Manual Prompt Generation For Language Model Probing

Sumit Dalal[1], Abhisek Sharma[1], Sarika Jain[1] and Mayank Dave[1]

[1]National Institute of Technology, Kurukshetra, India 136119.

Abstract

Language models (LMs) have capacity to remind semantic as well as relational information from the training data. For this reason LMs can be employed for knowledge base (KB) construction. Traditional knowledge base (KB) creation process need a schema and human supervision throughout the process. However, this is not required in KB creation from LMs. LM-KBC ISWC 2022 challenge is to use a pre-trained LM to extract object entities given subject entities and relationship. The challenge has three aspects upon which a researcher can focus: i) Different LMs to extract the stored knowledge ii) Threshold value to filter out the obtained results iii) Manual, automatic or semi-automatic approach to generate multiple prompts. From these we have targeted aspect ii and iii. We have followed a manual approach to generate multiple prompts for given relations in the challenge and experimented with different threshold values for selecting an appropriate value for a feasible KB construction system. We have noticed that prompt quality have large impact on the probing performance; while threshold values have somewhat less impact.

1. Introduction

Lot of pre-trained transformers are available in different vocab and dimensionality sizes for distinct tasks. For example, NLP tasks (text classification, question answering), audio classification, and image classification. The transformers or language models (LMs) are trained using large text from books, or the web without any supervision and task specification. Some of the popular LMs are BERT [1], RoBERTa [2], or GPT [3]. Pre-trained LMs ability to remember the semantic and relational information has advanced a range of semantic tasks and introduced positive changes in NLP domain. In recent, authors are being utilizing them for knowledge extraction from the models itself. Using LMs, one can complete a text sequence or masked-out text parts to elicit a relational assertion for a given subject. For example, GPT-3 correctly completes the phrase “Alan Turing was born in” with “London”, which can be seen as yielding a subject-predicate-object triple ⟨ Alan Turing, born in, London ⟩. One can find single token or multi token information from LM and the process is called as LM probing. Single token information can be a subject entity related to an object entity or vice versa and multi token information can be relation between given pair of subject and object entity. The required information is masked and a natural language statement is passed to the model. For previous example, we pass "Alan Turing was born in [MASK]" to the model and receive multiple results with a confidence score. Researchers can write different statements or use different models for

LM-KBC'22: Knowledge Base Construction from Pre-trained Language Models, Challenge at ISWC 2022

sumitdalal9050@gmail.com (S. Dalal)

same query. Also varying threshold on confidence score can be used to find the optimal results for a query. Mainly there are three aspects that can be focused to efficiently extract required information from pre-trained LMs i) Different LMs can be tried to check which model provides better results. ii) One can use different thresholds on confidence score to filter out the retrieved results for a masked token. iii) Multiple prompts can be written to get better results[4].

The information extracted from LMs using the available entities can be used to create triples to form a KB. KB creation in traditional ways is a time consuming and hard working process. Researchers find multiple advantages in using LMs as KBs over traditional KBs. Like there is no need for human supervision or schema in LMs as KBs. Moreover, these are easy to extend and can be queried for open class of relations as no domain-specific dataset is used in training. This lead to the discussion as to what extent LMs could be an alternative to explicit knowledge bases (KBs). Although several works have explored this ability in a setting called LM probing using prompting or prompt-based learning [5], the viability of knowledge base construction from LMs has not yet been explored. Organizers of LM-KBC challenge at ISWC 2022 invited participants to build actual knowledge bases from LMs, for given subjects and relations. However, no simplifying assumptions on relation cardinalities are made in the challenge statement as done in LAMA probing [6]. For example, a subject-entity can stand in relation with zero, one, or many object-entities. The challenge is not just to rank the predictions but to make the concrete decisions on materializing the outputs. The outputs are evaluated using the established F1-score KB metric. The paper is organized into four sections including Introduction section. Second section provide summary regarding different concepts. Third section discusses the proposed approach and in the final section we discuss the results achieved.

2. Background

Pre-trained models in general have advanced various tasks of NLP domain like text classification, machine translation, information retrieval, and question answering. Using LM as KB means to employ a LM as an information source. The knowledge representation is inherently latent, given by the entirety of the neural network's parameter values (in the billions). However, KBs like DBpedia, Freebase and Yago are developed steadily over the years. They store information in triples of subject-predicate-object (SPO) along with qualifiers for non-binary statements. KBs are key sources in various applications, including search engines. Traditional approaches for KB creation have major issues of quality assurance and maintenance as the KB size increases. These approaches require human intervention throughout the KB life-cycle. However, LM based approaches don't need human in the loop approach and hence are more suitable for the KB creation purpose.

Researchers have focused on three aspects (discussed in section 1) of information extraction from a LM to build KB. The LM-KBC challenge offers two tracks where researchers can use i) BERT-base or BERT-large ii) an open track (any of of choice can be employed). Researchers have freedom to use any threshold confidence score to filter out the results and also they can probe various prompts for a sentence. We work on the first track and use BERT-large model as information source. We write multiple prompts for each relation. Prompting or probing is a method, used in literature, to extract information from the pre-trained LMs. A prompt is a string of text used with subject entity in place of KB relation to probe a language model.

3. Proposed Approach

3.1. Dataset

For LM-KBC challenge of ISWC 2022 the dataset covers a diverse set of 12 relations, on these relations different pairs of subject-entities along with list of object-entities (ground truth) pertaining to subject-relation-pair. Each row of the dataset contains a triple of subject-entity, relation, and list of all possible object-entities. Sometimes object-entities have multiple possible values, returning any one of them as output will be considered sufficient for the subject-entity - relation pair.

3.2. Prompt Generation

In literature, researchers follow manual as well as automatic template engineering process to create suitable prompts for LM probing [5]. We consider manual approach to create multiple prompts for each relation given in the challenge dataset. If we consider relations then for every relation at least three and at most fifteen prompts are formed. Table 1 presents the prompts generated by authors in this paper. The table doesn't provide all the prompts for a relation but only two best and two worst performing prompts for each relation. Count of prompts generated for each relation are also mentioned in the table. For a relation, {subject_entity} in prompts is replaced by the subjects provides for that relation in the dataset.

3.3. Number of Outputs per prompt

While probing a LM to find object entity for a subject entity and relation pair, a number of results will be returned with a confidence score. This score represents how much accurately an object entity fits into the mask-token space in prompt. The variable that controls number of results retrieved from LM for a pair is named as top_k. It returns a fixed number of object entities with highest confidence score. We use 100, 150, 180 and 200 values for top_k variable to check if results depend on it.

3.4. Threshold Selection

Varying threshold values (or confidence score) are used in filtering out the results received from LM probing. Seven threshold values are selected randomly to check the precision, recall and f1 score of the predictions. The results corresponding to these threshold values are discussed in the next section.

4. Results & Discussion

It is learnt from previous sections that we have control over three parameters once a language models is decided. These are i) Number of results returned by LM for a prompt (denoted by

Table 1
Best and Worst Performing Manually Generated Prompts For Each Relation.

Relation	Prompts	Total
ChemicalCompound Element	f"{subject_entity} contains atoms of {mask_token}.", f"{subject_entity} made up of {mask_token}.", f"{mask_token} atoms are present in {subject_entity}."",	5
CompanyParent Organization	f"{mask_token} is parent organization of {subject_entity}.", f"{mask_token} owns {subject_entity}.", f"{subject_entity} is a subsidiary of {mask_token}."	5
CountryBorders WithCountry	f"{subject_entity} shares border with {mask_token}.", f"{subject_entity} and {mask_token} are border countries.", f"{mask_token} is neighbour of {subject_entity}.", f"{subject_entity} has neighbouring countries {mask_token}."	13
CountryOfficial Language	f"The official language of {subject_entity} is {mask_token}.", f"{mask_token} is the official language of {subject_entity}.", f"{mask_token} is spoken in {subject_entity}."	8
PersonCause OfDeath	f"{mask_token} took the life of {subject_entity}.", f"{mask_token} led to the death of {subject_entity}.", f"{subject_entity} died due to {mask_token}.", f"{subject_entity} died of {mask_token}."	4
PersonEmployer	f"{subject_entity} is an employee at {mask_token}." ,f"{subject_entity} works at {mask_token}.", f"{subject_entity} is founder of {mask_token}",f"{subject_entity} works in {mask_token}.", f"{subject_entity} is director at {mask_token}"	12
PersonInstrument	f"{subject_entity} plays {mask_token}, which is an instrument", f"{subject_entity} plays {mask_token}.", f"{subject_entity} plays an instrument {mask_token}."	3
PersonLanguage	f"{subject_entity} speaks in {mask_token}.", f"{mask_token} is the native language of {subject_entity}.", f"{subject_entity} speaks {mask_token}."	5
PersonPlace OfDeath	f"{subject_entity} death place {mask_token}.", f"{subject_entity} died at {mask_token}.", f"{subject_entity} died in {mask_token}."	3
PersonProfession	f"{subject_entity} works as an {mask_token}.", f"{subject_entity} is a {mask_token}.", f"{subject_entity} works as {mask_token}.",	6
RiverBasins Country	f"{subject_entity} river basins in {mask_token}.", f"{subject_entity} is longest river of {mask_token}.", f"{subject_entity} flows through {mask_token}.", f"{subject_entity} river flows through {mask_token}."	7
StateShares BorderState	f"{subject_entity} and {mask_token} are bordering states.", f"{subject_entity} and {mask_token} shares border.", f"{mask_token} is neighbour state of {subject_entity}.", f"{subject_entity} neighbouring states {mask_token}."	15

top_k) ii) Number of suitable prompts for a relation. iii) Threshold value to filter results. While probing BERT-large LM models through manually generated prompts we received multiple types of vague values as output like stop words, "unknown", punctuation marks. What a LM will return depends on the prompt passed to it. For example, if "{subject_entity} is an employer at {mask_token}." is passed then it returns values for masked token from the sentences (in LM)

Table 2
Average of Ensembled Output of All Prompts for Each Relation on Various Thresholds and 180 as top_k value

Threshold	Avg. Precision	Avg. Recall	Avg. F1-score
0.20	0.49	0.46	**0.33**
0.25	0.55	0.44	0.32
0.35	0.69	0.38	0.32
0.38	0.72	0.38	0.32
0.42	0.74	0.36	0.30
0.50	0.78	0.33	0.29
0.60	0.83	0.29	0.26

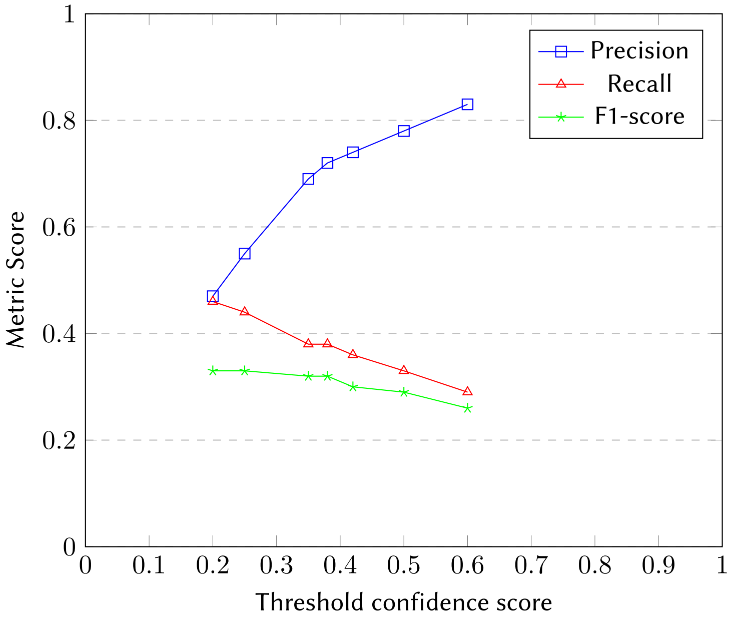

Figure 1: Threshold dependence of Various Metrics

which follows this prompt pattern and end with a name. If "{subject_entity} is an employer at {mask_token}" is passed then it returns values for masked token from the sentences (in LM) which follows this prompt pattern however, it may or may not end with a name.

A number of experiments are performed to check the performance of LM probing. In first experiment, varying threshold values are taken with top_k value 180 and all prompts for every relation. Table 2 displays the average of ensembled result of all prompts for each relation, like for StateSharesBorderState relation the result is accumulation of all fifteen prompts and for CompanyParentOrganization it is collection of all five prompts. Likewise for each relation ensemled result is found. Average of all those ensembled result is calculated and presented in the table. This experiment examines which threshold value is more suitable with all prompts. We observed that as the threshold value increases, precision value increases and recall value

Table 3
Ensembled F1 Score for Maximum Three Prompts, Maximum Five Prompts and All Prompts for Each Relation with 0.20 as Threshold Value and 180 as top_k value

Relation	Max. Three Prompts	Max. Five Prompts	All Prompts
ChemicalCompoundElement	**0.355**	0.333	0.333
CompanyParentOrganization	**0.493**	0.372	0.372
CountryBordersWithCountry	0.448	0.452	**0.496**
CountryOfficialLanguage	**0.653**	0.650	0.637
PersonCauseOfDeath	0.055	0.055	0.055
PersonEmployer	0.017	0.017	**0.026**
PersonInstrument	0.391	0.391	0.391
PersonLanguage	**0.743**	0.612	0.612
PersonPlaceOfDeath	0.352	0.352	0.352
PersonProfession	0.108	0.116	**0.128**
RiverBasinsCountry	0.418	**0.423**	0.377
StateSharesBorderState	0.123	0.144	**0.159**

decreases leading to the fall of f1-score. It is also visible from the figure 1. We get best f1-score of 0.33 with threshold at 0.20. We can also notice that F1-score does not change much if threshold value is below 0.50.

In second experiment, BERT-large is probed differing number of prompts for each relation (like 3,5 prompts for each relation) with threshold value 0.20 and top_k value 180. Table 3 exhibits the ensemled F1 score for maximum three prompts, maximum five prompts and all prompts (total number of prompts are mentioned in table 1 and provided in appendix A) for each relation. Three and five prompts are selected randomly for each relation from total prompts. Table 1 displays two best and one worst performing prompt on the basis of F1-score. Green color marks best and red color is for worst performance. From table 2, we noticed that for some relations F1-score improves as the number of prompts are increased (), for example, CountryBordersWithCountry, PersonEmployer, PersonProfession, and StateSharesBorderState. For some relations the F1-score decreases, like ChemicalCompoundElement, CompanyParentOrganization, CountryOfficialLanguage, and PersonLanguage. There are also cases where increasing number of prompts don't have any impact on F1-score PersonCauseOfDeath, PersonInstrument, PersonPlaceOfDeath. However, in case of RiverBasinsCountry relation F1-score first increases for maximum five prompts and then decreases for all prompts experiment. In summary, we observe that there is not much improvements in the results as we increase number of prompts for each relation. Reason for picking 0.20 as threshold value is the best F1 score at this value in table 2.

In third experiment, multiple values are used for top_k variable (100, 150, 180 and 200), with 0.20, 0.25, 0.35 and 0.38 as threshold value and all prompts for each relation. We observe that there is no difference for different top_k values for a fixed threshold value. For example if threshold is fixed at 0.20 then for all four top_k values average F1-score is static at 0.328 (table 4). Another example is table 5, where threshold is 0.25 and we achieved a F1-score of 0.338 for all values. However, from tables 4, 5, 6 and 7, we can notice that if threshold is changed then

Table 4
Average of Ensembled Output of All Prompts for Each Relation for Distinct top_k values and 0.20 as Threshold Value

Threshold	Avg. Precision	Avg. Recall	Avg. F1-score
100	0.487	0.464	0.328
150	0.487	0.464	0.328
180	0.487	0.464	0.328
200	0.487	0.464	0.328

Table 5
Average of Ensembled Output of All Prompts for Each Relation for Distinct top_k values and 0.25 as Threshold Value

Threshold	Avg. Precision	Avg. Recall	Avg. F1-score
100	0.763	0.381	0.338
150	0.763	0.381	0.338
180	0.763	0.381	0.338
200	0.763	0.381	0.338

Table 6
Average of Ensembled Output of All Prompts for Each Relation for Distinct top_k values and 0.35 as Threshold Value

Threshold	Avg. Precision	Avg. Recall	Avg. F1-score
100	0.692	0.387	0.323
150	0.692	0.387	0.323
180	0.692	0.387	0.323
200	0.692	0.387	0.323

Table 7
Average of Ensembled Output of All Prompts for Each Relation for Distinct top_k values and 0.38 as Threshold Value

Threshold	Avg. Precision	Avg. Recall	Avg. F1-score
100	0.719	0.375	0.318
150	0.719	0.375	0.318
180	0.719	0.375	0.318
200	0.719	0.375	0.318

we got different results. For example, for threshold 0.35 the F1-score is 0.323 and for 0.38 is 0.318. However these changes are not much effective. In future we will try to develop more better prompts via manual or automatic approaches. Also BERT-large can be replaced by other transformers based models like RoBERTa. In this work threshold is selected randomly. Some approach can be followed to find the optimal threshold. Which value, precision or recall, we want to keep better? It depends on the task at hand for which we are probing the LM.

Appendices

Table 8: Prompts for Each Relation

"CountryBorders WithCountry"	f"{subject_entity} shares border with {mask_token}.", f"{subject_entity} has {mask_token} as neighbour.", f"{subject_entity} is a neighbouring country of {mask_token}.", f"{mask_token} is neighbour of {subject_entity}.", f"{subject_entity} and {mask_token} are neighbour countries.", f"{subject_entity} and {mask_token} are border countries.", f"{subject_entity} has neighbouring countries {mask_token}.", f"{subject_entity} is bounded by {mask_token}.", f"{mask_token} is a neighbouring country of {subject_entity}.", f"{subject_entity} borders {mask_token}.", f"{subject_entity} borders {mask_token}, which is a country.", f"{subject_entity} is border country of {mask_token}.", f"{mask_token} is a border country of {subject_entity}."
"CountryOfficial Language"	f"The official language of {subject_entity} is {mask_token}.", f"{subject_entity} speaks {mask_token}.", f"{subject_entity} has official language {mask_token}.", f"{mask_token} is the official language of {subject_entity}.", f"People of {subject_entity} speaks {mask_token}.", f"People speak {mask_token} in {subject_entity}.", f"{mask_token} is the language of {subject_entity}.", f"{mask_token} is spoken in {subject_entity}."
"StateShares BorderState"	f"{subject_entity} shares border with {mask_token}.", f"{subject_entity} has {mask_token} as neighbour.", f"{subject_entity} is a neighbour state of {mask_token}.", f"{mask_token} is neighbour state of {subject_entity}.", f"{subject_entity} and {mask_token} are neighbour states.", f"{subject_entity} and {mask_token} are bordering states.", f"{subject_entity} neighbouring states {mask_token}.", f"{subject_entity} bounded by {mask_token}.", f"{mask_token} is a neighbouring state of {subject_entity}.", f"{subject_entity} borders {mask_token}.", f"{subject_entity} borders {mask_token}, which is a state.", f"{subject_entity} is border state of {mask_token}.", f"{mask_token} is a border state of {subject_entity}.", f"{subject_entity} is neighbouring state of {mask_token}.", f"{subject_entity} and {mask_token} shares border"

"RiverBasinsCountry"	f"{subject_entity} river basins in {mask_token}.", f"{subject_entity} is a river of {mask_token}.", f"{subject_entity} flows through {mask_token}.", f"{subject_entity} is a river in {mask_token}.", f"{subject_entity} is a stream in {mask_token}.", f"{subject_entity} river flows through {mask_token}.", f"{subject_entity} is longest river of {mask_token}."
"Chemical CompoundElement"	f"{subject_entity} consits of {mask_token}, which is an element", f"{subject_entity} contains {mask_token}.", f"{subject_entity} contains atoms of {mask_token}.", f"{mask_token} atoms are present in {subject_entity}.", f"{subject_entity} made up of {mask_token}."
"PersonLanguage"	f"{subject_entity} speaks in {mask_token}.", f"{mask_token} is the native language of {subject_entity}.", f"{subject_entity} can speak {mask_token}.", f"{subject_entity} speaks {mask_token}.", f"{subject_entity} is a proficient speaker of {mask_token}."
"PersonProfession"	f"{subject_entity} is a {mask_token} by profession", f"{subject_entity} is a {mask_token}.", f"{subject_entity} works as an {mask_token}.", f"by profession {subject_entity} is a {mask_token}.", f"{subject_entity} works as {mask_token}.", f"{subject_entity} works as a {mask_token}."
"PersonInstrument"	f"{subject_entity} plays {mask_token}, which is an instrument", f"{subject_entity} plays an instrument {mask_token}.", f"{subject_entity} plays {mask_token}."
"PersonEmployer"	f"{subject_entity} is an employer at {mask_token}, which is a company", f"{subject_entity} is founder of {mask_token}", f"{subject_entity} works at {mask_token}.", f"{subject_entity} works in {mask_token}.", f"{subject_entity} is director at {mask_token}", f"{subject_entity} work in {mask_token}.", f"{subject_entity} is an employee at {mask_token}.", f"{subject_entity} is executive director of {mask_token}", f"{subject_entity} is a recruiter at {mask_token}", f"{subject_entity} is working as a recruiter at {mask_token}", f"{subject_entity} is a manager at {mask_token}", f"{subject_entity} works in {mask_token}, which is a company."

"PersonPlaceOf Death"	f"{subject_entity} died at {mask_token}.", f"{subject_entity} death place {mask_token}.", f"{subject_entity} died in {mask_token}."
"PersonCauseOf Death"	f"{subject_entity} died due to {mask_token}.", f"{subject_entity} died of {mask_token}.", f"{mask_token} took the life of {subject_entity}.", f"{mask_token} led to the death of {subject_entity}."
"CompanyParent Organization"	f"The parent organization of {subject_entity} is {mask_token}.", f"{mask_token} is parent organization of {subject_entity}.", f"{mask_token} owns {subject_entity}.", f"{subject_entity} is a subsidiary of {mask_token}.", f"{subject_entity} is owned by {mask_token}."

References

[1] J. Devlin, M.-W. Chang, K. Lee, K. Toutanova, Bert: Pre-training of deep bidirectional transformers for language understanding, arXiv preprint arXiv:1810.04805 (2018).

[2] Y. Liu, M. Ott, N. Goyal, J. Du, M. Joshi, D. Chen, O. Levy, M. Lewis, L. Zettlemoyer, V. Stoyanov, Roberta: A robustly optimized bert pretraining approach, arXiv preprint arXiv:1907.11692 (2019).

[3] A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, I. Sutskever, et al., Language models are unsupervised multitask learners, OpenAI blog 1 (2019) 9.

[4] Semantic web challenge on knowledge base construction from pre-trained language models (lm-kbc), in: Proc. of the 21st International Semantic Web Conference (ISWC 2022), 2022. URL: https:https://lm-kbc.github.io/.

[5] P. Liu, W. Yuan, J. Fu, Z. Jiang, H. Hayashi, G. Neubig, Pre-train, prompt, and predict: A systematic survey of prompting methods in natural language processing, arXiv preprint arXiv:2107.13586 (2021).

[6] F. Petroni, T. Rocktäschel, P. Lewis, A. Bakhtin, Y. Wu, A. H. Miller, S. Riedel, Language models as knowledge bases?, arXiv preprint arXiv:1909.01066 (2019).