, it can generate valid entailments. In 42.3% cases, it can generate valid entailments for both the original and transformed examples. BART performs worst among three pre-trained models. We find that BART tends to duplicate one premise as the conclusion. This may be related to the objective of reconstructing the input during the model pre-training. In terms of different evaluation metrics, even though the entailments generated by LSTM are almost invalid, the score of BLEU still achieves 31.0. The same applies to metrics like ROUGE. This implies that the absolute values of these metrics are not very meaningful for this task.

5.3 Multi-Task

Under this task, we investigate whether the baseline generative models can perform classification and generation tasks simultaneously. Table 7 shows the performance of baseline models on this task.

The experimental results show that the performance of the baselines is poor under this task. The accuracy on the multi-task declines compared with that on classification task, indicating that it is challenging for the models to perform the classification and generation task simultaneously. Among all models, the best generator model can only achieve 59.1% in terms of validation, which means that even SOTA pre-trained models can only generate valid textual entailment 59.1% of times.

5.4 Ablation Studies

Performance under Different Transformations We report the performance of models under different transformations in Figure 3 (denoted by "★" and "★" for classification and generation task respectively).

Table 6: Performance of baseline models on generation task. "Human" denotes human evaluation on 100 random samples. "Group"/"Pairwise" denote group/pairwise validation respectively. "BLEURT†" denotes BLEURT finetuned on human ratings.

Model	Human	Standard	Pairwise	Group	BLEU	ROUGE-1	ROUGE-2	BERTScore	SentenceBERT	BLEURT	BLEURT†
T5	72.0	66.9	50.8	42.3	55.4	79.9	65.3	0.875	0.910	0.762	0.469
BART	56.0	47.8	35.6	28.3	50.6	75.8	60.9	0.852	0.854	0.708	0.135
GPT	65.0	57.5	41.2	33.7	48.5	76.2	59.1	0.852	0.891	0.727	0.353
LSTM	15.0	11.6	8.0	4.8	31.0	57.9	39.4	0.738	0.640	0.512	−0.633

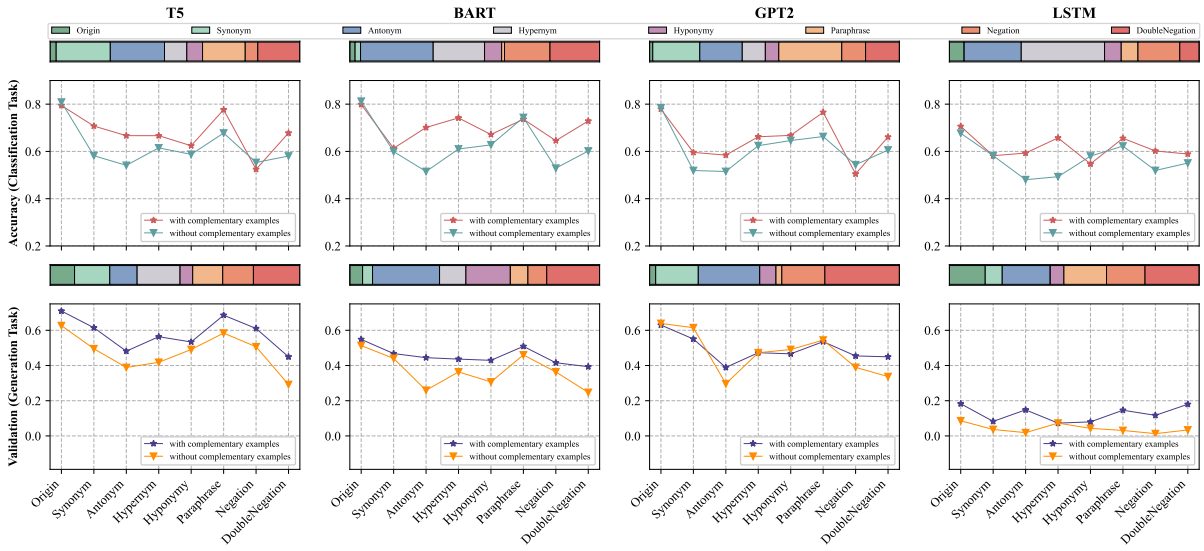

Figure 3: Performance(%) of the baselines on the classification and generation task under different transformation. "★" and "★" denotes models trained with complementary examples. "▼" and "▼" denotes models trained without complementary examples. We show the performance changes brought by complementary examples under different transformations via stacked bar. Performance changes of each transformation are indicated by the length of bars of different colors.

Table 7: The overall performance(%) of baseline models on multi-task. The metric validation is evaluated on the whole test set. A generated conclusion is evaluated as valid if both the predicted category is correct and the conclusion is valid.

Model	Accuracy			Validation			Human
	Standard	Pairwise	Group	Standard	Pairwise	Group	
T5	65.4	48.5	28.8	54.9	30.6	16.0	55.0
BART	69.5	40.9	24.3	59.1	23.3	13.6	58.0
GPT	67.3	38.0	22.9	58.7	24.3	13.1	57.0
LSTM	60.6	24.3	14.2	50.6	8.3	5.9	51.0

Overall, the models perform well on the original and paraphased instances. For classification task, the models are poor at classifying instances under "Negation" transformation. T5 and GPT2 can only achieve 50% classification accuracy on these examples, nearly the same as the random guess. What's more, in terms of "Synonym" transformation, the baselines except T5 can only achieve about 60% accuracy. This implies that when we transform the middle terms of given premises into their synonyms, these models may fail to capture the semantic associations without word overlap. For generation task, the models perform poor under transformations, such as "Antonym" and "DoubleNegation", with no more than 40% validation under these two transformations.

Influence of complementary examples To investigate the influence of complementary examples collected in stage 2, we train baseline models without complementary examples, e.g. only with data collected in the first stage. We then evaluate the trained models on the classification and generation task and report the performance (Table 8).

Table 8: Performance of baseline models trained on examples without complementary examples of the classification (Accuracy) and generation task (Validation) respectively.

Accuracy	Standard	Pairwise	Group
T5	68.7(**3.6** ↓)	46.4(**8.7** ↓)	26.2(**9.4** ↓)
BART	69.8(**4.0** ↓)	48.0(**6.0** ↓)	26.9(**8.2** ↓)
GPT	67.5(**2.6** ↓)	39.8(**7.4** ↓)	21.1(**6.7** ↓)
LSTM	60.7(**4.1** ↓)	32.9(**8.2** ↓)	18.1(**6.4** ↓)
Validation	**Standard**	**Pairwise**	**Group**
T5	54.6(**12.3** ↓)	40.4(**10.4** ↓)	33.7(**8.6** ↓)
BART	43.6(**4.2** ↓)	29.1(**6.5** ↓)	22.0(**6.3** ↓)
GPT	55.8(**1.8** ↓)	39.6(**1.6** ↓)	30.0(**3.7** ↓)
LSTM	5.7(**5.9** ↓)	2.6(**5.5** ↓)	1.5(**3.2** ↓)

Overall, the performance of all models has declined without training on complementary examples. For classification task, the group and pairwise accuracy decline more than the standard accuracy. But for generation task, all three metrics drop

evenly. Among all models, GPT2 is least affected by complementary examples with validation scores decrease by only $1\% \sim 4\%$.

We also report performance under different transformations of the classification and generation task respectively (denoted by "▼" and "▼" in Figure 3). we further quantitatively compare the performance changes brought by complementary examples under different transformations. We find that the complementary examples mostly benefit "Antonym" and "DoubleNegation" transformation across different models and benefit performance on original examples least.

5.5 Transfer from Related Datasets

In this section, we evaluate models transferred from other datasets to our test set. This experiment is conducted to show that whether inference abilities acquired from other dataset are sufficient to handle our test cases. We compare models trained on the following datasets: EntailmentBank (Dalvi et al., 2021), QASC (Khot et al., 2020), ParaPattern (Bostrom et al., 2021), e-SNLI (Camburu et al., 2018) and RuleTaker (Clark et al., 2020).[10] We evaluate T5 model under the same setting with *Generation Task*. Specifically, we evaluate models only on instances with valid entailments, based on the consideration that previous datasets don't include the cases of "NVE".

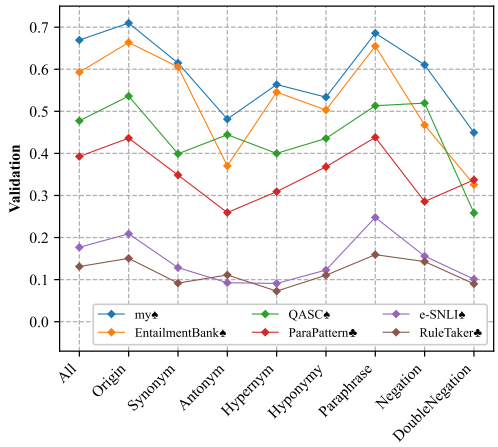

Figure 4: Validation of the T5 model trained on different datasets under the generation task. "♣" denotes datasets built on logic templates. "♠" denotes datasets built with human annotations. "All" denotes the overall performance on the generation task, while other labels of x-axis denote performance under different transformations.

Figure 4 shows the validation of conclusions generated by T5 trained on different datasets. As the figure shows, performance varies widely across datasets. T5 trained on RuleTaker and e-SNLI can hardly transfer to our dataset. ParaPattern, consisting of two logical types, substitution and contraposition, is not enough to cover inference types in our dataset. EntailmentBank and QASC show higher transferability compared with the above three datasets. However, they still have difficulties in solving transformations of "Antonym" and "DoubleNegation".

[10] Among them, RuleTaker and ParaPattern are datasets constructed by filling logic templates. The other three datasets are built with human involved.

6 Related Work

Camburu et al. (2018); Peng et al.; Rudinger et al. (2020) built explanation generation datasets by extending SNLI (Bowman et al., 2015) with human-annotated full-sentence explanations of the classification decisions. Rajani et al. (2019) collected human explanations for commonsense reasoning built on top of CommonsenseQA (Talmor et al., 2019) and introduced CoS-E. Bhagavatula et al. (2020) formally proposed language-based abductive reasoning. They built the $\mathcal{ART}$ dataset by crowdsourcing the plausible and implausible explanation options for observations. Our formulation is critically distinct from abductive reasoning. $\mathcal{ART}$ requires reasoning about commonsense implications of observations, with less focus on logical inference. However, we focus more on logical reasoning rather than commonsense reasoning, since the knowledge required for inference is explicitly provided in premises.

Combining multiple premises to form a conclusion overlaps with the idea of multi-hop reasoning. In the context of textual question-answering (QA), recent work has shown that deep models can perform multi-hop reasoning by generating proof graphs, with facts and rules expressed in natural language (Clark et al., 2020; Dalvi et al., 2021; Tafjord et al., 2020; Sun et al., 2021; Khot et al., 2020). Inspired by this, we dig further into the basic and fundamental single-step inference and build more complex reasoning scenarios to evaluate pre-trained generative models.

7 Conclusion

We define the task of generating textual entailment and build a challenging dataset $\mathcal{STEG}$. For future work, benchmarks can be built towards multi-hop textual entailment generation to enable complex reasoning capabilities in AI systems.

References

Chandra Bhagavatula, Ronan Le Bras, Chaitanya Malaviya, Keisuke Sakaguchi, Ari Holtzman, Hannah Rashkin, Doug Downey, Wen-tau Yih, and Yejin Choi. 2020. Abductive commonsense reasoning. In ICLR.

Kaj Bostrom, Xinyu Zhao, Swarat Chaudhuri, and Greg Durrett. 2021. Flexible generation of natural language deductions. In Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing.

Samuel R. Bowman, Gabor Angeli, Christopher Potts, and Christopher D. Manning. 2015. A large annotated corpus for learning natural language inference. In Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, pages 632–642. Association for Computational Linguistics.

Oana-Maria Camburu, Tim Rocktäschel, Thomas Lukasiewicz, and Phil Blunsom. 2018. e-snli: Natural language inference with natural language explanations. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, Advances in Neural Information Processing Systems 31, pages 9539–9549.

Peter Clark, Oyvind Tafjord, and Kyle Richardson. 2020. Transformers as soft reasoners over language. In IJCAI.

Ido Dagan, Bill Dolan, Bernardo Magnini, and Dan Roth. 2010. Recognizing textual entailment: Rational, evaluation and approaches–erratum. Natural Language Engineering, 16(1):105–105.

Bhavana Dalvi, Peter Jansen, Oyvind Tafjord, Zhengnan Xie, Hannah Smith, Leighanna Pipatanangkura, and Peter Clark. 2021. Explaining answers with entailment trees. In Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing.

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.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. Neural computation, 9(8):1735–1780.

Tushar Khot, Peter Clark, Michal Guerquin, Peter Jansen, and Ashish Sabharwal. 2020. Qasc: A dataset for question answering via sentence composition. In Proceedings of the AAAI Conference on Artificial Intelligence, volume 34, pages 8082–8090.

Daniel Z Korman, Eric Mack, Jacob Jett, and Allen H Renear. 2018. Defining textual entailment. Journal of the Association for Information Science and Technology, 69(6):763–772.

Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Ves Stoyanov, and Luke Zettlemoyer. 2019. Bart: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. arXiv preprint arXiv:1910.13461.

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

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.

Bill MacCartney and Christopher D Manning. 2008. Modeling semantic containment and exclusion in natural language inference. In Proceedings of the 22nd International Conference on Computational Linguistics (Coling 2008), pages 521–528.

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.

Shiya Peng, Lu Liu, Chang Liu, and Dong Yu. Exploring reasoning scheme: A dataset for syllogism figure identification.

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. 2019. Exploring the limits of transfer learning with a unified text-to-text transformer. arXiv preprint arXiv:1910.10683.

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.

Rachel Rudinger, Vered Shwartz, Jena D Hwang, Chandra Bhagavatula, Maxwell Forbes, Ronan Le Bras, Noah A Smith, and Yejin Choi. 2020. Thinking like a skeptic: Defeasible inference in natural language. In Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: Findings, pages 4661–4675.

Thibault Sellam, Dipanjan Das, and Ankur Parikh. 2020. Bleurt: Learning robust metrics for text generation. In Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, pages 7881–7892.

Timothy J Smiley. 1973. What is a syllogism? Journal of philosophical logic, pages 136–154.

Changzhi Sun, Xinbo Zhang, Jiangjie Chen, Chun Gan, Yuanbin Wu, Jiaze Chen, Hao Zhou, and Lei Li. 2021. Probabilistic graph reasoning for natural proof generation. arXiv preprint arXiv:2107.02418.

Oyvind Tafjord, Bhavana Dalvi Mishra, and Peter Clark. 2020. Proofwriter: Generating implications, proofs, and abductive statements over natural language. arXiv preprint arXiv:2012.13048.

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.

Xiao Wang, Qin Liu, Tao Gui, Qi Zhang, Yicheng Zou, Xin Zhou, Jiacheng Ye, Yongxin Zhang, Rui Zheng, Zexiong Pang, et al. 2021. Textflint: Unified multilingual robustness evaluation toolkit for natural language processing. In Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing: System Demonstrations, pages 347–355.

Zhengnan Xie, Sebastian Thiem, Jaycie Martin, Elizabeth Wainwright, Steven Marmorstein, and Peter Jansen. 2020. Worldtree v2: A corpus of science-domain structured explanations and inference patterns supporting multi-hop inference. In Proceedings of the 12th Language Resources and Evaluation Conference, pages 5456–5473.

Boon Peng Yap, Andrew Koh, and Eng Siong Chng. 2020. Adapting bert for word sense disambiguation with gloss selection objective and example sentences. In Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: Findings, pages 41–46.

Tianyi Zhang, Varsha Kishore, Felix Wu, Kilian Q Weinberger, and Yoav Artzi. 2019. Bertscore: Evaluating text generation with bert. In International Conference on Learning Representations.

A Appendix

A.1 Adapting Existing Datasets into Generative Textual Entailment Paradigm

Single-step entailment generation scenario implicitly exists across various tasks, such as multi-hop QA and explainable NLI.

In the task of explainable NLI, e-SNLI (Camburu et al., 2018) extends the SNLI(MacCartney and Manning, 2008) dataset with additional human-annotated natural language explanations of the entailment relations. To adapt e-SNLI into generative paradigm, we extract `premise` and `explanation` as the pair of premises, and the `hypothesis` as the conclusion.

In the task of multi-hop QA, RuleTaker(Clark et al., 2020) targets at generating the answer for a question by emulating deductive reasoning over facts and rules, which are expressed in natural language. Tafjord et al. (2020) augments RuleTaker by introducing logical implications of statements. We extract all 1-step implications derived from only two facts or rules. The `facts or rules` can be used as the premises and the `implication` as the conclusion. EntailmentBank (Dalvi et al., 2021) can be used to generate explanations in the form of multi-step entailment trees, namely a tree of entailment steps from facts, through intermediate conclusions, to the final answer. The `entailment steps` are the data we need: deriving intermediate conclusions from given facts. QASC is a dataset for two-hop QA. It provides annotation for supporting facts as well as their composition to answer a question. The annotated facts can be adapted into generative paradigm by using `two facts` as the premises, and the `composed fact` as the conclusion.

A.2 Construct Premises based on WorldTree Corpus

For each table in the WorldTree, we manually define templates describing columns selected for retrieving candidate premises. Each template defines two types of columns. The required columns containing content which must be satisfied when building query for search, while the preferred columns are used as an additional ranking criterion when multiple sentences satisfying the required columns. In addition, we stem the words and remove stopwords in sentences before querying.

A.3 Construct Complementary Examples

In this part, we first describe how we apply the first four transformations on one premise. The first four transformations are based on lexical replacement of the middle term. To identify the middle term of given premises, we first stem words and remove stopwords in two premises. Then, we use the common words contained in both premises as the middle term. Next, for each word in the middle term, we can replace it with its synonyms, antonyms, hypernyms or hoponymys from WordNet. As there are some cases where using WordNet for substitution leads to unnatural sentences due to the context mismatch, we determine which "sense" (meaning) of a word in the WordNet is activated before word substitution. We adopt a BERT-based model (Yap et al., 2020) to solve this word sense disambiguation problem. To further ensure data quality, we use a grammar error correction tool Gingerit [11] to correct spelling and grammar mistakes in the perturbed premises.

For the latter three transformations, we use TextFlint (Wang et al., 2021) to transform premises to its paraphased, negative and double negative forms and use Gingerit for grammar error correction.

The above effort are made to ensure the fluency of the transformed premises. Though, annotators are allowed to refuse to annotate certain samples if they think the premises are not natural. The proportion of rejected samples is limited to below 5%.

A.4 Experimental Details

We describe details on the training of baseline models here. For classification task, all baseline models are fed with concatenation of two premises and output labels of “yes/no”, indicating whether or not the given premises entail any valid conclusions. For generation task, LSTM, BART and T5 models are trained by taking the concatenation of two premises as input and output conclusions. GPT2 is finetuned on data by concatenating the premises and conclusions, to maximize a causal language modeling objective . During inference, GPT2 takes in two premises and generates the conclusion.

All implementations are based on Fairseq[12]. The LSTM model consists of a 2-layer encoder and 2-layer decoder with hidden size set to 512. All mod-

[11] https://gingerit.readthedocs.org

[12] https://github.com/pytorch/fairseq

els are train to maximize the cross entropy loss with label smoothing set to 0.1. The mini-batch size is 32. The training epoch is 10 for pretrain models and 20 for LSTM. The models are optimized via Adafactor (Yap et al., 2020) with polynomial learning rate decay. The learning rate is 2e-5 for pretrained models and 1e-3 for LSTM. The beam size is set to 1 for GPT2 and 5 for other models. All models are train on 1 Tesla V100 GPU.

A.5 Inference Types

To understand the inference challenges in our dataset, we analyze model performance under different inference types. We identify 6 challenging high-level categories of inference in our dataset and manually select 50 samples for each inference type. A T5-large model trained on multi-task is evaluated in terms of classification accuracy and generation validation. Table 9 shows the identified inference types and the performance of T5. More descriptions about these inference types and typical examples can be referred in Appendix A.5.

Table 9: Standard accuracy and validation(%) of T5 under different inference types.

Inference Types	Accuracy	Validation
Containment Relationship	0.78	0.66
Relational Inference	0.62	0.55
Modal Judgement	0.56	0.54
Cause-Effect	0.65	0.64
Hypothetical Inference	0.64	0.52
Monotonicity Inference	0.54	0.44

We identify six challenging high-level categories of inference in our dataset. Specifically, *containment relationship* inference requires a determination of the containment relationship of two objects. *Relational inference* requires relational judgments between two objects. *Modal judgement* refers to inference requiring a determination of the possibility and necessity of events. *Cause-effect* inference requires to determine the response of an effect variable when a cause of the effect variable is changed. hypothetical inference *Hypothetical inference* requires to deduce from a specific rule, which is specified as one of the given premise. *Monotonicity inference* requires to replace a phase with a more specific or general one to derive a valid conclusion.

For further understanding of model behavior under different inference types, we list several typical cases and model outputs in Table 10.

Table 10: Typical cases of six inference types identified in our dataset $\mathcal{SEG}$. We also provide conclusions generated by T5 and human evaluations on these conclusions.

Containment Relationship	**premise1**: index fossils are formed from species that existed for a relatively short time. **premise2**: trilobite fossils are a kind of index fossil. **conclusion**: trilobite fossils are formed from species that existed for a relatively short time. ✔
	premise1: limestone is formed by water evaporating from a solution of water and minerals. **premise2**: limestone is a kind of sedimentary rock. **conclusion**: limestone is formed by water evaporating from a solution of water and minerals. ✗
Relational Inference	**premise1**: magnesium is less dense than nickel. **premise2**: silicate is less dense than nickel. **conclusion**: magnesium and silicate are less dense than nickel. ✔
	premise1: a law is more tested than a theory. **premise2**: a theory is more tested than a hypothesis. **conclusion**: no valid conclusion. ✗
Modal Judgement	**premise1**: crops does not cause nutrients in the soil to increase. **premise2**: a decrease in soil fertility has a negative impact on environment. **conclusion**: no valid conclusion. ✔
	premise1: elements are not impure substances. **premise2**: a pure substance is made of only one kind of atom. **conclusion**: no valid conclusion. ✗
Cause Effect	**premise1**: using less resources usually causes money to be saved. **premise2**: saving money has a positive impact on a person. **conclusion**: using less resources has a positive impact on a person. ✔
	premise1: using less resources usually does not cause money to be saved. **premise2**: saving money has a positive impact on a person. **conclusion**: no valid conclusion. ✗
Hypothetical Inference	**premise1**: an atom is always neutral charged. **premise2**: if an atom has neutral charge, then the atom will have same numbers of protons and electrons. **conclusion**: the atom has same numbers of protons and electrons. ✔
	premise1: if something is made of a material then that something contains that material. **premise2**: a kit usually contains materials for assembling something. **conclusion**: a kit usually contains materials for assembling something. ✗
Monotonicity Inference	**premise1**: Montenegro is located in the northern geographic region. **premise2**: if it is winter in the northern hemisphere then it is summer in the southern hemisphere. **conclusion**: if it is winter in Montenegro then it is summer in the southern geographic region. ✗
	premise1: Croatia is located in the northern, eastern hemisphere. **premise2**: march is cold in temperature in northern hemisphere. **conclusion**: no valid conclusion. ✗