Can Language Models Be Specific? How?

Anonymous ACL submission

Abstract

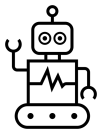
A good speaker needs not only to be correct but also to be specific, and so are language models. In this paper, we propose to measure how specific the language of pre-trained language models (PLMs) is. To achieve this, we introduce a novel approach to build a benchmark for specificity testing by forming masked token prediction tasks with prompts. For instance, given "J. K. Rowling was born in [MASK].", we want to test whether a more specific answer will be better filled in by PLMs, e.g., *Yate* instead of *England*.

From our evaluations, we show that existing PLMs have only a slight preference for more specific answers, indicating that PLMs are weak in specificity. We identify underlying factors affecting the specificity and design two prompt-based methods to improve the specificity. Results show that the specificity of the models can be improved by the proposed methods without additional training. We believe this work can provide a new insight for language modeling and encourage the research community to further explore this important but understudied problem.[1]

1 Introduction

Pre-trained language models (PLMs) such as BERT (Devlin et al., 2019) and GPT-2/3 (Radford et al., 2019; Brown et al., 2020) have achieved quite impressive results in various natural language processing tasks. Recent works show that the parameters of these models contain significant amounts of knowledge (Petroni et al., 2019; Roberts et al., 2020; Jiang et al., 2020a,b; Wang et al., 2020), and knowledge stored in PLMs can be extracted by predicting the mask token(s) using prompts. For instance, given prompt "J. K. Rowling was born in [MASK].", PLMs can predict the birthplace of Rowling based on its knowledge.

[1]We will release the benchmark as open-source after the review process (attached in the supplementary materials).



Cat is a subclass of animal.

Dante is a person.

Toronto is located on the earth.

Figure 1: Examples of language modeling that lack specificity. More specific descriptions could be: feline, poet, and in Ontario, respectively.

However, there may exist multiple answers for a query, while not all answers are equally specific. For the example above, the masked token can be replaced by *Yate* (a town), *Gloucestershire* (a county), or *England* (a country). We prefer the model to fill in *Yate* since *Gloucestershire* and *England* can be further predicted using prompts "Yate is located in [MASK]." and "Gloucestershire is located in [MASK]." respectively. This means, if the prediction is more specific, we can retrieve more fine-grained information from language models, and further acquire more information (in this example, the town, the county, and the country where Rowling was born can be extracted). Besides, sometimes, the less specific answer is not useful. For instance, it is well known that *Chicago* is located in *the USA*, users will not get additional information if the model only predicts *Chicago* is located in *the USA* instead of *Illinois*. More examples are shown in Figure 1. To make an analogy: A good speaker needs not only to be correct but also to be specific. The same is true for language models.

Although there are works on measuring how much knowledge is stored in PLMs or improving the *correctness* of the predictions (Petroni et al., 2019; Roberts et al., 2020; Jiang et al., 2020b), none attempted to measure or improve the *specificity* of predictions made by PLMs. Understanding how specific the language of PLMs is can help us better understand the behavior of language models and facilitate downstream applications such

as question answering, text generation, and information extraction (Liu et al., 2021a; Khashabi et al., 2020; Brown et al., 2020; Wang et al., 2020), e.g., making the generated answers/sentences or extracted information more specific or fine-grained.

Therefore, we propose to build a benchmark to measure the specificity of the language of PLMs. For reducing human effort and easier to further expand the dataset (e.g., to specific domains), we introduce a novel way to construct test data automatically based on transitive relations in Wikidata (Vrandečić and Krötzsch, 2014). Specifically, we extract reasoning paths from Wikidata, e.g., (J. K. Rowling, *birthplace*, Yate, *location*, Gloucestershire, *location*, *England*). Based on the average distance of each object to the subject and the property of transitive relations, we form masked-token-prediction based probing tasks to measure the specificity, e.g., whether the masked token in "J. K. Rowling was born in [MASK]." is better filled by *Yate* than *England* by PLMs. The resulting benchmark dataset contains more than `20,000` probes covering queries from 5 different categories. The quality of the benchmark is high, where the judgment is $\sim 97\%$ consistent with humans.

We provide in-depth analyses on model specificity and study two factors that affect the specificity with our benchmark. As shown by our evaluations in Section 4, existing PLMs, e.g., BERT and GPT-2, similarly have only a slight preference for more specific answers (in only about 60% of cases where a more specific answer is preferred). We also show that, in general, PLMs prefer less specific answers without subjects given, and they only have a weak ability to differentiate coarse-grained/fine-grained objects by measuring their similarities to subjects. The results indicate that specificity was neglected by existing research on language models. How to improve it is undoubtedly an interesting and valuable problem.

Based on our observations and analyses, we propose two techniques to improve the specificity of the predictions by modifying the prompts without additional training: ***Few-shot Prompting***, where demonstrations with more specific answers are provided to guide the models to produce more specific answers; and ***Cascade Prompting***, where *which clauses* are added as suffixes to bias the predictions to be more specific. Results show that Few-shot Prompting can improve the specificity for unidirectional language models like GPT-2 well, while Cascade Prompting works well for bidirectional language models such as BERT.

The main contributions of our work are summarized as follows:

- We report the first attempt to investigate the specificity of the language of pre-trained language models.
- We propose a novel automatic approach to build a benchmark for specificity testing based on the property of transitive relations.
- We show existing PLMs perform poorly on specificity and study two factors that affect the specificity.
- We propose two methods to improve the specificity by modifying the prompts without additional training.
- We provide in-depth analyses and discussions, suggesting further works to explore and further improve the specificity.

2 Background and Related Work

Pre-Trained Language Models: Pre-trained language models (PLMs) are language models pre-trained on large corpora. In this paper, we will cover two types of pre-trained language models: unidirectional language models, such as GPT-2 (Radford et al., 2019), where the prediction of the current token is only based on previous tokens; and bidirectional language models, such as BERT (Devlin et al., 2019) and RoBERTa (Liu et al., 2019), where both left and right contexts are utilized to predict the current token.

Knowledge Retrieval from LMs and Prompting: Previous works have worked on extracting factual knowledge from PLMs without incorporating external knowledge, which is usually achieved by creating prompts and letting PLMs predict the masked token(s) (Petroni et al., 2019; Bouraoui et al., 2020; Jiang et al., 2020a,b; Wang et al., 2020). They demonstrated that PLMs contain a significant amount of knowledge. By creating appropriate prompts with some additional training, such methods can even achieve performance comparable to SOTA for some specific tasks (Shin et al., 2020; Liu et al., 2021b). Our work is inspired by these works; but different from these works, where the focus is to measure or improve the *correctness* of the predictions, our work focuses on measuring and improving the *specificity* of the predictions.

3 S-TEST: Specificity Testing

In this section, we introduce our specificity testing (S-TEST) task, describe the creation process of the dataset, and design the metric to measure the specificity of predictions made by pre-trained language models.

3.1 Task Formulation

Specificity is a semantic feature of language to describe things specifically in a given context. In this work, we focus on measuring the specificity of the predictions produced by pre-trained language models for entity relations. For instance, to extract the answer (object) for relation (Toronto, *location*, **X**), we convert the query to a masked token prediction task using prompts, e.g., "Toronto is located in [MASK]." and let PLMs predict the masked token. The answer here can be a coarse-grained one, e.g., *Canada*, or a fine-grained one, e.g., *Ontario*. The model is considered to perform well in terms of specificity if it tends to fill in *Ontario* instead of *Canada*. More general scenarios are discussed in Section 7 as future work.

3.2 Test Data Construction

We build a benchmark dataset for measuring the specificity based on Wikidata (Vrandečić and Krötzsch, 2014), which is a knowledge base containing a large number of entities and relations. Specifically, we utilize transitive relations[2] in Wikidata to create the test set automatically. Transitive relations are binary relations with properties such that (x, r, y) and (y, r, z) implies (x, r, z), where entity y can be considered as a more fine-grained object of x than entity z under relation r.

For instance, relation *P131* is a transitive relation, whose label is "located in the administrative territorial entity". From Wikidata, we can extract facts (Toronto, *P131*, Ontario) and (Ontario, *P131*, Canada), which furthermore forms a reasoning path (Toronto, *P131*, Ontario, *P131*, Canada). And *Ontario* is considered more fine-grained (specific) than *Canada* in terms of relation *P131* because its distance to *Toronto* is shorter than *Canada* in the reasoning path. Based on this, for a transitive relation, we collect reasoning paths with length ≤ 5 for each subject and calculate the average distance of each object to the subject. In this way, we can construct pairs with coarse-grained/fine-grained objects for each subject, e.g., (Toronto, Ontario) and (Toronto, Canada) for *Toronto* in terms of relation *P131* (or a triplet denoted as (Toronto, Ontario, Canada)). The constructed pairs can be used to test the specificity with prompt: "Toronto is located in [MASK]."

We also combine different relations to form tasks. For instance, for relation *P19*, whose label is "place of birth", we combine it with *P131* and further form a mask token prediction task, such as "[X] was born in [MASK]." An example reasoning path containing coarse-grained/fine-grained objects is (John G. Bennett, *P19*, London, *P131*, England), corresponding to pairs (John G. Bennett, London) and (John G. Bennett, England).

Considering the representativeness and comprehensiveness, we select 5 relations and randomly sample up to 5,000 pairs for each relation, with the difference of average distance of the objects to the subject being greater than or equal to 1 (to filter out entity pairs whose specificity is difficult to differentiate). Similar to (Petroni et al., 2019), we only choose single-token objects as the prediction targets, since multi-token generation is still an area that needs further exploration, and the multi-token decoding process will introduce many tunable parameters that obscure the performance (Welleck et al., 2019; Jiang et al., 2020a). Statistics and examples of the resulting benchmark dataset are shown in Table 1.

3.3 Metric

If a model tends to be more specific, it should have higher confidence that the more specific answer is correct. For instance, given "Toronto is located in [MASK].", the model should assign a higher probability for *Ontario* than *Toronto*. Therefore, we can measure the specificity by calculating how much times the probability of the fine-grained answer is higher than that of the coarse-grained answer, which is

$$p_r = \frac{1}{|\mathcal{T}_r|} \sum_{(x,y_1,y_2)\in\mathcal{T}_r} 1[c(y_1|x,r) > c(y_2|x,r)],$$

where $\mathcal{T}_r$ is the set of test examples for relation r. y_1 is the fine-grained object and y_2 is the coarse-grained object. $c(y|x,r)$ is the probability of the model with y as the prediction of the masked token, and x refers to the subject. p_r ranges from 0 to 1, and 0.5 means the model does not have

[2] https://www.wikidata.org/wiki/Wikidata:List_of_properties/transitive_relation

ID	Relation	Number	Prompt	Answer 1	Answer 2
P19	birthplace	5,000	John G. Bennett was born in [MASK].	London	England
P106	occupation	5,000	Jenny Burton is a [MASK] by profession.	singer	musician
P131	location	5,000	Carey River is located in [MASK].	Victoria	Australia
P279	subclass-of	5,000	Tracking ship is a subclass of [MASK].	vessel	vehicle
P361	part-of	628	Hard palate is part of [MASK].	mouth	head

Table 1: Statistics and examples of the S-TEST benchmark, where we use the same templates in (Petroni et al., 2019) to create prompts. *Answer 1* is more specific than *Answer 2*.

a preference in terms of specificity. The metric is similar to the one used in (Marvin and Linzen, 2018), which compares the probability of a pair of words for creating a grammatical sentence, e..g, *The author laughs* (grammatical) vs *The author laugh* (ungrammatical).

4 Analysis

In this section, we first analyze the results of S-TEST and then identify and study two underlying factors that affect the specificity of predictions produced by pre-trained language models.

4.1 Experimental Setup

We test on the following pre-trained case-sensitive language models: GPT-2, BERT-Base, BERT-Large, RoBERTa-Base, and RoBERTa-Large. For a fair comparison, following (Petroni et al., 2019), we use the intersection of the vocabularies of all the models as the unified vocabulary for prediction (∼18k case-sensitive tokens).

To measure the upper bound of the performance and verify the quality of the dataset, we also estimate human performance on the task. We randomly sampled 400 examples (80 for each relation) and asked human annotators to fill in the masked token with coarse-grained/fine-grained answers given.

4.2 Results of S-TEST

Table 2 reports the results of specificity testing. We observe that existing pre-trained language models perform badly in terms of specificity, where the probability that more specific answers are preferred by them is just around 60%. This is reasonable since the training of PLMs does not introduce any constraint/bias in terms of specificity; therefore, PLMs do not have much perception that more specific answers should be preferred. We also observe that model performance is far from human bound. The high performance achieved by humans also demonstrates that the quality of the dataset is high.

Another interesting finding is that for a single relation, the specificity of different models is highly correlated. For instance, for relation *location*, the specificity measured by p_r of all models is slightly lower than 50%, while for relation *part-of*, the specificity of all models is around 60%. The average pairwise Pearson correlation coefficient among all relations is 0.803. We think this is because pre-trained language models are all trained on large corpora; therefore, their knowledge overlaps to a large extent, as is the preference on the specificity of predictions.

4.3 Factors Affecting Specificity

Some types of questions may be answered specifically naturally. For instance, when discussing anyone's occupation, people may be inclined to use a more specific description; but for the location of a place, people may not be so. In addition, specific answers may be easier to relate to the entities in the query than the coarse-grained ones since their connections may be more close, e.g., $sim(\text{Toronto}, \text{Ontario}) > sim(\text{Toronto}, \text{Canada})$. In this case, the models will tend to be more specific. Based on the above analysis, the specificity of the predictions mainly depends on question types (e.g., relations) and entities in the query (e.g., subjects), which is also indicated by the metric for measuring specificity, i.e., $c(y|x, r)$. To investigate the effect of each component, we split the query, e.g., "Toronto is located in [MASK].", into two parts: the relations, e.g., *is located in*, and the subjects, e.g, *Toronto*, corresponding to *naturalness* and *relatedness* respectively.

Naturalness: For some questions, they may be answered more specifically naturally than others by PLMs. For instance, for questions about the place of birth, if in the corpora, the birthplace is usually

	birthplace	occupation	location	subclass-of	part-of	*Average*
GPT-2	59.72	57.28	48.25	57.98	60.86	56.82
BERT-Base	60.68	70.46	49.09	67.64	67.41	63.06
BERT-Large	56.52	71.76	42.36	77.25	66.77	62.93
RoBERTa-Base	54.48	61.80	49.99	61.59	59.11	57.39
RoBERTa-Large	42.16	71.44	43.28	80.63	59.27	59.36
Human Bound	*98.75*	*92.50*	*100.00*	*96.25*	*97.75*	*97.05*

Table 2: Results of specificity testing with $p_r(\%)$.

		birthplace	occupation	location	subclass-of	part-of	*Average*
GPT-2	Naturalness	46.42	50.86	10.94	60.06	51.12	43.88
	Relatedness	68.51	78.50	82.84	40.00	50.16	64.00
BERT-Base	Naturalness	64.81	75.04	4.99	47.96	50.80	48.72
	Relatedness	74.89	51.96	76.43	71.67	58.79	**66.75**
BERT-Large	Naturalness	66.35	79.22	10.03	48.92	47.60	**50.42**
	Relatedness	54.46	49.16	56.22	72.96	65.50	59.66
RoBERTa-Base	Naturalness	44.80	61.12	23.27	42.06	36.90	41.63
	Relatedness	68.73	58.50	65.73	39.51	56.87	57.87
RoBERTa-Large	Naturalness	31.37	66.24	3.67	43.64	41.69	37.32
	Relatedness	47.82	41.32	34.89	55.17	64.22	48.68

Table 3: Relatedness and naturalness measured with $p_r(\%)$.

described more specifically, e.g., ... *was born in Honolulu, Hawaii*, PLMs will also describe the birthplace more specifically. This is intuitive since PLMs are trained on large corpora based on tasks like masked language modeling; therefore, it will produce more fine-grained predictions conditioned with contexts that are more likely to associate with specific answers.

To measure how natural a type of questions will be answered more specifically by PLMs, we mask the subject in each prompt, e.g., "[MASK] was born in [MASK].", and let PLMs predict the second masked token. We get the probability of each token in the vocabulary, i.e., $c(y|\cdot, r)$, and use our metric and dataset to measure the naturalness, e.g., how natural birthplace will be described more specifically in general.

Relatedness: Considering the following situation: the model knows both *A* and *B* are correct answers, and thinks *A* is more related to the the subject than *B* in general. Intuitively, it will prefer answer *A*.

Therefore, another factor that affects the specificity of predictions made by PLMs is *relatedness*, i.e., to what extent are the fine-grained objects more related to the corresponding subjects than the coarse-grained ones considered by PLMs. (More generally, this is the ability of PLMs to identify more related entities).

We measure relatedness based on phrase embeddings from PLMs. Following (Yu and Ettinger, 2020; Wang et al., 2021), we use the mean-pooled representations over the final-layer outputs from PLMs as phrase embeddings, and calculate the cosine similarities between the subject and the corresponding objects. If the cosine similarity between the subject and the fine-grained object is higher than that between the subject and the coarse-grained object, we think PLMs consider the fine-grained one is more related to the subject. According to this, we can use our metric and dataset to measure the relatedness, with confidence, i.e., $c(y|x, \cdot)$, based on cosine similarity between x and y.

Findings. In Table 3, we report the *naturalness* and *relatedness* with p_r as the metric. We find that, 1) the highest average naturalness and relatedness are achieved by BERT-Large and BERT-Base, respectively, corresponding to the highest average specificity; 2) in many cases, naturalness is lower

than 0.5, which indicates that, without the subjects provided, PLMs are more likely to provide coarse-grained answers; 3) relatedness is usually higher than 0.5, which means PLMs have a certain ability to distinguish fine-grained/coarse-grained answers based on semantic similarities between entities. But the ability is weak since the average scores are just around 60%.

5 Can Language Models Be *MORE* Specific?

From the previous sections, we observe that existing pre-trained language models perform unsatisfactorily in terms of specificity. We also observe that PLMs achieve *naturalness* lower than 0.5, i.e., naturally, PLMs tend to fill in coarse-grained answers with respect to certain types of questions, and *relatedness* around 0.6, i.e., PLMs only have a weak ability to distinguish more related entities. Naturalness depends on both the parameters of PLMs and prompts while relatedness only depends on the parameters of PLMs. Since it is expensive to change the parameters of PLMs (both time and space), to improve the specificity, we focus on improving the naturalness by modifying the prompts.

Intuitively, to get more specific answers, a practical approach is to ask more specific questions. For instance, to know where Toronto is located more specifically, we may change the prompt "Toronto is located in [MASK]." to "Toronto is located in the province of [MASK]." However, to achieve this, humans are required to have additional knowledge, e.g., Toronto is a city, and in Canada, the administrative unit larger than city is province rather than state. Besides, designing such manually crafted prompts can also be time-consuming and laborious if there are a large number of queries. Furthermore, some questions may be difficult to ask more specifically. For instance, for question "Hard palate is part of [MASK].", it is not easy to come up with a more specific query.

Based on the above considerations, we propose two novel and simple techniques to improve the specificity of the predictions. The proposed methods can apply to different models on various types of queries while no additional training is required.

5.1 Few-Shot Prompting

We refer to using prompts in Table 1 to extract answers as ***Vanilla Prompting***. Vanilla Prompting cannot give specific answers since the designed prompts can not tell the models the preference regarding specificity; therefore, the models are not aware of whether a more specific answer is preferred.

Based on the above analysis, we need to give the model some "hints" in terms of specificity, which can be achieved by providing some demonstrations. For instance, to predict where Toronto is located, if we provide some examples with coarse-grained answers using prompt "Melbourne is located in Australia, Guangzhou is located in China, Toronto is located in [MASK].", the model may know by analogy that we prefer a coarse-grained answer, which is Canada (a country). In contrast, if we provide some fine-grained answers using prompt "Melbourne is located in Victoria, Guangzhou is located in Guangdong, Toronto is located in [MASK].", the model may realize through analogy that we prefer a fine-grained answer here, which is Ontario (a province).

We refer to the method described above as ***Few-shot Prompting***, which supposes to bias the prediction to be more specific by providing some examples with fine-grained answers. The technique here is similar to the few-shot setting in GPT-3 (Brown et al., 2020), where several demonstrations of the task are given to the model as condition to help the model make the prediction.

5.2 Cascade Prompting

To make the answer more specific, we can also utilize the relationship between coarse-grained and fine-grained objects. For instance, in Table 1, *tracking ship* is a subclass of *vessel*, while *vessel* is also a subclass of *vehicle*. To combine the three entities, we can write a sentence: *Tracking ship is a subclass of vessel, which is a subclass of vehicle.* By masking the objects, we get prompt: "Tracking ship is a subclass of [MASK], which is a subclass of [MASK]." Intuitively, the first masked token will be more likely to be filled by *vessel*, while the second masked token tends to be *vehicle*. Another example in Table 1 is to predict the birthplace, we can create prompt "John G. Bennett was born in [MASK], which is located in [MASK]." to bias the prediction of the first masked token to be more specific.

We refer to the above method as ***Cascade Prompting***, which aims to improve the specificity by adding "*which clauses*" as constraints according to the relationship between coarse-grained and

Relation	Prompt
birthplace	John G. Bennett was born in [MASK], which is located in [MASK].
occupation	Jenny Burton is a [MASK] by profession, which belongs to [MASK].
location	Carey River is located in [MASK], which is located in [MASK].
subclass-of	Tracking ship is a subclass of [MASK], which is a subclass of [MASK].
part-of	Hard palate is part of [MASK], which is part of [MASK].

Table 4: Example prompts for Cascade Prompting. The prompts are created automatically based on the prompts for corresponding transitive relations.

	birthplace	occupation	location	subclass-of	part-of	*Average*
GPT-2 (VP)	59.72	57.28	48.25	57.98	**60.86**	56.82
GPT-2 (FP)	**81.01**	**71.66**	**50.33**	**64.15**	57.67	**64.96**
GPT-2 (CP)	59.72	57.28	48.25	57.98	**60.86**	56.82
BERT-Base (VP)	60.68	70.46	49.09	67.64	67.41	63.06
BERT-Base (FP)	**67.85**	**70.54**	50.11	**69.11**	53.83	62.29
BERT-Base (CP)	59.68	**70.54**	**55.06**	67.42	**69.49**	**64.44**
BERT-Large (VP)	56.52	**71.76**	42.36	77.25	66.77	62.93
BERT-Large (FP)	66.17	64.70	50.37	65.44	52.24	59.78
BERT-Large (CP)	**82.25**	70.02	**53.55**	**77.67**	**71.88**	**71.07**
RoBERTa-Base (VP)	54.48	61.80	49.99	61.59	59.11	57.39
RoBERTa-Base (FP)	**64.85**	**72.38**	35.85	**63.01**	51.11	57.44
RoBERTa-Base (CP)	63.09	64.54	**54.56**	61.81	**62.78**	**61.36**
RoBERTa-Large (VP)	42.16	71.44	43.28	**80.63**	59.27	59.36
RoBERTa-Large (FP)	70.51	71.94	42.26	73.70	62.94	64.27
RoBERTa-Large (CP)	**89.00**	**74.02**	**66.09**	79.87	**65.18**	**74.83**

Table 5: Results of specificity testing with $p_r(\%)$ with different prompts. The best results in each group are **bold**. VP: Vanilla Prompting, FP: Few-shot Prompting, CP: Cascade Prompting.

fine-grained answers. The "which clauses" here can be considered as suffixes and the prediction of the first masked token is returned as the answer.

Compared to Few-shot Prompting, Cascade Prompting does not need any demonstrations. However, Cascade Prompting cannot be applied to unidirectional language models such as GPT-2/3 (Radford et al., 2019; Brown et al., 2020) since the model can only see the tokens before the masked token and cannot see the suffix. In our experiments in Section 6, we find that Cascade Prompting works well for bidirectional language models while Few-shot Prompting works well for unidirectional language models.

6 Experiments

In this section, we conduct experiments with the prompt-based methods proposed in Section 5.

6.1 Experimental Setup

We follow the setup in Section 4.1. For Few-shot Prompting, we set K, i.e., the number of demonstrations, as 10. For Cascade Prompting, we apply the prompts in Table 4, which are constructed automatically based on the prompts for the transitive relations, e.g., "... is located in [MASK]." ⇒ "..., which is located in [MASK]."

6.2 Results

Table 5 summarizes the results of specificity testing with different prompting methods. From the results, we observe that Cascade Prompting achieves the best performance in most cases. In addition, the performance improvement for BERT-Large and RoBERTa-Large is quite significant. We think this is because the large models can understand which clauses better than the base models.

We also observe that Few-shot Prompting does

	GPT-2	BERT-Base	BERT-Large	RoBERTa-Base	RoBERTa-Large
Naturalness w/ VP	43.88	48.72	50.42	41.63	37.32
Specificity w/ VP	56.82	63.06	62.93	57.39	59.36
Naturalness w/ FP	52.02	51.05	47.36	49.11	49.96
Specificity w/ FP	**64.96**	62.29	59.78	57.44	64.27
Naturalness w/ CP	43.88	51.44	56.54	45.81	57.69
Specificity w/ CP	56.82	**64.44**	**71.07**	**61.36**	**74.83**

Table 6: Average naturalness measured with $p_r(\%)$ with different prompts, with corresponding average specificity as reference. *w/ VP* means vanilla prompting is used to create prompts. For each model, the best naturalness is underlined and the best specificity is **bold**.

not improve the specificity for bidirectional language models. However, for GPT-2, which is a unidirectional language model, Few-shot Prompting achieves a significant performance improvement, while the results of Cascade Prompting are the same as those of Vanilla Prompting.

We also measure *naturalness* of different models with different prompting methods. From Table 6, we find that, for each model, the best prompting method is usually associated with the highest naturalness: Cascade Prompting improves the naturalness for bidirectional language models significantly, which corresponds to better performance on specificity; while for GPT-2, the naturalness using Few-shot Prompting is the highest, corresponding to the highest specificity.

7 Discussion

Specificity Testing in More General Scenarios: In this work, we test the specificity of PLMs on several relations with manually crafted prompts, with test data created automatically based on the property of transitive relations. For future work, we may test the specificity in more general scenarios. For instance, for numerical knowledge (Lin et al., 2020), we can test how specifically PLMs describe the numbers, e.g., *Obama was born in 1961* vs *Obama was born in 1960s*, *A car has four wheels* vs *A car has several wheels*. In addition, we may test on multi-token answers (Jiang et al., 2020a), and measure the specificity of sentences generated by PLMs, e.g., *This is a very good paper. I really like it.* vs *This paper conducts a very novel and interesting study, which provides a new insight for future work on language models.*

Further Improvement of Specificity: In this paper, we propose *Few-shot Prompting* and *Cascade Prompting* to improve the specificity of PLMs without any additional training. Future work may improve the specificity by including prompt-based fine-tuning (Shin et al., 2020; Gao et al., 2021). The weakness of existing pre-trained language models in terms of specificity also encourages future work to take into account the specificity, e.g., adding constraints regarding specificity, in the pre-training process.

8 Conclusion

In this paper, we build a benchmark to measure the specificity of predictions produced by pre-trained language models. To achieve this, we propose a novel approach to construct test data for specificity testing automatically. From our evaluations, we show that exiting pre-trained language models are weak in making the predictions more specific. We also identify and study two underlying factors that affect the specificity and propose two prompt-based methods, i.e., Few-shot Prompting and Cascade Prompting, to improve the specificity of the predictions. Extensive experiments and in-depth analyses demonstrate the effectiveness of the proposed methods. We also suggest some directions for future work in Section 7. We believe this work will encourage the community to explore and further improve the specificity of PLMs – an important property that was understudied by existing research. We also hope this work can give some insight to improve downstream tasks such as question answering, information extraction, and text generation – to make the answers, the extracted information, or the generated sentences more specific.

References

Zied Bouraoui, Jose Camacho-Collados, and Steven Schockaert. 2020. Inducing relational knowledge

from bert. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 7456–7463.

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.

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.

Tianyu Gao, Adam Fisch, and Danqi Chen. 2021. Making pre-trained language models better few-shot learners. In *Association for Computational Linguistics (ACL)*.

Zhengbao Jiang, Antonios Anastasopoulos, Jun Araki, Haibo Ding, and Graham Neubig. 2020a. X-factr: Multilingual factual knowledge retrieval from pretrained language models. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 5943–5959.

Zhengbao Jiang, Frank F Xu, Jun Araki, and Graham Neubig. 2020b. How can we know what language models know? *Transactions of the Association for Computational Linguistics*, 8:423–438.

Daniel Khashabi, Sewon Min, Tushar Khot, Ashish Sabharwal, Oyvind Tafjord, Peter Clark, and Hannaneh Hajishirzi. 2020. Unifiedqa: Crossing format boundaries with a single qa system. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: Findings*, pages 1896–1907.

Bill Yuchen Lin, Seyeon Lee, Rahul Khanna, and Xiang Ren. 2020. Birds have four legs?! numersense: Probing numerical commonsense knowledge of pretrained language models. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 6862–6868.

Pengfei Liu, Weizhe Yuan, Jinlan Fu, Zhengbao Jiang, Hiroaki Hayashi, and Graham Neubig. 2021a. Pre-train, prompt, and predict: A systematic survey of prompting methods in natural language processing. *arXiv preprint arXiv:2107.13586*.

Xiao Liu, Yanan Zheng, Zhengxiao Du, Ming Ding, Yujie Qian, Zhilin Yang, and Jie Tang. 2021b. Gpt understands, too. *arXiv preprint arXiv:2103.10385*.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*.

Rebecca Marvin and Tal Linzen. 2018. Targeted syntactic evaluation of language models. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 1192–1202.

Fabio Petroni, Tim Rocktäschel, Sebastian Riedel, Patrick Lewis, Anton Bakhtin, Yuxiang Wu, and Alexander Miller. 2019. Language models as knowledge bases? In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 2463–2473.

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

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

Taylor Shin, Yasaman Razeghi, Robert L Logan IV, Eric Wallace, and Sameer Singh. 2020. Eliciting knowledge from language models using automatically generated prompts. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 4222–4235.

Denny Vrandečić and Markus Krötzsch. 2014. Wikidata: a free collaborative knowledgebase. *Communications of the ACM*, 57(10):78–85.

Chenguang Wang, Xiao Liu, and Dawn Song. 2020. Language models are open knowledge graphs. *arXiv preprint arXiv:2010.11967*.

Shufan Wang, Laure Thompson, and Mohit Iyyer. 2021. Phrase-bert: Improved phrase embeddings from bert with an application to corpus exploration.

Sean Welleck, Kianté Brantley, Hal Daumé Iii, and Kyunghyun Cho. 2019. Non-monotonic sequential text generation. In *International Conference on Machine Learning*, pages 6716–6726. PMLR.

Lang Yu and Allyson Ettinger. 2020. Assessing phrasal representation and composition in transformers. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 4896–4907.