LSLlama: Fine-Tuned LLaMA for Lexical Simplification

Anthony Baez
Massachusetts Institute of Technology
acbaez@mit.edu

Horacio Saggion
Universitat Pompeu Fabra
horacio.saggion@upf.edu

Abstract

Generative Large Language Models (LLMs), such as GPT-3, have become increasingly effective and versatile in natural language processing (NLP) tasks. One such task is Lexical Simplification, where state-of-the-art methods involve complex, multi-step processes which can use both deep learning and non-deep learning processes. LLaMA, an LLM with full research access, holds unique potential for the adaption of the entire LS pipeline. LLaMA was fine-tuned to create LSLlama, which performs comparably to previous LS baseline models LSBert and UniHD.

Introduction

Lexical Simplification

Lexical Simplification (LS) is a sub-task within the field of Text Simplification in which complex words are substituted with simpler words while maintaining the meaning of the surrounding sentence.

Problem

Current state-of-the-art LS models:

- involve a multi-step process to generate and rank substitution candidates.
- improve performance by incorporating deep learning, but can require large amounts of compute

LLaMA

- Generative Large Language Model with full research access
- achieves similar performance to GPT-3 in various NLP tasks (Touvron et al., 2023)
- fine-tuned on 52K question-answer prompts to create Alpaca, a model which was found to often behave similarly to ChatGPT in answering broad sets of questions (Taori et al., 2023)

LSLlama: LLaMA for Lexical Simplification

LLaMA fine-tuned on an LS task:

- simplifies multi-step process into single step, taking advantage of natural language input and output
- reduces significant compute needed to train LLM

LSLlama's performance was evaluated and compared to previously existing benchmark LS models that use deep learning, such as LSBert and UniHD using three testing datasets.

Method

Training

LLaMA was fine-tuned using the TSAR-2022 English gold standard dataset, which contained 373 instances of a sentence, complex word, and substitute candidates. The data was made into a natural language prompt.

Prompt:
Respond with a list of different, simpler synonyms of the complex word in the given context.
Complex Word: ***prototype***
Sentence: *This discovery helped to establish yet another spectral class even cooler than L dwarfs, known as "T dwarfs", for which Gliese 229B is the* ***prototype****.*
Response:

Ranked Candidate List:
['model', 'sample', 'original', 'example', 'template', 'base', 'archetype', 'test', 'first']

Inference

LSLlama was prompted with a similar variation of the above prompt, specifying to respond with a list of 10 synonyms. The repetition penalty generation parameter needed to be tuned so that the model responded with a sufficient number of different candidates.

Evaluation

LSLlama was compared to two benchmark LS models: LSBert (Qiang et al., 2020) and UniHD (Aumiller and Gertz, 2022). The below metrics were used to evaluate the models on three datasets of the same format as TSAR 2022: NNSeval, BenchLS, and LexMTurk.

Metrics

Accuracy@1 *(ACC@1)*: the percent of instances where the top-ranked substitute candidate is in the test dataset candidate list

Accuracy@k@Top1 (*ACC@k@Top1*): the percent of instances where at least one of the k top-ranked substitute candidates matched the top-ranked candidate of the test dataset

Potential@k (*POT@k*): the percent of instances where at least one of the k top-ranked substitute candidates is present in the test dataset candidate list

Mean Average Precision@k (*MAP@k*): a measure that incorporates the percent of k top-ranked substitute candidates that are present in the test dataset candidate list and the relative ranking of the proposed substitute candidate list

Models

LSBert - Benchmark

- Candidate generation: BERT-based model
- Size: 110M parameters
- Ranking method: combined model output, word embeddings, and semantic similarity scores

UniHD - Benchmark

- Candidate generation: 6 runs of GPT-3 with different prompt variations
- Size: 175B parameters
- Ranking method: compiled the results of the runs
- *Highest scoring English language model in TSAR-2022 Shared Task on Lexical Simplification (Saggion et al., 2022)*

LSLlama

- LLaMA fine-tuned on LS task
- Size: 7B parameters
- Candidate generation/ranking done in single step

Results

Dataset	Model	ACC@1	ACC@1@Top1	ACC@2@Top1	ACC@3@Top1	POT@3	POT@5	POT@10	MAP@3	MAP@5	MAP@10
NNSeval	LSBert	0.4310	0.2469	0.3766	0.4519	0.6946	0.7699	0.8619	0.2894	0.2180	0.1349
	UniHD	**0.5732**	0.2803	**0.3849**	0.4435	**0.7824**	**0.8619**	**0.9163**	**0.3661**	**0.2659**	**0.1629**
	LSLlama	0.4519	**0.3096**	0.3808	**0.4686**	0.7657	0.8536	0.8703	0.3233	0.2513	0.1425
BenchLS	LSBert	0.6631	0.3703	0.5016	0.5748	0.8396	0.8859	0.9225	0.4471	0.3341	0.2042
	UniHD	0.7234	0.3057	0.4564	0.5436	0.8751	0.9214	0.9483	0.4766	0.3552	0.2137
	LSLlama	**0.7820**	**0.4700**	**0.5700**	**0.6460**	**0.9420**	**0.9720**	**0.9760**	**0.5519**	**0.4329**	**0.2453**
LexMTurk	LSBert	0.8300	0.3200	0.4300	0.4920	0.9620	0.9680	0.9900	0.6044	0.4591	0.2865
	UniHD	**0.8480**	0.4060	0.5560	0.6260	**0.9700**	**0.9900**	**1.0000**	**0.6067**	**0.4638**	**0.2893**
	LSLlama	0.8060	**0.4680**	**0.5740**	**0.6440**	**0.9700**	0.9860	0.9880	0.5777	0.4556	0.2620

Discussion

By Metric

Accuracy@1 - UniHD always scored highest on **NNSeval & LexMTurk**, LSLlama always scored highest on **BenchLS**

Accuracy@k@Top1 - LSLlama scored highest 8 out of 9 times **over all datasets**

Potential@k - UniHD always scored highest on **NNSeval & LexMTurk**, LSLlama always scored highest on **BenchLS**

MAP@k - UniHD always scored highest on **NNSeval & LexMTurk**, LSLlama always scored highest on **BenchLS NNSeval & LexMTurk**

Overall

The mixed results indicate that UniHD and LSLlama performed comparably when looking at all datasets. Additionally, for all trials, LSBert was never the top scoring model on any metric.

Errors & Limitations

Variation in Output Length - LSLlama did not respond with a consistent number of substitution candidates.

- generation was done with a single token, no parameter that could directly control the length of the list
- fine-tuning dataset did not have a consistent length of candidate lists

Prompt Stability - the quality of the response list varied dramatically depending on the exact wording of the prompt

Prompt without descriptors:
Respond with a list of synonyms of the complex word in the given context.

Prompt with descriptors:
Respond with a list of different, simpler synonyms of the complex word in the given context.

Adding "different" and "simpler" significantly improved the quality of the candidate lists. Before this, the model was not responding with words that would satisfy the LS task.

Prompt with "replace":
Respond with a list of words to replace the complex word in the given context.

Prompt with "synonyms":
Respond with a list of synonyms of the complex word in the given context.

Intuitively, using the first wording is more accurate as the best substitution candidates are not necessarily exact synonyms of the complex word, and the best candidate to replace a word might be a phrase of two or more words. However, the second wording of the prompt noticeably outperformed the first wording.

Computation - LSLlama took about 30 minutes to fine-tune, needed to fine-tune for every prompt and hyperparameter change

Dataset Size - TSAR contained 373 instances, whereas Alpaca's fine-tuning dataset contained 52K instances

Conclusions

Performance Summary

- LSLlama outperformed UniHD in all metrics on BenchLS
- UniHD outperformed LSLlama on most, but not all, metrics on NNSeval and LexMTurk
- LSBert also never scored the highest on any metric.

Conclusion

- LSLlama was able to simplify the multi-step process of LSBert
  - LSLlama used more compute than LSBert
- LSLlama was overall able to perform comparably to UniHD on evaluation metrics
  - LSLlama used drastically less compute than UniHD
- LLaMA and other LLMs that are fine-tuned on a LS task have the potential to improve upon existing benchmarks in Lexical Simplification

References

1. Jipeng Qiang, Yun Li, Yi Zhu, Yunhao Yuan, and Xindong Wu. 2020. Lexical simplification with pre-trained encoders. In Proceedings of the AAAI Conference on Artificial Intelligence, volume 34, pages 8649–8656.
2. Horacio Saggion, Sanja ˇStajner, Daniel Ferres, Kim Cheng Sheang, Matthew Shardlow, Kai North, and Marcos Zampieri. 2022. Findings of the TSAR-2022 shared task on multilingual lexical simplification. In Proceedings of the Workshop on Text Simplification, Accessibility, and Readability (TSAR-2022), pages 271–283, Abu Dhabi, United Arab Emirates (Virtual). Association for Computational Linguistics.
3. Dennis Aumiller and Michael Gertz. 2022. UniHD at TSAR-2022 shared task: Is compute all we need for lexical simplification? In Proceedings of the Workshop on Text Simplification, Accessibility, and Readability (TSAR-2022), pages 251–258, Abu Dhabi, United Arab Emirates (Virtual). Association for Computational Linguistics.
4. Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothee Lacroix, Baptiste Roziere, Naman Goyal, Eric Hambro, Faisal Azhar, et al. 2023. Llama: Open and efficient foundation language models. arXiv preprint arXiv:2302.13971.
5. Rohan Taori, Ishaan Gulrajani, Tianyi Zhang, Yann Dubois, Xuechen Li, Carlos Guestrin, Percy Liang, and Tatsunori B. Hashimoto. 2023. Stanford alpaca: An instruction-following llama model. https://github.com/tatsu-lab/stanford_alpaca