Linguistically Conditioned Semantic Textual Similarity

Anonymous ACL submission

Abstract

Semantic textual similarity (STS) is a fundamental NLP task that measures the semantic similarity between a pair of sentences. In order to reduce the inherent ambiguity posed from the sentences, a recent work called Conditional STS (C-STS) has been proposed to measure the sentences' similarity conditioned on a certain aspect. Despite the popularity of C-STS, we find that the current C-STS dataset suffers from various issues that could impede proper evaluation on this task. In this paper, we reannotate the C-STS validation set and observe an annotator discrepancy on 55% of the instances resulting from the annotation errors in the original label, ill-defined conditions, and the lack of clarity in the task definition. After a thorough dataset analysis, we improve the C-STS task by leveraging the models' capability to understand the conditions under a QA task setting. With the generated answers, we present an automatic error identification pipeline that is able to identify annotation errors from the C-STS data with over 80% F1 score. We also propose a new method that largely improves the performance over baselines on the C-STS data by training the models with the answers. Finally we show the conditionality annotation can benefit from the typed-feature structure (TFS) of entity types. Our pilot annotation study shows that the TFS is able to provide a linguistic foundation for constructing C-STS data with new conditions.

1 Introduction

Semantic textual similarity (STS) is an essential NLP task that measures the semantic similarity between two sentences (Agirre et al., 2012). It is also a popular benchmark for developing tasks such as text embedding learning (Conneau and Kiela, 2018; Reimers and Gurevych, 2019; Thakur et al., 2021) and language understanding (Wang et al., 2018). While the STS datasets have been developed and improved over the past years (Agirre et al., 2013, 2014, 2015, 2016; Cer et al., 2017), the task itself still suffers from sentence ambiguity and subjectivity to judgment (Deshpande et al., 2023).

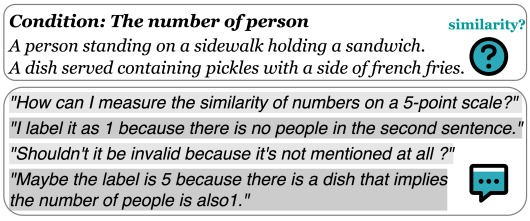

Figure 1: A problematic example from the C-STS dataset. The binarity of the condition cannot be mapped to a 5-point similarity scale. The label can be subjective depending on how much inference is made from the context. No guideline on the scenario when the information regarding the condition is missing.

A new task called Conditional STS (C-STS) has been proposed to resolve those issues (Deshpande et al., 2023). It is designed to disambiguate the similarity between two sentences by measuring the similarity on a given condition. An accompanying dataset was also proposed to test models on the C-STS task. Despite the popularity of C-STS, we observe certain limitations in the C-STS dataset that could hinder the understanding and proper evaluation of models on this task. As illustrated in Figure 1, these limitations primarily revolve around annotation errors, ill-defined conditions, and a general lack of clarity in task definition.

Taking into account the significance of these issues, we intend to improve the C-STS dataset by addressing the existing problems that we observed. We start by reannotating the C-STS validation set. By identifying an apparent annotation error rate of 55%[1] in their validation set, we analyze the provenance of the errors and discrepancies between the original and relabeled datasets.

To further explore the utility of the condition and

[1] Calculated from the comparison between the original and reannotated labels.

how it is understood by language models, we treat it as a Question Answering (QA) task and leverage large language models (LLMs) to generate the answer to the question that is constructed from the condition. We find that the LLM-generated answers can better capture the similarity between two sentences and fit closely to our reannotated labels by having a higher Spearman's Correlation. Based on this finding, we propose an approach to identify potential annotation errors from the C-STS dataset utilizing the LLM-generated answers, achieving over 80% F1 score on the validation set. We also propose a new method to improve the C-STS task by training the models with the answers. We show that both supervised and generative models can efficiently and effectively learn the condition information encoded in the answers, improving the performance over baselines by a large margin.

Finally, we propose a new annotation specification of the conditionality that aims to improve the formulation of the conditions with a more concrete semantic base. We use the entity type identified from the sentence pair as the surface condition text that is described by its underlying typed-feature structure (TFS) (Carpenter, 1992; Copestake, 2000; Penn, 2000). We conduct a pilot annotation study with the new conditionality to show that TFS-based conditions can be successfully adopted to sentence pairs from the current C-STS dataset. We will make the source code and dataset publicly available.

2 Related Work and Background

Semantic similarity tasks The semantic similarity between texts has long been a key issue in understanding natural language. Agirre et al. (2012) proposed the first STS task that measures the similarity between a sentence pair. Following this, Agirre et al. (2013) proposed the second STS task that covered more text genres in the dataset. There are also STS tasks (Agirre et al., 2014, 2015, 2016; Cer et al., 2017) with focuses on measuring sentence similarity under a multilingual and cross-lingual setting. Abdalla et al. (2023) proposed a new text similarity task that measures the semantic relatedness between two sentences.

Conditional STS More related to our work, the C-STS task (Deshpande et al., 2023) introduced conditions on top of the traditional STS tasks, and it measured the sentence similarity regarding the given condition. The C-STS dataset includes 18,908 instances. Each instance contains a sentence pair, a condition, and a scalar for the similarity score on the 5-point Likert scale (Likert, 1932). In this paper, we conduct the annotation and experiments on the C-STS validation set that consists of 2,834 instances. Deshpande et al. (2023) evaluated the C-STS dataset on different baselines such as SimCSE (Gao et al., 2021) and GPT models (Brown et al., 2020; OpenAI, 2023) by training or prompting with the sentence pairs and the conditions directly. However, our QA-based method uses the generated answers as the model input.

QA-facilitated tasks Question answering tasks are useful for extracting and inferring information from the texts that is relevant to the question. Recent work utilized QA to improve other NLP tasks. Eyal et al. (2019) and Deutsch et al. (2021) applied QA as an automatic evaluation metric for the summarization. Gunasekara et al. (2021) used QA to improve the summarization directly. Other works involved the application of QA for data augmentation (Mekala et al., 2022) and question generation (Tu et al., 2022). In this paper, we apply QA to generate condition-based answers for error identification and to improve models on the C-STS task.

3 Reannotating C-STS

We analyze the dataset and describe the annotation process for relabeling the C-STS validation set.[2]

Condition analysis We analyze the composition of the condition texts in the dataset. We observe that the majority of the conditions are short phrases in the format of *[feature] of [entity]* (e.g., *the color of animals*) or simply a singleton *[entity]* (e.g., *the hobby*). By plotting the distributions of frequent features and entities from the condition texts in the full C-STS dataset (Appendix A.1), we notice that the dataset is skewed by having a long tail distribution of the conditions. The top 10 frequent features and entities appear in 88% and 45% of the total conditions respectively.

Annotation Analysis To understand how conditions affect the human judgment of sentence similarity, we conduct a pilot annotation study on 150 instances sampled based on the frequency of condition features (e.g., *type*, *number*, etc.) in the C-STS training set. Annotators are asked to reannotate

[2]The original labels of the C-STS test set is not publicly available, so we use the validation split for further annotation and experiments in this paper.

Sentence Pair	Condition	Label	Issue
Female tennis player, standing on one foot after returned the ball. A tennis player is getting ready to hit the ball at a tennis match.	number of people	3 / 5	Incommensurable Mapping
An oak table and chairs in a dining room, with a doorway to the kitchen. Three couches positioned around a coffee table in a living room.	the table	1 / 3	Ambiguous Condition
A toilet in a stall with a sink attached and a console attached to the lid. A bathroom with tiled walls, a toilet, sink and a garbage can in it.	room type	2 / 4	Inference Degree
A cat hissing as it attempts to fit itself into a bowl that it is to big to fit in. A black cat with a red tag sitting down with a bookshelf in the background.	color of animal	2 / -1	Invalid Condition

Table 1: Examples with common issues that cause the judgment divergence between the original and reannotated labels. Text that is relevant to the conditions is highlighted.

those instances following the public C-STS annotation guideline. We measure the agreement between the original labels and reannotated labels and find a low agreement of 40% accuracy (exact label match) and 50.4 Spearman's Correlation.

We characterize the common issues that cause the annotation divergence in Table 1. The similarity of the sentences under *number* conditions cannot be mapped to a 5-point scale due to the binarity of the value comparison (e.g., $1 = 1$, $1 \neq 5$). This issue also happens to other condition features such as *gender* and *age*. The condition can also be ambiguous, especially when it is a singleton. In the second example, the similarity between the two mentions of the *table* can be subjective, based either on the *type of table* or multiple features associated with the table such as *shape*, *size*, etc. The original C-STS task does not specify how much inference from the context is allowed to judge the similarity. This increases the label inconsistency between the annotators. In the third example, although the room type is not explicitly mentioned in the first sentence, we can still confidently infer it is *bathroom* because of the mention of *toilet* and *sink* in the context. In the last example, the condition can be invalid if the information regarding the condition cannot be extracted or inferred from the sentence.

Condition-aware Annotation on C-STS We reannotate the C-STS validation set to fix common annotation errors and resolve the aforementioned issues that cause the low agreement score. In addition to the original C-STS guideline, we propose additional annotation rules to instruct annotators to deal with cases when the condition is binary or ambiguous, and etc. In the end, after removing 214 instances with invalid conditions, we create a relabeled C-STS validation set that consists of 2,620 instances. Compared to the original set, the new annotation contains more extreme labels such as 1 and 5, while labels in between such as 3, 4, 5 are less frequent. This is due to the high frequency of the instances with binary condition features in the original dataset. We show the annotation process and rules in Appendix A.2.

4 QA for C-STS

With the consideration on scaling the relabeling task to the full C-STS dataset, we explore effective approaches to identifying potential mislabeled instances automatically. We apply QA as a pre-task for identifying information from the sentences that is relevant to the condition, and leverage LLMs to generate answers from condition-transformed questions. We show that the generated answers correlate better to the reannotated labels, and can be used as a reliable intermediate resource for identifying potential annotation errors from the original C-STS dataset.

4.1 Answer Generation

GPT prompting For each instance, we start by transforming its condition into a question with the format *What is [condition]?*.[3] In order to generate high-quality answers, we conduct the QA task with LLMs under a prompting fashion. Each prompt consists of a brief instruction, the original sentence and the condition-transformed question (Appendix A.3). In the instruction, we ask the model to summarize each answer into a word or phrase to reduce potential noise and hallucinated content (Bouyamourn, 2023). We experiment with GPT-3.5 (Brown et al., 2020) and GPT-4 (OpenAI, 2023) to generate answers. We use the OpenAI API with versions `gpt-3.5-turbo-1106` and `gpt-4-0125-preview`.

Answer quality analysis We evaluate the quality of the answers that are generated from the two models on 200 instances sampled from the C-STS validation set. We ask two annotators to measure

[3] We convert the condition text to lowercase and remove the period at the end of the text.

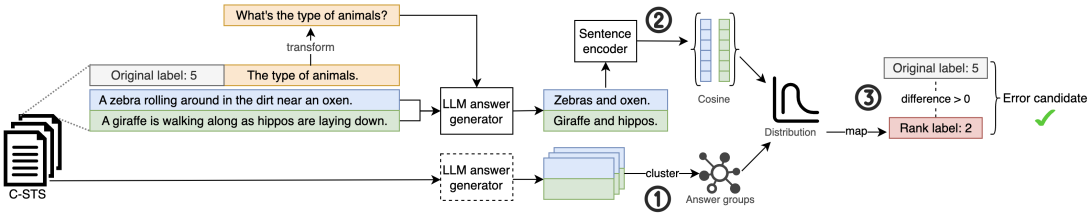

Figure 2: Answer generation and error identification pipeline on the C-STS validation set.

the quality of the answer on a 5-point Likert scale from *unrelated* to *very accurate*. Answers from GPT-3.5 have a Spearman's Correlation of 4.54 and accuracy of 75% (labels with 5), While the Spearman and accuracy of GPT-4 answers are 4.07 and 69.5% respectively. We observe from the data that although GPT-4 is a more recent and proficient model, it tends to have a "laziness" (webtailken, 2024) in making predictions with no clear instructions or contexts.

(1) *A baby is swaddled and someone is putting his hands on it. [the age of child]*
GPT-3.5: **Infant**, GPT-4: **Not specified**

In example 1, the answer from GPT-3.5 is *infant*, which implies a rough range of the age, while GPT-4 refuses to generate an answer because the actual child age is not explicitly mentioned in the text. Since our task involves measuring the text similarity of answers, we choose to use GPT-3.5 because it tends to generate more content-enriched answers than GPT-4.

Answer correlation analysis Given the high quality of the GPT-generated answers, We evaluate the correlation between the answers and the reannotated labels. We first run GPT-3.5 to generate the answers for all the instances from the C-STS validation set. Then we encode each answer into an embedding using the GPT embedding encoder with version `text-embedding-ada-002`, and compute the cosine similarity between the embeddings of the answers from the same instance. We calculate the Spearman's Correlation between the answer similarities and reannotated labels (55.44), and between the original labels and reannotated labels (49.22). The result shows that the cosine similarity of GPT-generated answers correlates more closely with the reannotated labels, and thus can better reflect the similarity of the sentence pairs on the given condition.

4.2 Error Identification

With the analysis that answers generated from GPT-3.5 are of high quality and correlate better with the reannotated labels, we propose a method to automatically identify potential annotation errors from the C-STS dataset. As illustrated in Figure 2, the generated answers are the input to our error identification pipeline that consists of three steps: (1) clustering answers into groups with different topics; (2) encoding and ranking answer pairs in each cluster and mapping the similarity ranks to the label from 1 to 5; (3) identifying error candidates based on the difference between original and new rank labels.

Answer clustering Most of the generated answers can be grouped into different topics (e.g, answers related to *numbers*, *colors*, etc.), we apply K-means (Arthur and Vassilvitskii, 2007) to cluster the answer pairs. Each answer pair is concatenated and then encoded into a single embedding for the clustering. The purpose of this step is to rank the similarity of the answers more accurately by clustering similar answer pairs together.

Answer similarity mapping We encode each answer from a pair into an embedding and compute the cosine similarity between the two embeddings. Within each answer cluster, We rank all the answer pairs based on their embedding similarity and map the ranking of each answer pair to a new label (called *rank label*) on a scale from 1 to 5 based on a predefined ranking distribution.

Error candidate selection we identify the potential error candidates by comparing the original labels to the rank labels. For each answer pair, if its rank label is different from the original label, we choose this instance as a candidate.

4.3 Evaluation

We evaluate our error identification pipeline on the C-STS validation set. We use half of the dataset to generate answer clusters and ranking distributions,

Distribution	No Cluster		Clustered		
	F1	Spear.	F1	Spear.	Best K
Even	74.3	52.9	74.5	60.8	3
Original	75.4	55.4	76.6	59.6	10
Reannotated	79.4	49.1	**82.4**	**65.7**	10

Table 2: Error identification results with or without k-means clustering under different distribution approaches on the test split of the relabeled C-STS validation set. The results under clustered are from the cluster number (Best K) selected in the range of 1-20 that yields the highest F1 score.

and the other half to evaluate three ranking distributions with different number of clusters. For the even distribution, we map the sorted answer pair rankings evenly to rank labels from 1 to 5 (e.g, 20% of lowest ranked instances has rank label 1). For the original or Reannotated label distribution, we map the rankings based on the distribution of original/reannotated labels. We use precision, recall, and F1 as the evaluation metrics.

4.3.1 Results

We show the evaluation results in Table 2. Our pipeline is effective in identifying potential annotation errors with the baseline F1 of 74.3, significantly higher than the 55% error rate in the C-STS validation set. The distribution from the reannotated label performs the best among all three distributions, suggesting the label consistency in the relabeled data. Comparing to no cluster, ranking within the answer clusters improves the most on the reannotated label distribution, boosting F1 from 79.4 to 82.4. As a positive byproduct, clustering can also improve the Spearman's Correlation between the rank labels and reannotated labels, indicating that answers clustered into optimal number of groups can help produce more accurate and correlated rank mapping of the similarity. Overall, we show the effectiveness of our error identification pipeline. It can also be easily extended and applied to identify errors from the rest of the C-STS dataset with the help from the generated answers.

4.3.2 Analysis

We briefly characterize the cases where annotation errors are not identified by the pipeline. The complete examples are shown in Appendix A.5.

Incorrect answer The incorrectness of the generated answers can be caused by: (1) hallucination from the GPT model; (2) vague or invalid condition. For example, in the sentence *A couple of cake ... a white plate.*, the GPT wrongly answers *two pieces of cake* to the condition *the amount of plates*. In another example, the condition *the kite* is not specific enough for both annotators and the GPT model, so the generated answer is highly depended on subjectivity. These mistakes lead to an inaccurate answer similarity and thus cause a misaligned rank label.

Semantic mismatch Error identification relies on the ranking of the answer pair similarity. However, answers even with the opposite meaning can have a high similarity in the embedding space. In an example, the answers are *No meat* and *Meat included*, which are opposite to each other. However, the embeddings of the two answers have a high similarity due to the overlapped token *meat*.

5 Experiments

We propose a new method that can improve the C-STS task by utilizing the LLM-generated answers. Instead of using the sentence pairs and the conditions directly as the model input, our method decomposes C-STS into two subtasks: generating answers that encode the essential semantic information about the condition, and learning the similarity score between the answer pair. We evaluate our method and compare with baseline models (Deshpande et al., 2023) under both fine-tuning and prompting settings. We randomly select 70% instances from the reannotated C-STS validation set for training and the remaining 30% for testing.

5.1 Model Setup

Fine-tuning models We evaluate our method on three encoding configurations, cross-encoder, bi-encoder (Reimers and Gurevych, 2019) and tri-encoder (Deshpande et al., 2023). Unlike the baselines that encode sentences directly, our method chooses to encode the *answers* on all three encoding configurations. Cross-encoder encodes the concatenation of the answer pair and condition all together, while bi-encoder concatenates the condition to each answer and encodes them separately. Tri-encoder has three encoders that encode sentences and the condition all separately. We use supervised SimCSE (Gao et al., 2021), one of the best-performing embedding models as the base sentence encoder for all three encoding configurations. We fine-tune each model on the training set and evaluate on the testing set. We use Spearman's

Model	Method	Spearman	Pearson
Bi-encoder	$\text{SimCSE}_{\text{BASE}}$	49.6	48.5
	$\text{SimCSE}_{\text{LARGE}}$	71.7	70.8
	$\text{QA-SimCSE}_{\text{BASE}}$	73.9	73.4
	$\text{QA-SimCSE}_{\text{LARGE}}$	**75.9**	**75.4**
Tri-encoder	$\text{SimCSE}_{\text{BASE}}$	0.8	1.7
	$\text{SimCSE}_{\text{LARGE}}$	12.8	13.2
	$\text{QA-SimCSE}_{\text{BASE}}$	**73.9**	**73.4**
	$\text{QA-SimCSE}_{\text{LARGE}}$	73.4	73.1
Cross-encoder	$\text{SimCSE}_{\text{BASE}}$	37.2	38.7
	$\text{SimCSE}_{\text{LARGE}}$	43.0	43.3
	$\text{QA-SimCSE}_{\text{BASE}}$	71.4	71.1
	$\text{QA-SimCSE}_{\text{LARGE}}$	**72.9**	**72.3**
GPT-3.5	Base Prompt	9.1	13.5
	QA Prompt	**66.1**	**64.8**
GPT-4	Base Prompt	64.2	**63.3**
	QA Prompt	**64.4**	60.2

Table 3: Evaluation results on the test split of the C-STS relabeled set. We compare our methods (QA-based) with baselines under different model settings.

Correlation as the primary evaluation metric.

Prompting models We compare our method with LLM baselines under a zero-shot prompt learning setting. We evaluate the results on GPT-3.5 and GPT-4. Similar to the experiments on fine-tuned models, instead of asking about the sentences directly, we formulate the prompt by instructing the model to score the similarity between the answers regarding the condition (Appendix A.3).

5.2 Results

We show the model results in Table 3. For all models, our method (QA-based) improves the baselines by a large margin (73.1 improvement on tri-encoder), indicating the usefulness of model learning with answers. Fine-tuned models generally perform better than LLMs models, suggesting that the C-STS task is sensitive to the in-domain training.

Compared with the large version of the models, our method makes more improvement to the base models. For example in the bi-encoder setting, the improvement for large and base model baselines is 4.2 and 24.3 respectively. This indicates the effectiveness of the method, especially on models with smaller sizes and fewer parameters. This may be due to the answers being already encoded with relevant semantic information, thus reducing the reasoning complexity for the small models. This similar pattern also applies to LLM baselines, where GPT-4 performs much better than GPT-3.5 on the base prompt. However on the QA prompt, GPT-3.5 achieves slightly better result than GPT-4. QA transforms C-STS into an easier sentence similarity task that enables the use of more cost or resource efficient models without harming the performance much.

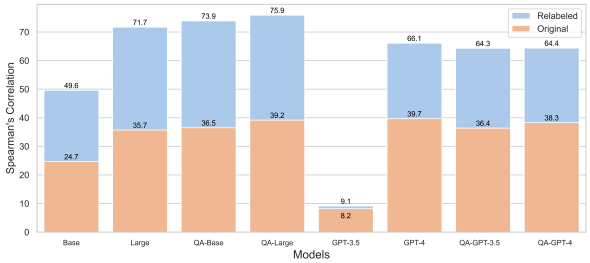

Figure 3: Model (SimCSE with bi-encoder and GPT) evaluation results on original and relabeled C-STS validation set.

5.3 Analysis

Problems with the mislabeled data We evaluate the effect of different label sets on the model performance. In Figure 3, we show the model results on the C-STS validation set with original or reannotated labels. We notice that for all models, performance on the reannotated labels is increased by at least 40% over the original label (except GPT-3.5). This suggests that the original C-STS dataset may not be able to truly reflect the capability of the existing models. The noise and vagueness from the original labels pose "challenges" to models to learn linguistic patterns. Among the LLM baselines, we notice that C-STS is particularly difficult to GPT-3.5 (9.1 Spearman). However, it can be improved significantly with our QA prompt.

Semantic information encoded with QA We evaluate how much semantic information can be encoded in the generated answers. As shown in Table 4, the condition is critical to the finetuned baselines (-0.4 to 49.6 on the $\text{SimCSE}_{\text{BASE}}$). However, training with no condition has minimal effect on the performance with the QA methods (2.4 difference on $\text{SimCSE}_{\text{LARGE}}$). Under the inference setting, the baselines perform poorly due to the additional reasoning complexity from the conditions. However, our method still shows strong performance even without any fine-tuning. This echoes our previous finding that the QA subtask in C-STS is able to improve the efficiency of model training .

Non-GPT answer generation performance We evaluate different answer generation models including multiple versions of Flan-T5 (Chung

Method	Fine-tuning	No condition	Inference
SimCSE$_{\text{BASE}}$	49.6	-0.4	1.2
SimCSE$_{\text{LARGE}}$	71.7	4.4	1.0
QA-SimCSE$_{\text{BASE}}$	73.9	73.3	49.4
QA-SimCSE$_{\text{LARGE}}$	**75.9**	**73.5**	**54.7**

Table 4: Evaluation results (Spearman) from bi-encoder models under different training settings. *Fine-tuning*: using models fine-tuned on the training data; *No condition*: encoding answers only for fine-tuning; *Inference*: inferencing results on the untuned models.

Generation Model	Size (Param.)	Spearman
Flan-T5$_{\text{SMALL}}$	77M	↓43.7
Baseline‡	N/A	49.6
Flan-T5$_{\text{BASE}}$	248M	↑53.9
Flan-T5$_{\text{LARGE}}$	783M	↑55.8
Flan-T5$_{\text{XL}}$	2.75B	↑62.3
GPT-3.5	175B	↑66.1

Table 5: Evaluation results from QA-SimCSE$_{\text{BASE}}$ bi-encoder with difference answer generation models. ‡: Fine-tuning on sentences only.

et al., 2022) and GPT-3.5. We fine-tune the QA-SimCSE$_{\text{BASE}}$ on bi-encoder setting with answers generated from different models, and compare the results in Table 5. Except Flan-T5$_{\text{SMALL}}$, all the other models perform better than the baseline which is fine-tuned on sentences only. The results suggest that our method is robust and does not necessarily rely on proprietary models such as GPT-3.5. Flan-T5$_{\text{XL}}$ is able to achieve comparable results to GPT-3.5 with only 1.5% of the parameter size. Even the base version of Flan-T5 with 248M parameters performs better than the baseline.

6 Improving Conditionality

Although the idea of C-STS is highly appreciated, we notice that the current C-STS dataset has certain issues ranging from the errors in the annotation to the lack of rigor in the condition definition. In this section, we propose a new annotation specification of conditionality based on a word's lexical attribute value features, or Typed Feature Structure (TFS) (Carpenter, 1992; Copestake, 2000), and exemplify the annotation of the new conditionality on a subset of sentence pairs from the C-STS dataset.

6.1 TFS as the Condition

A TFS is a data structure that can be used to represent systematic linguistic information about both words and phrases in language. For lexical items, the feature structure is defined as a set of attributes and their values for a word type. Each feature can have a distinct value for an object of that type (Pustejovsky and Batiukova, 2019). For example, in the term *small table*, the value of the feature SIZE for the object *table* is *small*. In order to adopt the TFS to construct conditions in the C-STS, we use the word/entity type as the condition, and use the values of feature structure to annotate the similarity score regarding the condition. Consider Figure 4 as an example. We set the entity type *Animal* as the condition, and create a feature structure for each of the sentences. The final similarity score is calculated from the weighted sum of the individual similarity label annotated for each non-empty feature. The feature *type* is the primary feature that contributes the most to the final similarity score.

It is worth noting that TFS-based conditionality is highly extendable and customizable depending on the annotation needs. The condition can be selected from any node in a linguistic type hierarchy (e.g., *animal* to *mammal* to *dog* in WordNet (Miller, 1994)). Correspondingly, the feature set can also be modified and the scope for comparing the feature values will be changed. For example, the scope for *type of animal* can be *dog*, *cat* or *capybara*; while the scope for *type of dog* is *poodle*, *bulldog* and etc. Lastly, one can also decide how much weight from each feature needs to be assigned for calculating the final similarity score.

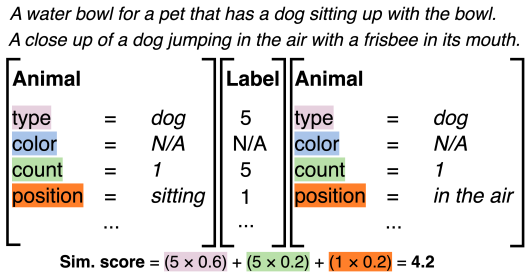

Figure 4: Illustration of the feature structures for the condition *Animal*. Similarity score is calculated from the weighted sum of the similarity label for each feature.

6.2 Pilot Annotation Study on TFS Conditions

We show an annotation study of C-STS sentence pairs with TFS conditions. For the purpose of generalizability, we identify 5 generic entity types from frequent entities in the original conditions: PERSON, ANIMAL, ACTIVITY, LOCATION and FOOD.

Feature structure identification We use existing lexical resources such as Schema.org[4] and Con-

[4] https://schema.org/docs/about.html

ceptNet (Speer et al., 2016), to identify the feature set for each entity type. Schema.org maintains a rich schema that describes various properties of an entity type. We pick the properties that are commonly instantiated in the C-STS data as the features (e.g., we pick *gender* as a feature of PERSON, but exclude *birthPlace*). ConceptNet is a semantic network that represents the meaning of words as knowledge graphs. We identify common features of a word type based on the ConceptNet knowledge triples. For example, we characterize the feature *container* from the *FoodLocatedIn* triples. We define *gender* as the primary feature for `person`, and *type* for the other four entity types. The full table of feature set for each entity type is in Appendix A.7.

TFS Annotation We sample 20 sentence pairs with each entity type in the C-STS validation set. We use the entity type as the condition surface text for each sentence pair (e.g., *the food* from the type FOOD), and its associated feature set as the underlying data structure to evaluate the sentence similarity. We ask annotators to identify all TFSs with distinct primary features regarding the condition. For example, there will be two TFSs for the *cat* and *dog* respectively in the sentence *Two dogs and a cat play together*. The value of a feature is N/A if it cannot be extracted or inferred from the sentence. With the annotated TFSs, we ask annotators to rate the similarity between feature values from different sentences on a 5-point Likert scale, similar to the relabeling process of C-STS in §3.

Weighted similarity score The final similarity score is calculated from the weighted sum of the labels for all the features. If the feature values from both sentences are N/A, this feature will not be accounted for in the final similarity score. If only one of the feature values is N/A, we default the label as 1 to indicate the dissimilarity due to information missing. In practice, we assign 0.6 weight ratio to the primary feature, and the rest 0.4 is assigned evenly to other non-N/A features. This ensures that the similarity score prioritizes the semantic meaning of the entity itself.

Qualitative analysis TFS conditionality enables a finer-grained mapping from entity semantic similarity to the final label, and mitigates the binarity from the evaluation on a single condition feature.

(2) S1: *On the plate there is croissant sandwich ...*
S2: *... next to a hamburger on a green plate.*
Condition: **the food**, Score: **2.9**

Consider example 2. Although the type of the food is different (*croissant sandwich* v.s. *hamburger*), other features have high similarity: both food items are in the *sandwich* category (i.e., category is the supertype of the type of the food) and held in a *plate*; the amount of food are both *one dish*. While most of the entity features have very similar or identical values, the primary feature balances it out and maps the final score to a relatively low similarity.

TFS also improves the clarity and objectivity of the conditions. Even when the entity type is the same, other feature values inferred from the sentence are able to differentiate the entities.

(3) S1: *A little girl is posing with a baseball bat ...*
S2: *A kid is holding a baseball in a glove ...*
Condition: **the activity**, Score: **3.8**

In example 3, both sentences indicate the type of the activity is *baseball*, but participant and instrument are different: the first sentence mentions *a girl with a bat*; while the second one omits the gender of the *kid* and the instrument is *glove*.

Our TFS-based condition sets provide structured lexical information to annotate and interpret sentence semantic similarity. It is also extendable to other data with new types and features with the linguistics base. The annotation of TFSs also provides useful data for other semantic understanding tasks that involve separate or integrated encoding of the contextualized information associated with different entity features.

7 Conclusion

In this paper, we made a comprehensive analysis and improvement to the C-STS task. With the reannotation effort on the original C-STS data, we identified and resolved annotation errors and discrepancies that could hinder the evaluation of the task. We showed that C-STS can be naturally treated as a two-step reasoning task. We applied QA to accomplish the first reasoning step by producing high-quality and correlated answers, and showed that the generated answers can be used effectively to automatically identify annotation errors and improve the C-STS task under both supervised and prompting model settings. Finally, we proposed to use the typed-feature structure in C-STS to construct more semantically informed conditions. We believe that our work has led to a better execution of the C-STS task. We hope that our analysis and improvement on the C-STS can facilitate further developments by future researchers.

8 Limitation

We reannotate the C-STS validation set and show that our method can improve the model performance on the test split of the validation set. We primarily use our reannotated labels as the *gold* for the analysis and modeling. Although we have four annotators who are well trained for this work, there might still exist unintentional errors or subjective judgment.

Due to the limitations of the resources, we could not scale the manual annotation to the full dataset by the time the paper is written. Finally we decide to use the validation set because the original labels for the test set are not public. However, our dataset analysis (§3) on the 150 instances from the train set does show that the errors and issues do exist in the whole C-STS dataset. We believe that our error identification method can facilitate a more efficient reannotation work, and our modeling results and TFS-based conditionality can be generalized to the rest of C-STS data.

References

Mohamed Abdalla, Krishnapriya Vishnubhotla, and Saif Mohammad. 2023. What makes sentences semantically related? a textual relatedness dataset and empirical study. In *Proceedings of the 17th Conference of the European Chapter of the Association for Computational Linguistics*, pages 782–796, Dubrovnik, Croatia. Association for Computational Linguistics.

Eneko Agirre, Carmen Banea, Claire Cardie, Daniel Cer, Mona Diab, Aitor Gonzalez-Agirre, Weiwei Guo, Iñigo Lopez-Gazpio, Montse Maritxalar, Rada Mihalcea, German Rigau, Larraitz Uria, and Janyce Wiebe. 2015. SemEval-2015 task 2: Semantic textual similarity, English, Spanish and pilot on interpretability. In *Proceedings of the 9th International Workshop on Semantic Evaluation (SemEval 2015)*, pages 252–263, Denver, Colorado. Association for Computational Linguistics.

Eneko Agirre, Carmen Banea, Claire Cardie, Daniel Cer, Mona Diab, Aitor Gonzalez-Agirre, Weiwei Guo, Rada Mihalcea, German Rigau, and Janyce Wiebe. 2014. SemEval-2014 task 10: Multilingual semantic textual similarity. In *Proceedings of the 8th International Workshop on Semantic Evaluation (SemEval 2014)*, pages 81–91, Dublin, Ireland. Association for Computational Linguistics.

Eneko Agirre, Carmen Banea, Daniel Cer, Mona Diab, Aitor Gonzalez-Agirre, Rada Mihalcea, German Rigau, and Janyce Wiebe. 2016. SemEval-2016 task 1: Semantic textual similarity, monolingual and cross-lingual evaluation. In *Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016)*, pages 497–511, San Diego, California. Association for Computational Linguistics.

Eneko Agirre, Daniel Cer, Mona Diab, and Aitor Gonzalez-Agirre. 2012. SemEval-2012 task 6: A pilot on semantic textual similarity. In **SEM 2012: The First Joint Conference on Lexical and Computational Semantics – Volume 1: Proceedings of the main conference and the shared task, and Volume 2: Proceedings of the Sixth International Workshop on Semantic Evaluation (SemEval 2012)*, pages 385–393, Montréal, Canada. Association for Computational Linguistics.

Eneko Agirre, Daniel Cer, Mona Diab, Aitor Gonzalez-Agirre, and Weiwei Guo. 2013. *SEM 2013 shared task: Semantic textual similarity. In *Second Joint Conference on Lexical and Computational Semantics (*SEM), Volume 1: Proceedings of the Main Conference and the Shared Task: Semantic Textual Similarity*, pages 32–43, Atlanta, Georgia, USA. Association for Computational Linguistics.

David Arthur and Sergei Vassilvitskii. 2007. k-means++: the advantages of careful seeding. In *ACM-SIAM Symposium on Discrete Algorithms*.

Adam Bouyamourn. 2023. Why LLMs hallucinate, and how to get (evidential) closure: Perceptual, intensional, and extensional learning for faithful natural language generation. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 3181–3193, Singapore. Association for Computational Linguistics.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel Ziegler, Jeffrey Wu, Clemens Winter, Chris Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language models are few-shot learners. In *Advances in Neural Information Processing Systems*, volume 33, pages 1877–1901. Curran Associates, Inc.

Robert L Carpenter. 1992. The logic of typed feature structures. *(No Title)*.

Daniel Cer, Mona Diab, Eneko Agirre, Iñigo Lopez-Gazpio, and Lucia Specia. 2017. SemEval-2017 task 1: Semantic textual similarity multilingual and crosslingual focused evaluation. In *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*, pages 1–14, Vancouver, Canada. Association for Computational Linguistics.

Hyung Won Chung, Le Hou, S. Longpre, Barret Zoph, Yi Tay, William Fedus, Eric Li, Xuezhi Wang, Mostafa Dehghani, Siddhartha Brahma, Albert Webson, Shixiang Shane Gu, Zhuyun Dai, Mirac Suzgun, Xinyun Chen, Aakanksha Chowdhery, Dasha

Valter, Sharan Narang, Gaurav Mishra, Adams Wei Yu, Vincent Zhao, Yanping Huang, Andrew M. Dai, Hongkun Yu, Slav Petrov, Ed Huai hsin Chi, Jeff Dean, Jacob Devlin, Adam Roberts, Denny Zhou, Quoc V. Le, and Jason Wei. 2022. Scaling instruction-finetuned language models. *ArXiv*, abs/2210.11416.

Alexis Conneau and Douwe Kiela. 2018. SentEval: An evaluation toolkit for universal sentence representations. In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018)*, Miyazaki, Japan. European Language Resources Association (ELRA).

Ann Copestake. 2000. Definitions of typed feature structures. *Natural Language Engineering*, 6(1):109–112.

Ameet Deshpande, Carlos Jimenez, Howard Chen, Vishvak Murahari, Victoria Graf, Tanmay Rajpurohit, Ashwin Kalyan, Danqi Chen, and Karthik Narasimhan. 2023. C-STS: Conditional semantic textual similarity. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 5669–5690, Singapore. Association for Computational Linguistics.

Daniel Deutsch, Tania Bedrax-Weiss, and Dan Roth. 2021. Towards question-answering as an automatic metric for evaluating the content quality of a summary. *Transactions of the Association for Computational Linguistics*, 9:774–789.

Matan Eyal, Tal Baumel, and Michael Elhadad. 2019. Question answering as an automatic evaluation metric for news article summarization. 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 3938–3948, Minneapolis, Minnesota. Association for Computational Linguistics.

Tianyu Gao, Xingcheng Yao, and Danqi Chen. 2021. SimCSE: Simple contrastive learning of sentence embeddings. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 6894–6910, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Chulaka Gunasekara, Guy Feigenblat, Benjamin Sznajder, Ranit Aharonov, and Sachindra Joshi. 2021. Using question answering rewards to improve abstractive summarization. In *Findings of the Association for Computational Linguistics: EMNLP 2021*, pages 518–526, Punta Cana, Dominican Republic. Association for Computational Linguistics.

Rensis Likert. 1932. A technique for the measurement of attitude scales.

Dheeraj Mekala, Tu Vu, Timo Schick, and Jingbo Shang. 2022. Leveraging QA datasets to improve generative data augmentation. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 9737–9750, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

George A. Miller. 1994. WordNet: A lexical database for English. In *Human Language Technology: Proceedings of a Workshop held at Plainsboro, New Jersey, March 8-11, 1994*.

OpenAI. 2023. Gpt-4 technical report. *ArXiv*, abs/2303.08774.

Gerald Penn. 2000. *The algebraic structure of attributed type signatures*. Ph.D. thesis, Citeseer.

James Pustejovsky and Olga Batiukova. 2019. *The lexicon*. Cambridge University Press.

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*. Association for Computational Linguistics.

Robyn Speer, Joshua Chin, and Catherine Havasi. 2016. Conceptnet 5.5: An open multilingual graph of general knowledge. In *AAAI Conference on Artificial Intelligence*.

Nandan Thakur, Nils Reimers, Johannes Daxenberger, and Iryna Gurevych. 2021. Augmented SBERT: Data augmentation method for improving bi-encoders for pairwise sentence scoring tasks. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 296–310, Online. Association for Computational Linguistics.

Jingxuan Tu, Kyeongmin Rim, and James Pustejovsky. 2022. Competence-based question generation. In *Proceedings of the 29th International Conference on Computational Linguistics*, pages 1521–1533, Gyeongju, Republic of Korea. International Committee on Computational Linguistics.

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel Bowman. 2018. GLUE: A multi-task benchmark and analysis platform for natural language understanding. In *Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pages 353–355, Brussels, Belgium. Association for Computational Linguistics.

webtailken. 2024. Gpt4-turbo more “stupid/lazy” - it’s not a gpt4.

A Appendix

A.1 Features and Entities Distribution in C-STS dataset

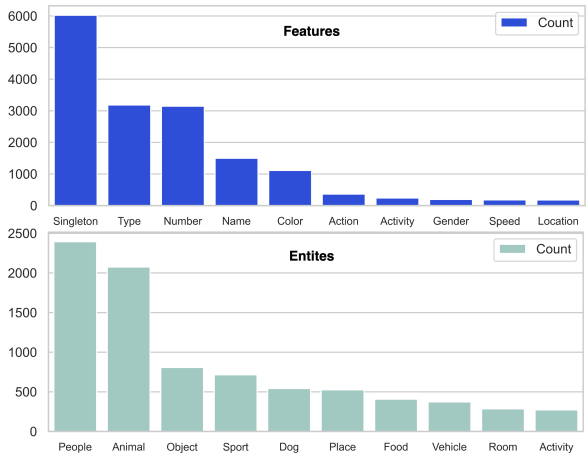

Figure 5: Distribution of top 10 frequent features and entities from the conditions in the C-STS dataset. For the `singleton` with no explicit mention of the feature, we default the condition features from this group to *type*.

A.2 Annotation Process

We reannotate the C-STS validation set to fix common annotation errors and resolve the aforementioned issues that cause the low agreement score. The annotation was done by 4 researchers and graduate students from the linguistics and computer science departments of a US-based university. Each annotator is familiar with the C-STS annotation guideline and is well-trained through the trial annotation on the pilot set with 150 instances. To resolve the issues that are identified from the pilot study, we ask annotators to follow additional annotation rules that are detailed as follows.

Incommensurable mapping For binary conditions, only labels 1, 5, or 3 are permitted, representing equal, unequal and possible equal (e.g., comparing *3* and *several* in the *number* conditions).

Ambiguous condition Given the intuition that *type* is always the primary feature in comparing the similarity between two entities, if conditions are singletons or have no features, we default it to *the type of [entity]*.

Inference degree Annotators are only allowed to make direct inference to the implicitly mentioned information with high confidence. For example, *snow hill* indicates the weather, *tennis* indicates the instruments being used, etc.

Invalid condition We annotate invalid instances with -1 and exclude those instances from the dataset.

After removing 214 instances with invalid conditions, we create a relabeled C-STS validation set that consists of 2,620 samples. Figure 6 shows the label distribution of the original and relabeled C-STS validation set.

Figure 6: The similarity score distribution of the original and relabeled C-STS validation set.

A.3 Prompts

We include various prompts that we used for the experiments in this section. Figure 7 shows the prompt for the answer generation on GPT and Flan-T5 models. Figure 8 and Figure 9 show the prompts for LLM baselines and our method. We instruct the LLMs to score based on answers instead of questions.

```
"Your task is to answer a question based on the information
provided from the given sentence. Your should summarize
your answer into a word or a phrase that is clear and concise.

Sentence: [sentence text]
Question: [question text]

Answer:
```

Figure 7: GPT prompt for the answer generation.

```
On a scale between 1 and 5, how similar are the following
two sentences with respect to the condition provided?
Respond only with a score between 1 and 5.

Input:
Sentence 1: [sentence text]
Sentence 2: [sentence text]
Condition: [condition text]

Output:
```

Figure 8: GPT prompt for the LLM baselines that is adopted from (Deshpande et al., 2023).

A.4 Answer Clustering

We plot the change of the error identification results with different number of clusters in Figure 10. The reannotated distribution consistently performs better than the other two. The optimal cluster number is 10, 10, 3 for reannotation, original, and even distributions respectively.

```
On a scale between 1 and 5, how similar are the following
two answers with respect to the condition provided?
Respond only with a score between 1 and 5.

Input:
Answer 1: [answer text]
Answer 2: [answer text]
Condition: [condition text]

Output:
```

Figure 9: GPT prompt for our QA method on LLM.

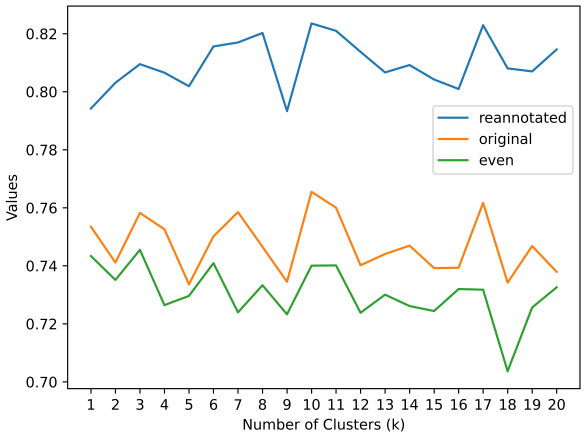

Figure 10: Error identification performance (F1 score) with different number of clusters on three distribution methods.

We show the answer topics identified from the clusters in Table 6. We use the cluster setting that produces the best result when the number of clusters is 10 and summarize the topics from each cluster. Most clusters capture the topics or patterns uniformly from the answers, such as *color* and *food*. A few clusters have a mix of topics.

A.5 Unidentified Annotation Errors

We show the full examples of the cases where the errors cannot be identified by our pipeline in Table 8.

A.6 Model Details

We use OpenAI API to run GPT models. The pricing for the models used in the paper is shown in Table 7. We fine-tune $\text{SimCSE}_{\text{BASE}}$ and $\text{SimCSE}_{\text{LARGE}}$ with different encoding configurations on a single Titan Xp GPU. We use the same hyperparameter setting with the baseline models in (Deshpande et al., 2023). The training time is less than 10 minutes. We use Flan-T5 with different sizes for inference only. We run $\text{Flan-T5}_{\text{SMALL}}$ and $\text{Flan-T5}_{\text{BASE}}$ on CPU machines, and run $\text{Flan-T5}_{\text{LARGE}}$ and $\text{Flan-T5}_{\text{XL}}$ on a Titan Xp GPU.

1	2	3	4	5
food	various topics	number	color	object
6	7	8	9	10
gender, animal	location	activity	room related	activity

Table 6: Topic from answer Clusters.

Model	Input	Output
`gpt-3.5-turbo-0125`	\$0.0005 / 1K tokens	\$0.0015 / 1K tokens
`gpt-4-0125-preview`	\$0.01 / 1K tokens	\$0.03 / 1K tokens
`text-embedding-ada-002`	\$0.00010 / 1K tokens	N/A

Table 7: Pricing of GPT models used in the paper.

A.7 Feature Set of Entity types

We show the identified feature set for each entity type in Table 9.

Sentence Pair	**Condition**	**Response**	**Issue**
A couple of small pieces of cake sitting on top of a white plate. A paper plate with a sandwich and a slice of pizza, both partly eaten.	the amount of plates	Two small pieces of cake One plate	GPT hallucination
A group of people on a hill looking a city and two people are flying a kite. Man playing with kite far up in the sky, showing only the deep blue sky.	the kite	Recreational activity High-flying entertainment	Vague condition
A plate full of cut salad with a fork and a glass full of cold drink with ice. A dish which contains cauliflower and meat is on top of a wooden tray.	presence of meat	No meat Meat included	Semantic mismatch

Table 8: Examples with common issues that cause the unidentified annotation errors.

Entity Type	Features
PERSON	**gender**, count, activity, age, location, position
ANIMAL	**type**, color, size, count, action, location, position
ACTIVITY	**type**, instrument, location, participant
LOCATION	**type**, category, size, purpose
FOOD	**type**, category, amount, container

Table 9: Feature set of the entity types. The primary feature of each entity type is highlighted.