Evaluating Language Models for Common Sense Reasoning on Knowledge Graphs using Perplexity distributions

Federico Reyes Gomez
Stanford University
Department of Computer Science
frg100@stanford.edu

Abstract

As they become more ubiquitous and influential, language models (LMs) need new methods of evaluation to provide a robust view of their biases and performance. We present a new framework for creating common sense reasoning datasets from Knowledge Graphs (KGs) and evaluating language models on these datasets using the distributions of sentence-level perplexity of natural language representations of positive and negative KG triples. Evaluations were performed on four LMs: BERT (Base), RoBERTa (Base), GPT2, and GPT2 (Large) on two KGS: FreeBase15k (FB15K-237) and WordNet18 (WN18RR). To measure whether a LM contains knowledge from a KG, we calculate the p-value from an Independent T-Test on the perplexity distributions of positive and negative samples from the KG. Initial results show high but differing levels of discriminatory ability of pretrained LMs from HuggingFace, with p-values below $p = 0.05$ all the way down to an average $p \approx 1 \times 10^{-75}$.

1 Introduction

Throughout the training process, Language Models consume ever-increasing amounts of unstructured text in order to properly model the distribution of language. Many evaluation metrics directly attempt to measure the quality of the model's ability to generate language that is most likely from the distribution of the language.

While many of these models are being deployed in order to generate language that is natural-sounding, LMs are increasingly being seen as information databases, stored in their parameters, and measured by performance on a variety of Common Sense Reasoning (CSR) datasets. These datasets are important measures of how much knowledge these LMs contain and thus, how suitable they are for knowledge-heavy applications such as question answering. However, these datasets are currently both static and very specialized, leading to overfitting on these tasks and benchmark saturation.

Given this, we sought to develop a framework for creating Common Sense Reasoning datasets from arbitrary Knowledge Graphs, which store information in the form of sets of entities, relations, and triples of (entity A, relation, entity B). Since LMs were meant to be used on natural language, we turn an arbitrary triple (h, r, t) into a natural language sentence with a customizable mapping function.

From a KG, we can sample arbitrary numbers of positive examples, triples that exist in the graph, and negative examples, triples that don't exist in the graph, and measure their perplexity in the case of autoregressive models like GPT2 or pseudo-perplexity (Salazar et al., 2019) in the case of context-based models like BERT and RoBERTa. Given these (pseudo) perplexities, we can then perform an Independent T-Test on the sentence-level perplexity distributions of the positive and negative samples, respectively, and report the resulting p-value. Lower p-values indicate a lower chance that the distributions came from the same source and, by confirming that the mean of the perplexities positive samples is lower than that of negative samples, we can conclude that the model is adept at assigning lower perplexities to real facts than fake facts. We should expect this to be the case given that true facts are more often repeated than fake facts, although the recent misinformation crisis might suggest we re-evaluate this assumption, and we do find this to be the case.

2 Prior Literature

Many ideas in this work are derived from methods developed in existing literature, but could greatly improve the evaluation of Language Models, specifically in the realm of Common Sense Reasoning.

2.1 Knowledge Graphs as information

KGs have previously been used in NLU as sources of structured information, such as in the case of retrofitting word vectors to semantic lexicons, specialized KGs, such as WordNet (Faruqui et al., 2014), in this case to augment a downstream model with information encoded in the KG triples.

Additionally, works such as (Omeliyanenko et al., 2020) have looked at evaluating a single KG triple using a LM, in this case with the introduction of defining mappings with simple, predefined rules that can translate natural language sentences from arbitrary KG triples and evaluating the model perplexity on those sentences. In their case, they then used these perplexities to define edge-weights on the triples. This augmented KG is then used to retrofit word embeddings on the new augmented KG, leading to improved performance on semantic similarity tasks.

Similar methods have previously been used for evaluating LMs for knowledge contained in KGs by (Aspillaga et al., 2021), but this method instead trains a probing classifier on top of the LM and then evaluates performance of the model on a dataset by measuring the tree edit distance from the original KG to a KG reconstructed by the probing classifier. This method is fascinating but it is limited to evaluating a LM on a specific KG with notable hierarchical structure, WordNet, enabling them to use a tree edit distance, and also using an expensive probing classifier method that makes special use of the fact that WordNet is a semantic lexicon.

Another notable mention that utilizes attention scores between the head and tail entities of KG triples to evaluate them is (Wang et al., 2020).

We seek to combine these approaches by first defining customizable mappings from arbitrary KG triples to natural language sentences, allowing for strict deterministic mappings, sets or samples of natural language representations of the same triple, model-based methods such as (Agarwal et al., 2020), and much more. We start out with simple mappings like (Omeliyanenko et al., 2020). Next, we take the approach of (Aspillaga et al., 2021) to use this method to evaluate specific triples, using perplexity for autoregressive models like GPT2 or pseudo-perplexity (Salazar et al., 2019) for context-based models like BERT and RoBERTa, instead of classification accuracy, as our baseline evaluation metric, which gives slightly more granularity, despite suffering from an inability to make direct comparisons between models without using an extra comparison metric.

2.2 Evaluation Metrics on Language Models

Perplexity, the exponentiation of the cross-entropy loss, is the most common used evaluation metric for generative language models. A lower perplexity is better, since it's inversely proportional to the inherent likelihood that the Language Model assigns to a single sample.

The perplexity score (PPL) for an Autoregressive Generative Language Model with parameters Θ on a corpus $\mathbb{W}$ with N tokens as the sum of log probabilities of the reference token given all the preceding tokens in the sentence.

$$PPL(\mathbb{W}) = exp(-\frac{1}{N}\sum_{W \in \mathbb{W}} PPL(W))$$

$$PPL(W) = \sum_{i}^{|W|} \log P_{GLM}(w_i | w_1, \ldots, w_{i-1}; \Theta)$$

where w_i are the tokens in sentence W.

This measure is not well-defined for contextual language models like BERT (Devlin et al., 2018) and RoBERTa (Liu et al., 2019) due to the fact that they don't generate language one token at a time, but rather use the whole context to calculate scores for tokens at each position given the others.

However, we can fortunately define a pseudo-perplexity ($PPPL$) score for a contextual language model with parameters Θ on a corpus $\mathbb{W}$ with N tokens as the sum of log probabilities of the reference token when it's masked out in its sentence.

$$PPPL(\mathbb{W}) = exp(-\frac{1}{N}\sum_{W \in \mathbb{W}} PPPL(W))$$

$$PPPL(W) = \sum_{i}^{|W|} \log P_{MLM}(w_i | W_{\setminus w_i}; \Theta)$$

where w_i are the tokens in sentence W and $P_{MLM}(t|W_{\setminus t}; \Theta)$ measures the probability that the Contextual Masked Language Model (MLM) assigns to token t when it's masked out and the rest of the sentence is left unchanged. This metric was introduced by (Salazar et al., 2019) and was implemented in our library as part of this project.

Given that each language model has its own vocabulary, tokenization scheme, and even measurement of perplexity, it's not fair to directly compare the perplexities of various models, so we have to come up with a new method for determining how powerful each model is compared to others.

The statistical T-Test has been commonly used in Machine Learning for measuring whether model A is statistically significantly better-performing than model B by looking at the p-value comparing the performance of each model on randomly sampled subsets of training and test sets (Kiani, 2019).

We adapt this method by comparing the perplexity values of positive samples to those of negative samples and looking at the resulting p-value to determine whether or not the model assigns statistically significantly lower perplexities to positive examples than negative examples.

3 Knowledge Graphs

While our method is intended to be able to be generally applied to any Knowledge Graph, we chose to do our initial testing with two notable Knowledge Graphs that we have available with entities and relations that we can turn into natural language. For this project we focused on the FreeBase15k (FB15K) (FB1) dataset and the WordNet18 (WN18) (wn1) datasets, both of which contain triples from the Freebase and WordNet Knowledge Graphs, respectively. We also use modified versions of both of these datasets which don't include inverse edges, FB15k-237 and WN18RR.

Dataset	# Relations	# Entities	# Triples
FB15k-237	237	14,505	310,079
WN18RR	11	40,943	93,003

Table 1: Statistics for both datasets from Papers With Code FB15K-237 (FB1), WN18RR (wn1)

Since we need all entities and relations to be in natural language format, we used the preprocessed datasets from (Villmow, 2018) which contain mappings from entity IDs to natural language. We augment this dataset with mappings from relation IDs to natural language templates with spaces for the head and tail entities.

Note that it is trivial to randomly sample anti-triples, incorrect or missing triples, from these knowledge graphs by sampling a head and tail entity as well as a relation either uniformly or based on some prior distribution.

4 Language Models

This project didn't focus on training models, but the choice of initial models to test is important. We started with four models, BERT (Base) (Devlin et al., 2018), RoBERTa (Base) (Liu et al., 2019), GPT2, and GPT2 (Large) (Radford et al., 2019). We wanted to be able to compare the same model with different numbers of parameters (GPT2 vs GPT2 Large), autoregressive models with contextual MLMs (GPT2 vs BERT), as well as newer models with older models of similar structure (RoBERTa vs BERT). All four models are available through the HuggingFace (Wolf et al., 2020) interface, so it will be easy to further extend the analysis to other models that have pre-trained weights on HuggingFace (Wolf et al., 2020).

Model	Year Created	# Parameters
BERT (bert-base-cased)	2018	110M
RoBERTa (roberta-base)	2019	125M
GPT2 (gpt2)	2019	117M
GPT2 (gpt2-large)	2019	774M

Table 2: Statistics for all pretrained models from HuggingFace (Wolf et al., 2020)

5 Methods

5.1 Knowledge Graph Preparation

In order to prepare a Knowledge Graph for use in our analysis, we need to first load the data in its raw format and define the graph structure, which includes all edges as triples of the form (h, r, t) where h is the head entity ID, r is the relation ID, and t is the tail entity ID. Next we need to define a function

$$tripleMap((h,r,t)) \longrightarrow \text{naturalLanguage}_{(h,r,t)}$$

5.1.1 Static Relation Templates

For the purpose of these experiments, we defined an entity mapping that turned entity IDs into their English names or titles, and a relation mapping that provided one static template for each relation that turned a triple into a natural language sentence by substituting the entity markers "{HEAD}" and "{TAIL}" in the template with the corresponding English entity names. For example, the Freebase relation **/film/director/film** has template *"{HEAD} is the director of the film {TAIL}"* and the triple (George Lucas, /film/director/film, Star Wars: Episode IV – A New Hope) becomes the sentence "George Lucas is the director of the film Star Wars: Episode IV – A New Hope".

5.1.2 Grammar Correction

Additionally, since we used static templates, sometimes the output natural language sentence contains grammatical errors with some entities, so we used a T5 Text-to-Text transformer from (Napoles et al., 2017) using the HuggingFace (Wolf et al., 2020) interface to correct such errors.

5.2 Model Preparation

In order to prepare the Language Models for our analysis, we need to load the model parameters from their pretrained weight. For GPT2 and GPT2 Large, we used the HuggingFace `GPT2LMHeadModel` model with the `GPT2TokenizerFast` tokenizer with model ids "gpt2" and gpt2-large". For BERT and RoBERTa we used the HuggingFace `BertForMaskedLM` model and `BertTokenizer` with the "bert-base-cased" model id and HuggingFace `RobertaForMaskedLM` model and `RobertaTokenizer` tokenizers with the "roberta-base" model ids.

5.2.1 (Pseudo) Perplexity Calculations

To calculate perplexity of autoregressive models, we calculate the sum of negative log likelihoods of the target token given its preceding context for all tokens, divide by the number of tokens, and take the exponent of that.

Drawing from (Salazar et al., 2019), to calculate pseudo-perplexity of contextual models, we calculate the sum of the log probabilities of the target token given all of its surrounding context and the target token masked out, divide by the number of tokens, multiply by negative one, and take the exponent of that. (Salazar et al., 2019) suggest that dividing by the number of words instead of the number of tokens gives pseudo-perplexities that are comparable to those of autoregressive models, but using word normalization gives poor empiric performance so we switch back to token normalization.

5.3 Experiments

5.3.1 Sampling from the KG

Given a Knowledge Graph, we can now sample n positive and negative examples each from the knowledge graph and create corresponding natural language sentence representations of them. To sample positive examples, we uniformly sampled the set of triples. To sample negative examples, we uniformly sampled from the set of entities to get the entities and uniformly samples from the set of relations to get the relation. We note that this leads to easily distinguishable negative examples, given that the sentence most times doesn't even make grammatical sense, and describe extensions to this sampling strategy for future work that would make the analysis even more interesting and robust.

5.4 Evaluating the model on the sentences

Next we take the n positive and n negative examples and calculate the model score (perplexity or pseudo-perplexity) on each example, giving us two sets of n numbers.

5.5 Independence T-Test

Next we can apply a standard Independent T-Test, we decided to use `scipy.stats.ttest_ind` with default parameters. This gives us a p-value that calculates the probability that both sets of perplexities came from different distributions. In every case the mean of the perplexities of positive examples was lower than the mean of the perplexities of negative examples but we encourage trying the use of the `less` value for the `alternative` parameter in future work given that we want to test whether the distribution of positive perplexities is significantly lower than that of negative perplexities.

5.6 Repetition

To provide more robust estimates of these p-values, which act as a proxy to the discriminatory power of each model on each dataset, we repeat these experiments r times and show box plots of all r runs for completion.

5.7 Hyperparameters

Due to resource and time constraints, we were only able to run experiments with $r = 25$ and $n = 1000$ for the four models and two graphs.

6 Results

We first report the mean and standard deviation of the p-values for each model and each run:

Model	**Knowledge Graph**			
	FB15k-237		*WN18RR*	
	μ	σ	μ	σ
GPT2	8.37e-08	2.45e-07	3.52e-48	1.57e-47
GPT2-Large	6.12e-07	3.00e-06	**1.10e-59**	*3.05e-59*
BERT	**5.93e-22**	*2.90e-21*	2.42e-22	1.18e-21
RoBERTa	5.98e-04	2.93e-03	1.76e-25	8.60e-25

Table 3: Mean (μ) and Standard Deviation (σ) of Independent T-Test p-values on each Model and Knowledge Graph pairing with $r = 25$ and $n = 1000$

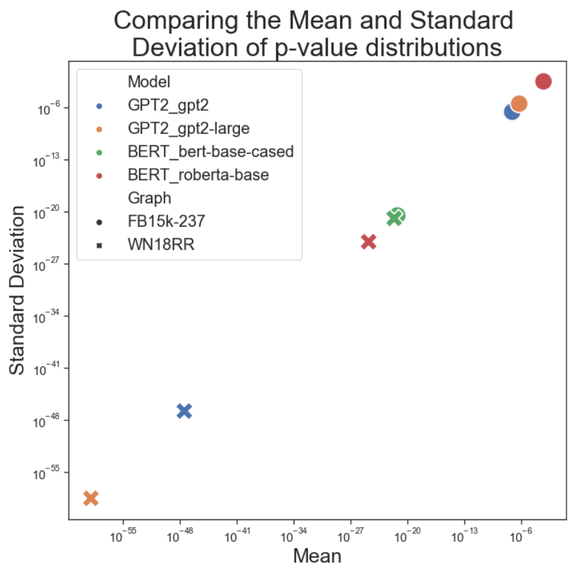

Figure 1: Mean and Standard Deviation comparison across graphs and models

Model	Knowledge Graph			
	FB15k-237		*WN18RR*	
	Median	**Spread**	**Median**	**Spread**
GPT2	2.70e-19	1.13e-06	1.71e-62	8.01e-47
GPT2-Large	1.26e-25	1.53e-05	**1.29e-75**	*1.23e-58*
BERT	**1.90e-30**	*1.48e-20*	4.41e-38	6.04e-21
RoBERTa	1.89e-20	1.50e-02	7.20e-43	4.39e-24

Table 4: Median and Spread (Range) of Independent T-Test p-values on each Model and Knowledge Graph pairing with $r = 25$ and $n = 1000$

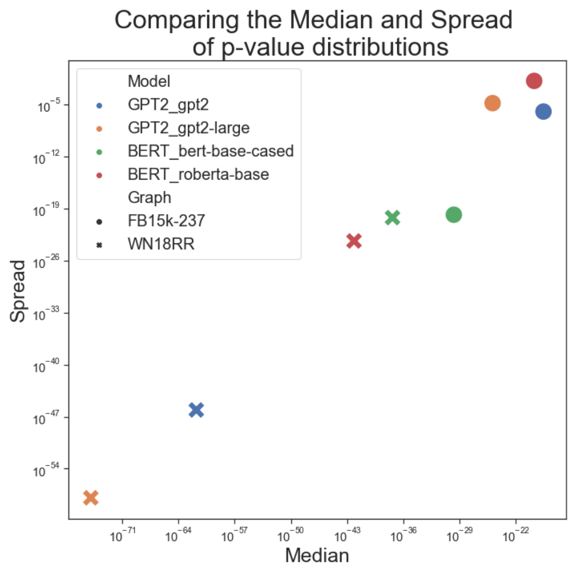

Figure 2: Mean and Spread comparison across graphs and models

There is an interesting log-linear correlation between the mean and standard deviation of the p-values, as well as the median and spread of the p-values, with close correlation between means and medians, as well as spreads and standard deviations across models and graphs.

We can also visualize the individual p-values across all r runs with box plots, one for each dataset.

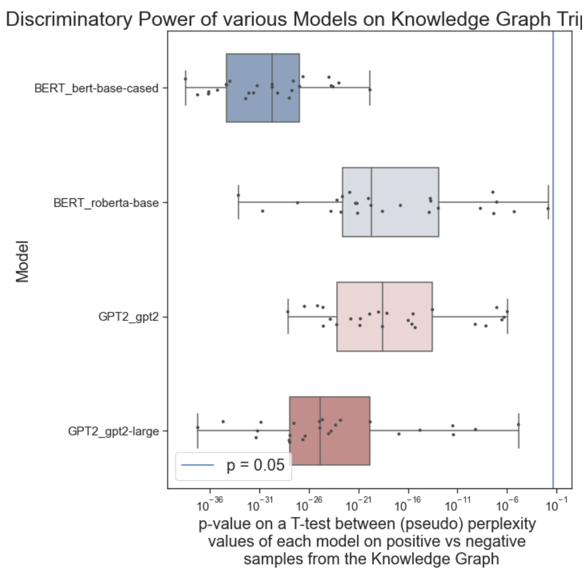

Figure 3: Box Plot of p-values for FB15k-237

For the Freebase dataset, BERT (Base) seems to be the most powerful and consistent, with a lower mean and median than any of the other models. We note that all values are very small so differences here are tiny, but there is weak evidence for BERT's strength with this KG.

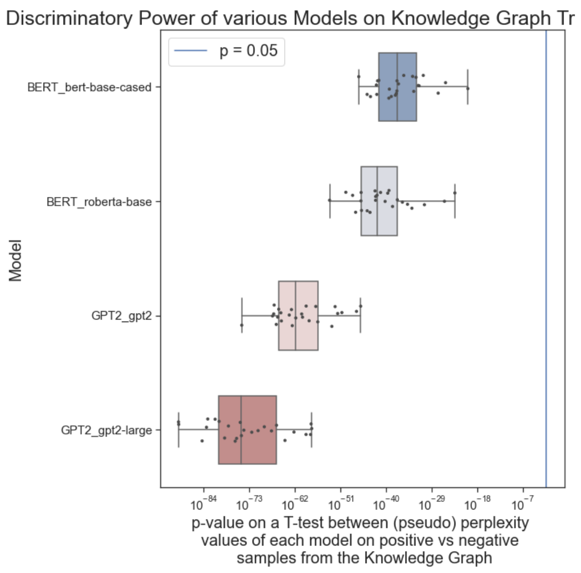

Figure 4: Box Plot of p-values for WN18RR

For the WordNet dataset, we clearly see that GPT2 Large has the best performance, likely due to its larger number of parameters, with the autoregressive GPT2 family of models generally performing better than their contextual BERT counterparts.

Overall, we see that all models lie below the $p = 0.05$ line, signifying that all models have some capacity to distinguish between real and fake facts, given that the models are statistically likelier to assign a lower perplexity value to the real facts.

Below we also plot the means, spreads, standard deviations, and medians as a function of model size

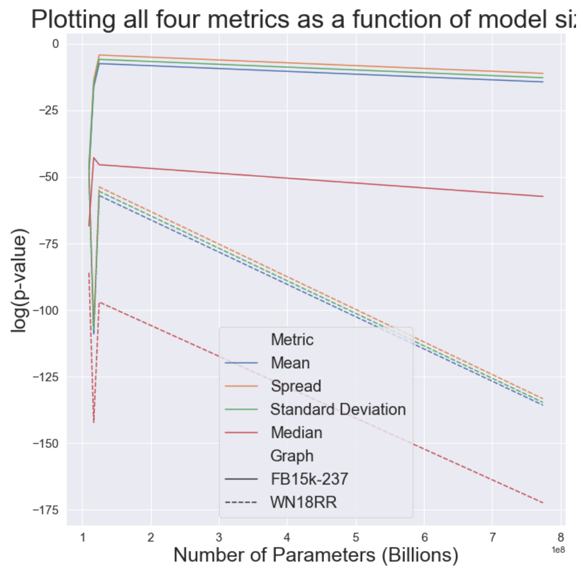

Figure 5: Metrics as a function of model size.
Parameters:

- BERT (110M)
- GPT2 (117M)
- RoBERTa (125M)
- GPT2-Large (774M)

We note that generally, as model size decreases, the p values also decrease, although we see notable outlier dips with BERT on FB15k-237 and GPT2 on WN18RR.

7 Analysis

7.1 Small vs Large Models

In comparing our small model, GPT2, and our large model, GPT2 Large, of the same type, we see that generally more parameters leads to better performance, as expected. It's theorized that more parameters allow LMs to store or memorize more facts and information, which would lead it to have better discriminatory ability between true and false facts.

7.2 Autoregressive vs Contextual Models

This comparison is less clean-cut and as such, we see mixed results. While BERT and GPT2 have similar numbers of parameters, BERT outperforms GPT2 with the FB15k-237 dataset while GPT2 outperforms BERT with the WN18RR dataset. This could be due to the fact that contextual models are better with datasets that contain more complex common-sense facts, such as those contained in the Freebase dataset, because they can read the whole sentence. While GPT2 Large outperforms RoBERTa on FB15k-237, this could be do to the sheer difference in number of parameters. On the other hand, a dataset like WordNet contains simpler phrases that contain fixed, known, and common semantic facts. It's unclear why a model family like GPT2 might perform better in this case, and would be a great area for further work.

7.3 Old vs Improved Models

The final dimension that we wanted to evaluate was whether a model that is a specific improvement of another model would outperform it in every case, but we see that this does not happen. BERT outperforms RoBERTa on the Freebase dataset, and is lagging but not too far behind RoBERTa in the WordNet dataset. More research into this would also be very interesting.

7.4 Overall Impressions

Overall these seem like incredibly promising results, showing significant differences between perplexities of positive and negative examples across the board. Additionally, we see variation across graphs, indicating that some models are better at discriminating some datasets than others. The experiments we were able to run were very limited but show very promising initial results. I'd love to see many more knowledge graphs and models added to this analysis in order to get a more robust image instead of a brief snapshot.

8 Conclusion

This project introduces a new way of evaluating the knowledge contained Language Models using Knowledge Graphs. We first create natural language sentences from positive and negative Knowledge Graph triples using a relation-specific substitution template and load various pretrained Language Models, evaluating the perplexity or pseudo-perplexity of each model on the natural language sentences. Using an independent T-Test, we collected p-values that calculate the probability that the distribution of perplexities on positive samples is lower than that of negative samples and analyzed the p-values collected from repeated runs. We see that all four models are able to discriminate between real facts and fake facts, given that all p-values were below $p = 0.05$, and we additionally see some interesting results, including

The code for this project is publicly available on GitHub @frg100/llmkg and Python pip with `pip install llmkg`. We expect to continue adding more Knowledge Graphs and Models, which we can continue evaluating and adding to the results. If you would like to add a Knowledge Graph, Model, or new analysis, feel free to submit a Pull Request!

Limitations and Future Work

Pseudo-Perplexity

More analysis needs to go into the comparability of perplexity and pseudo-perplexity (Salazar et al., 2019). We also ran the analysis BERT with pseudo-perplexity normalized by the number of words instead of the number of tokens in each sentence. This leads to poor performance with an average below $p = 0.05$ but many values from individual runs appearing above the image and encourage future work on understanding why this is the case.

Alternate Hypothesis

We used a two-sided independent t-test with an alternate hypothesis that both distributions are different, but with no prior knowledge of which one should have a lower mean. Empirically we saw that all models assigned lower perplexity scores to positive samples than negative samples, but in the future, we'd love to see an analysis with a one-sided t-test with the alternate hypothesis that the perplexities of positive examples should be lower than those of negative examples.

Consistency

We also used many runs in order to make the results robust and are confident in our results. However, it would be interesting to have each run use the same positive and negative examples for all models.

Negative Samples

The currently implemented negative sampling strategy is very simple and naive, using a simple uniform distribution on both triples and entities/relations. This, however, leads to very easy negative examples. There are two ways to slowly increase the difficulty of the negative samples

1. **Sample k-of-3**. This method suggests randomly sampling existing positive triples and then resampling k out of the three of head, relation, and tail. The current strategy fits under this framework with $k = 3$. To make the task increasingly harder, we can sample $k = 2$ or $k = 1$, which will provide harder negative samples to the model. With 3-of-3 sampling, most of the triples don't even make grammatical sense, which could lead to models performing better if they have better grammatical sense and not better common sense reasoning.

2. **Category-Aware Sampling**. This method can be combined with the k-of-3 method, with the addition of clustering or categorizing every entity and relation and only sampling replacements within that category. This again, encourages the measurement of common sense reasoning and not just grammatical or distributional knowledge.

Acknowledgements

This project was undertaken by Federico Reyes Gomez with no outside collaboration, although I wish to acknowledge Aasavari Kakne and Christopher Potts for their guidance and advice when undertaking this project for Stanford University's CS224U: Natural Language Understanding.

References

Fb15k-237 dataset. Papers With Code.

Wn18rr dataset. Papers With Code.

Oshin Agarwal, Heming Ge, Siamak Shakeri, and Rami Al-Rfou. 2020. Knowledge graph based synthetic corpus generation for knowledge-enhanced language model pre-training.

Carlos Aspillaga, Marcelo Mendoza, and Alvaro Soto. 2021. Inspecting the concept knowledge graph encoded by modern language models, arxiv.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding.

Manaal Faruqui, Jesse Dodge, Sujay K. Jauhar, Chris Dyer, Eduard Hovy, and Noah A. Smith. 2014. Retrofitting word vectors to semantic lexicons.

Jalal Kiani. 2019. Using the corrected paired student's t-test for comparing machine learning models.

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.

Courtney Napoles, Keisuke Sakaguchi, and Joel Tetreault. 2017. Jfleg: A fluency corpus and benchmark for grammatical error correction.

Janna Omeliyanenko, Albin Zehe, Lena Hettinger, and Andreas Hotho. 2020. Lm4kg: Improving common sense knowledge graphs with language models. In *The Semantic Web – ISWC 2020*, pages 456–473, Cham. Springer International Publishing.

Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners.

Julian Salazar, Davis Liang, Toan Q. Nguyen, and Katrin Kirchhoff. 2019. Pseudolikelihood reranking with masked language models. *CoRR*, abs/1910.14659.

Johannes Villmow. 2018. Datasets for knowledge graph completion with textual information about entities. `https://github.com/villmow/datasets_knowledge_embedding`.

Chenguang Wang, Xiao Liu, and Dawn Song. 2020. Language models are open knowledge graphs, arxiv.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander M. Rush. 2020. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45, Online. Association for Computational Linguistics.