Prompt Pruning for Large Language Models

Srijan Bansal
Language Technologies Institute
Carnegie Mellon University

Abstract

Recently, there has been a surge in large language models, which have shown to be effective as few-shot learners, but the computational budget puts an upper bound on the number of examples that can be used. This is a big limitation for scaling these models to more than 5-10 examples. Further, applying these models to the online setting continuously requires adapting the prompt to new examples without forgetting the old ones. In this study, we explore prompt pruning strategies, which would allow us to include more examples for few-shot settings and adapt the prompt continuously to more new examples for online learning.

1 Introduction

In recent years, large language models such as GPT (Brown et al., 2020), BLOOM (Workshop et al., 2022), OPT (Zhang et al., 2022), and FLAN-T5 (Chung et al., 2022) have brought a paradigm shift in Natural Language Processing (NLP), achieving state-of-the-art performance on many tasks. These models have shown striking capability in embedding language skills and world knowledge, popularizing the in-context learning paradigm. Under this paradigm, a language model is given a prompt, which typically contains task definition, a few training examples as well as a test instance as input and generates the output for the test instance directly, without any update to its parameters. These models act as a single solution for multiple language understanding tasks, which makes this training paradigm appealing to the NLP community.

However, running these models is computationally expensive and depends on the number of tokens used in the input prompt. For instance, OpenAI's Davinci engine is one of the most powerful models, which costs \$0.02/1K tokens. Thus, the computational budget sets an upper bound on the prompt length, which restricts us to very few examples. Further, applying these models to the online setting continuously requires adapting the prompt to new examples without forgetting the old ones. This puts forward an indubitable need to prune prompts such that only relevant tokens are retained. This would allow us to include more examples for few-shot settings or adapt the prompt continuously to more new examples.

2 Related Work

The performance of large language models on downstream tasks has been shown to depend strongly on the choice of in-context examples. Recently many approaches have tried using language models as the scoring function to retrieve good training examples for in-context learning. (Liu et al., 2021) retrieved the nearest examples for every test instance using an unsupervised sentence. (Das et al., 2021) trained a supervised prompt retriever for knowledge-base question answering. (Shin et al., 2021) used GPT-3 to select examples for the prompt for few-shot semantic parsing while (Rubin et al., 2021) trained a light-weight dense retriever from this purpose. Furthermore, (Lu et al., 2021) demonstrate that the order in which the samples are provided can make the difference between near state-of-the-art and random guess performance. They identify performant prompts based on entropy statistics of the candidate permutations. Prompt pruning can complement these techniques by distilling the prompt consisting of good in-context examples retrieved from these techniques and might alleviate the sensitivity of these models to the prompt order.

3 Methodology

Problem Setup : Given k examples $\{x_i, y_i\}_{i\in[k]}$ with task definition $\mathcal{T}$ and test instance $(\tilde{x}, \tilde{y})$. Assuming that model can take at maximum L_{max} tokens, prompt $\mathcal{P}$ for few-shot setting consists of $[\mathcal{T} + \{x_i, y_i\}_{i\in[m]} + \tilde{x}]$ where m ($< k$) corresponds to the maximum number of examples that can

be used with input length as L_{max}. + denotes concatenation.

Random Pruning : We consider prompt $\mathcal{P}$ as $[\mathcal{T} + \{x_i, y_i\}_{i\in[k]} + \tilde{x}]$ and randomly choose up to L_{max} prompt tokens.

3.1 Gradient-based Pruning (iterative) :

In this setting, we prune the prompt in an iterative fashion. Let $\mathcal{P}_i$ denote prompt at iteration i and $\mathcal{P}_0 = \mathcal{T}$. Let $L' = L_{max} - length(\tilde{x})$.

$$\mathcal{P'}_i = [\mathcal{P}_{i-1} + x_i]$$

In this method, we concatenate the prompt from the last iteration with a new example to get $\mathcal{P'}_i$ and then condition LLM to generate output $\hat{y}_i$. Loss ($\mathcal{L}$) between generated output $\hat{y}_i$ and target y_i is then used to compute gradient (∇L_j) for each prompt token ($j \in [length(\mathcal{P'}_i)]$). We used the encoder's last hidden state $\mathbf{h}_j$ for gradient computation. Inner product $< \nabla L_j, h_j >$ is used as a scoring function for each token. A lower score indicates higher relevance. Thus we choose up to L' tokens with the least scores. This is done iteratively for k examples to get prompt $\mathcal{P}_k$, which is there prepended to the test input $\tilde{x}$ to get the final prompt.

3.2 Gradient-based Pruning (window) :

In the previous setting, the first example is more prone to be pruned compared to the last example. We address this using a sliding window-based method to ensure equal odds of getting pruned for each example. We sample $w + 1$ (out of k) examples and use w^{th} examples to compute gradients for prompt tokens corresponding to w-1 examples. We do this k times so that each example gets its gradient computed w times. The average of these w gradients is used to compute the relevance score of each token. Let $L' = L_{max} - length(\tilde{x})$.

$$\mathcal{P}_i = [\mathcal{T} + \{x_i, y_i\}_{i\in[w-1]} + x_w]$$

LLM is conditioned on this $\mathcal{P}_i$ to generate output $\hat{y}_w$. Loss ($\mathcal{L}$) between generated output $\hat{y}_w$ and target y_w is then used to compute the gradient for each example in $\mathcal{P}_i$. After k-iterations, we get w gradients for each example. Let $\hat{\nabla} L^n_j$ denote average gradient for j^{th} token of n^{th} example. We used the encoder's last hidden state $\mathbf{h}^n_j$ for gradient computation. Inner product $< \hat{\nabla} L^n_j, h^n_j >$ is used as a scoring function for each token. A lower score indicates higher relevance. Thus we chose up to L' tokens with the least scores and prepended to the test input $\tilde{x}$ to get the final prompt. We choose w=2 in our experiments.

4 Datasets

Natural Instructions v2 (Wang et al., 2022) is a dataset consisting of a variety of NLP tasks and instructions that describe them in plain language. Task instructions define how an input text is expected to be mapped to an output text for a given task. Original datasets include 2-4 positive and 2-4 negative examples per task. We modified this dataset to support up to 50 in-context examples and 100 test instances per task. We cover **5** task categories and limit each task category to 10 tasks, each with 10 test instances per task.

Task Category	Task names
Question Answering	task615_moviesqa_answer_generation task165_mcscript_question_answering_commonsense task732_mmmlu_answer_generation_public_relations task1399_obqa_answer_generation task835_mathdataset_answer_generation task696_mmmlu_answer_generation_elementary_mathematics task717_mmmlu_answer_generation_logical_fallacies task690_mmmlu_answer_generation_college_medicine task706_mmmlu_answer_generation_high_school_mathematics task119_semeval_2019_task10_geometric_mathematical_answer_generation
Program Execution	task372_synthetic_palindrome_numbers task622_replace_alphabets_in_a_list_by_their_position_in_english_alphabet task488_extract_all_alphabetical_elements_from_list_in_order task370_synthetic_remove_divisible_by_3 task606_sum_of_all_numbers_in_list_between_positions_i_and_j task162_count_words_starting_with_letter task163_count_words_ending_with_letter task064_all_elements_except_first_i task099_reverse_elements_between_index_i_and_j task243_count_elements_in_set_intersection
Information Extraction	task179_participant_extraction task456_matres_intention_classification task1506_celebrity_minimal_dob_span task1568_propara_classification task292_storycommonsense_character_text_generation task683_online_privacy_policy_text_purpose_answer_generation task1517_limit_classfication task1510_evalution_relation_extraction task180_intervention_extraction task926_coached_conv_pref_word_generation
Text Completion	task138_detoxifying-lms_classification_fluency task1389_hellaswag_completion task964_librispeech_asr_text_auto_completion task299_storycloze_sentence_generation task455_swag_context_generation task297_storycloze_incorrect_end_classification task156_codah_classification_adversarial task139_detoxifying-lms_classification_topicality task296_storycloze_correct_end_classification task453_swag_answer_generation
Question Generation	task1657_gooaq_question_generation task861_prost_mcq_answers_generation task1326_qa_zre_question_generation_from_answer task1594_yahoo_answers_topics_question_generation task1602_webquestion_question_genreation task1665_trainglecopa_question_generation task594_sciq_question_generation task860_prost_mcq_generation task1580_eqasc-perturbed_question_generation task1660_super_glue_question_generation

Table 1: Task Categories with their associated task names

Task Category	Question Answering	Program Execution	Information Extraction	Text Completion	Question Generation
Few-shot	36.89	71.22	85.38	**62.33**	55.93
Random Pruning	35.89	70.07	84.24	61.81	**56.03**
Gradient Pruning (iterative)	34.89	**71.36**	85.41	61.31	55.71
Gradient Pruning (window)	**39.28**	67.33	**85.52**	57.73	54.87

Table 2: RougeL for different prompt pruning strategies with L_{max}=1024 with 12 examples per task.

5 Experimental Results

We compare gradient-based pruning strategies with random pruning and standard few-shot performance in Table 2. We evaluated on five task categories using the Flan-T5 large model. Pruning task definitions lead to degradation in performance. Thus, task definitions are not pruned and kept intact. Gradient-based pruning methods outperform random pruning and few-shot performance for question-answering, program execution, and information extraction, while random pruning works better for question generation. Iterative gradient pruning works better than the window-based variant. However, for question answering and information extraction, window-based pruning outperforms the iterative variant.

Variation with L_{max} : We evaluate pruning strategies for different values of L_{max} in Table 4. We also report the average number of examples that can be fit with maximum prompt length as L_{max}. Performance increases with higher L_{max} or, in other words, more computational budget. Performance degrades for random and iterative gradient pruning when $L_{max} = 1024$.

L_{max}	256	512	1024
examples (avg)	1.48	4.72	8.58
zero-shot	23.71	33.31	33.31
Few-shot	23.91	35.39	36.89
Random Pruning	37.51	38.01	35.89
Gradient Pruning (iterative)	34.91	35.56	34.89
Gradient Pruning (window)	36.43	38.94	39.28

Table 3: RougeL for different prompt pruning strategies on Question Answering category with 12 examples per task.

Variation with in-context examples : We evaluate pruning strategies with a different number of in-context examples. Few-shot performance does not vary much with increasing examples, as it cannot fit more than 8-9 examples with given L_{max}.

examples	8	10	12
few-shot	36.89	37.01	36.89
Random Pruning	36.89	39.01	35.89
Gradient Pruning (iterative)	36.89	36.01	34.89
Gradient Pruning (window)	35.61	38.28	39.28

Table 4: RougeL for different prompt pruning strategies on Question Answering category ($L_{max} = 1024$).

Iterative gradient pruning performance degrades with increasing examples. This can be attributed to the excessive pruning of the first few examples. Window-based pruning performance improves with more examples since the token relevance score based on the average of gradients gets less noisy.

6 Conclusion and Future Work

Prompt pruning methods show promise for few-shot learning and can also be utilized for online learning. We must evaluate these strategies on more task categories and other datasets. Gradient-based pruning methods are marginally better than random pruning. This motivates us to explore other ways of prompt pruning. One alternative is to prune prompt tokens based on confidence intervals for each token. Further, these strategies must be tried with other LLM models like BLOOM and OPT to comprehensively evaluate these approaches. The transferability of pruning strategies from less-expensive LLMs to very expensive LLMs is also a potential research direction.

References

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared 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 M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher 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.

Hyung Won Chung, Le Hou, Shayne Longpre, Barret Zoph, Yi Tay, William Fedus, Yunxuan Li, Xuezhi Wang, Mostafa Dehghani, Siddhartha Brahma, Albert Webson, Shixiang Shane Gu, Zhuyun Dai, Mirac Suzgun, Xinyun Chen, Aakanksha Chowdhery, Alex Castro-Ros, Marie Pellat, Kevin Robinson, Dasha Valter, Sharan Narang, Gaurav Mishra, Adams Yu, Vincent Zhao, Yanping Huang, Andrew Dai, Hongkun Yu, Slav Petrov, Ed H. Chi, Jeff Dean, Jacob Devlin, Adam Roberts, Denny Zhou, Quoc V. Le, and Jason Wei. 2022. Scaling instruction-finetuned language models.

Rajarshi Das, Manzil Zaheer, Dung Thai, Ameya Godbole, Ethan Perez, Jay Yoon Lee, Lizhen Tan, Lazaros Polymenakos, and Andrew McCallum. 2021. Case-based reasoning for natural language queries over knowledge bases. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 9594–9611, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Jiachang Liu, Dinghan Shen, Yizhe Zhang, Bill Dolan, Lawrence Carin, and Weizhu Chen. 2021. What makes good in-context examples for gpt-3?

Yao Lu, Max Bartolo, Alastair Moore, Sebastian Riedel, and Pontus Stenetorp. 2021. Fantastically ordered prompts and where to find them: Overcoming few-shot prompt order sensitivity.

Ohad Rubin, Jonathan Herzig, and Jonathan Berant. 2021. Learning to retrieve prompts for in-context learning.

Richard Shin, Christopher Lin, Sam Thomson, Charles Chen, Subhro Roy, Emmanouil Antonios Platanios, Adam Pauls, Dan Klein, Jason Eisner, and Benjamin Van Durme. 2021. Constrained language models yield few-shot semantic parsers. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 7699–7715, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Yizhong Wang, Swaroop Mishra, Pegah Alipoormolabashi, Yeganeh Kordi, Amirreza Mirzaei, Anjana Arunkumar, Arjun Ashok, Arut Selvan Dhanasekaran, Atharva Naik, David Stap, Eshaan Pathak, Giannis Karamanolakis, Haizhi Gary Lai, Ishan Purohit, Ishani Mondal, Jacob Anderson, Kirby Kuznia, Krima Doshi, Maitreya Patel, Kuntal Kumar Pal, Mehrad Moradshahi, Mihir Parmar, Mirali Purohit, Neeraj Varshney, Phani Rohitha Kaza, Pulkit Verma, Ravsehaj Singh Puri, Rushang Karia, Shailaja Keyur Sampat, Savan Doshi, Siddhartha Mishra, Sujan Reddy, Sumanta Patro, Tanay Dixit, Xudong Shen, Chitta Baral, Yejin Choi, Noah A. Smith, Hannaneh Hajishirzi, and Daniel Khashabi. 2022. Super-naturalinstructions: Generalization via declarative instructions on 1600+ nlp tasks.

BigScience Workshop, :, Teven Le Scao, Angela Fan, Christopher Akiki, Ellie Pavlick, Suzana Ilić, Daniel Hesslow, Roman Castagné, Alexandra Sasha Luccioni, François Yvon, Matthias Gallé, Jonathan Tow, Alexander M. Rush, Stella Biderman, Albert Webson, Pawan Sasanka Ammanamanchi, Thomas Wang, Benoît Sagot, Niklas Muennighoff, Albert Villanova del Moral, Olatunji Ruwase, Rachel Bawden, Stas Bekman, Angelina McMillan-Major, Iz Beltagy, Huu Nguyen, Lucile Saulnier, Samson Tan, Pedro Ortiz Suarez, Victor Sanh, Hugo Laurençon, Yacine Jernite, Julien Launay, Margaret Mitchell, Colin Raffel, Aaron Gokaslan, Adi Simhi, Aitor Soroa, Alham Fikri Aji, Amit Alfassy, Anna Rogers, Ariel Kreisberg Nitzav, Canwen Xu, Chenghao Mou, Chris Emezue, Christopher Klamm, Colin Leong, Daniel van Strien, David Ifeoluwa Adelani, Dragomir Radev, Eduardo González Ponferrada, Efrat Levkovizh, Ethan Kim, Eyal Bar Natan, Francesco De Toni, Gérard Dupont, Germán Kruszewski, Giada Pistilli, Hady Elsahar, Hamza Benyamina, Hieu Tran, Ian Yu, Idris Abdulmumin, Isaac Johnson, Itziar Gonzalez-Dios, Javier de la Rosa, Jenny Chim, Jesse Dodge, Jian Zhu, Jonathan Chang, Jörg Frohberg, Joseph Tobing, Joydeep Bhattacharjee, Khalid Almubarak, Kimbo Chen, Kyle Lo, Leandro Von Werra, Leon Weber, Long Phan, Loubna Ben allal, Ludovic Tanguy, Manan Dey, Manuel Romero Muñoz, Maraim Masoud, María Grandury, Mario Šaško, Max Huang, Maximin Coavoux, Mayank Singh, Mike Tian-Jian Jiang, Minh Chien Vu, Mohammad A. Jauhar, Mustafa Ghaleb, Nishant Subramani, Nora Kassner, Nurulaqilla Khamis, Olivier Nguyen, Omar Espejel, Ona de Gibert, Paulo Villegas, Peter Henderson, Pierre Colombo, Priscilla Amuok, Quentin Lhoest, Rheza Harliman, Rishi Bommasani, Roberto Luis López, Rui Ribeiro, Salomey Osei, Sampo Pyysalo, Sebastian Nagel, Shamik Bose, Shamsuddeen Hassan Muhammad, Shanya Sharma, Shayne Longpre, Somaieh Nikpoor, Stanislav Silberberg, Suhas Pai, Sydney Zink, Tiago Timponi Torrent, Timo Schick, Tristan Thrush, Valentin Danchev, Vassilina Nikoulina, Veronika Laippala, Violette Lepercq, Vrinda Prabhu, Zaid Alyafeai, Zeerak Talat, Arun Raja, Benjamin Heinzerling, Chenglei Si, Davut Emre Taşar, Elizabeth Salesky, Sabrina J. Mielke, Wilson Y. Lee, Abheesht Sharma, Andrea Santilli, Antoine Chaffin, Arnaud Stiegler, Debajyoti Datta, Eliza Szczechla, Gunjan Chhablani, Han Wang, Harshit Pandey, Hendrik Strobelt, Jason Alan Fries, Jos Rozen, Leo Gao, Lintang Sutawika, M Saiful Bari, Maged S. Al-shaibani, Matteo Manica, Nihal Nayak, Ryan Teehan, Samuel Albanie, Sheng Shen, Srulik Ben-David, Stephen H. Bach, Taewoon Kim, Tali Bers, Thibault Fevry, Trishala Neeraj, Urmish Thakker, Vikas Raunak, Xiangru Tang, Zheng-Xin Yong, Zhiqing Sun, Shaked Brody, Yallow Uri, Hadar Tojarieh, Adam Roberts, Hyung Won Chung, Jaesung Tae, Jason Phang, Ofir Press, Conglong Li, Deepak Narayanan, Hatim Bourfoune, Jared Casper, Jeff Rasley, Max Ryabinin, Mayank Mishra, Minjia Zhang, Mohammad Shoeybi, Myriam Peyrounette, Nicolas Patry, Nouamane Tazi, Omar Sanseviero, Patrick von Platen, Pierre Cornette, Pierre François Lavallée, Rémi Lacroix, Samyam Rajbhandari, San-

chit Gandhi, Shaden Smith, Stéphane Requena, Suraj Patil, Tim Dettmers, Ahmed Baruwa, Amanpreet Singh, Anastasia Cheveleva, Anne-Laure Ligozat, Arjun Subramonian, Aurélie Névéol, Charles Lovering, Dan Garrette, Deepak Tunuguntla, Ehud Reiter, Ekaterina Taktasheva, Ekaterina Voloshina, Eli Bogdanov, Genta Indra Winata, Hailey Schoelkopf, Jan-Christoph Kalo, Jekaterina Novikova, Jessica Zosa Forde, Jordan Clive, Jungo Kasai, Ken Kawamura, Liam Hazan, Marine Carpuat, Miruna Clinciu, Najoung Kim, Newton Cheng, Oleg Serikov, Omer Antverg, Oskar van der Wal, Rui Zhang, Ruochen Zhang, Sebastian Gehrmann, Shachar Mirkin, Shani Pais, Tatiana Shavrina, Thomas Scialom, Tian Yun, Tomasz Limisiewicz, Verena Rieser, Vitaly Protasov, Vladislav Mikhailov, Yada Pruksachatkun, Yonatan Belinkov, Zachary Bamberger, Zdeněk Kasner, Alice Rueda, Amanda Pestana, Amir Feizpour, Ammar Khan, Amy Faranak, Ana Santos, Anthony Hevia, Antigona Unldreaj, Arash Aghagol, Arezoo Abdollahi, Aycha Tammour, Azadeh HajiHosseini, Bahareh Behroozi, Benjamin Ajibade, Bharat Saxena, Carlos Muñoz Ferrandis, Danish Contractor, David Lansky, Davis David, Douwe Kiela, Duong A. Nguyen, Edward Tan, Emi Baylor, Ezinwanne Ozoani, Fatima Mirza, Frankline Ononiwu, Habib Rezanejad, Hessie Jones, Indrani Bhattacharya, Irene Solaiman, Irina Sedenko, Isar Nejadgholi, Jesse Passmore, Josh Seltzer, Julio Bonis Sanz, Livia Dutra, Mairon Samagaio, Maraim Elbadri, Margot Mieskes, Marissa Gerchick, Martha Akinlolu, Michael McKenna, Mike Qiu, Muhammed Ghauri, Mykola Burynok, Nafis Abrar, Nazneen Rajani, Nour Elkott, Nour Fahmy, Olanrewaju Samuel, Ran An, Rasmus Kromann, Ryan Hao, Samira Alizadeh, Sarmad Shubber, Silas Wang, Sourav Roy, Sylvain Viguier, Thanh Le, Tobi Oyebade, Trieu Le, Yoyo Yang, Zach Nguyen, Abhinav Ramesh Kashyap, Alfredo Palasciano, Alison Callahan, Anima Shukla, Antonio Miranda-Escalada, Ayush Singh, Benjamin Beilharz, Bo Wang, Caio Brito, Chenxi Zhou, Chirag Jain, Chuxin Xu, Clémentine Fourrier, Daniel León Periñán, Daniel Molano, Dian Yu, Enrique Manjavacas, Fabio Barth, Florian Fuhrimann, Gabriel Altay, Giyaseddin Bayrak, Gully Burns, Helena U. Vrabec, Imane Bello, Ishani Dash, Jihyun Kang, John Giorgi, Jonas Golde, Jose David Posada, Karthik Rangasai Sivaraman, Lokesh Bulchandani, Lu Liu, Luisa Shinzato, Madeleine Hahn de Bykhovetz, Maiko Takeuchi, Marc Pàmies, Maria A Castillo, Marianna Nezhurina, Mario Sänger, Matthias Samwald, Michael Cullan, Michael Weinberg, Michiel De Wolf, Mina Mihaljcic, Minna Liu, Moritz Freidank, Myungsun Kang, Natasha Seelam, Nathan Dahlberg, Nicholas Michio Broad, Nikolaus Muellner, Pascale Fung, Patrick Haller, Ramya Chandrasekhar, Renata Eisenberg, Robert Martin, Rodrigo Canalli, Rosaline Su, Ruisi Su, Samuel Cahyawijaya, Samuele Garda, Shlok S Deshmukh, Shubhanshu Mishra, Sid Kiblawi, Simon Ott, Sinee Sang-aroonsiri, Srishti Kumar, Stefan Schweter, Sushil Bharati, Tanmay Laud, Théo Gigant, Tomoya Kainuma, Wojciech Kusa, Yanis Labrak, Yash Shailesh Bajaj, Yash Venkatraman, Yifan Xu, Yingxin Xu, Yu Xu, Zhe Tan, Zhongli Xie, Zifan Ye, Mathilde Bras, Younes Belkada, and Thomas Wolf. 2022. Bloom: A 176b-parameter open-access multilingual language model.

Susan Zhang, Stephen Roller, Naman Goyal, Mikel Artetxe, Moya Chen, Shuohui Chen, Christopher Dewan, Mona Diab, Xian Li, Xi Victoria Lin, Todor Mihaylov, Myle Ott, Sam Shleifer, Kurt Shuster, Daniel Simig, Punit Singh Koura, Anjali Sridhar, Tianlu Wang, and Luke Zettlemoyer. 2022. Opt: Open pre-trained transformer language models.