# Leveraging Pre-Trained Large Language Models for Context Compression

**Satchel Grant**
Department of Psychology
Stanford University
Stanford, CA 94305
grantsrb@stanford.edu

**Sahil Kulkarni**
Department of Statistics
Stanford University
Stanford, CA 94305
sahil1@stanford.edu

## Abstract

Large Language Models (LLMs) have demonstrated remarkable performance on a wide range of language modeling tasks. Furthermore, LLMs have demonstrated an ability to learn new tasks from clever prompt sequences, without the need for gradient updates. The length of an LLM's context window, however, has quadratic computational complexity, making large context windows prohibitively expensive. To address the challenges of large context windows, we introduce a technique that uses pretrained LLMs to create compressed representations of subsequences within the context. We introduce a new token type that can be trained to compress a history of tokens at inference without additional gradient updates after training. We use this technique to augment the open source Bloom models, and we show that the compressed representations can recover 80% of the performance of the LLMs using the full context.

## 1 Introduction

Large Language Models (LLMs) have been shown to be powerful language modeling tools [5, 4, 10], and have been shown to be powerful sequence modeling tools in many other domains [7, 20, 22]. Furthermore, pretrained LLMs have been shown to have a remarkable capacity for learning patterns and tasks in context, without gradient updates [8, 5]. Depending on how the task is prompted to the LLM, the models can often perform far better than having the LLM perform the task directly [23, 27, 12].

Despite LLM's obvious successes, they are often too computationally expensive for researchers with limited funding. Due to the quadratic computational complexity with growing sequence length, even the best funded research institutions eventually reach a performance limit dictated by computational constraints [21]. In the absence of computational constraints, an obvious solution to many remaining short-comings of LLMs is to include a few in-context examples for all types of problems we would like them to solve. This approach, however, quickly becomes intractable due to the computational demands of growing sequence length. This same issue applies for using LLMs to create cognitive models with long-term, personalized memory. Blaring issues with using LLMs as models of cognition are their perfect short-term memory and their, debatably, non-existent long-term memory.

In this work, we propose a context compression algorithm that leverages the computational pathways of pre-trained LLMs. The approach can roughly be described as a combination of prompt tuning [13] and sentence embeddings [18, 14]. We use causal language modeling (next token prediction) as a training signal to compress a length $k$ sub-sequence of the original $n + k$ length context into a sequence of $j$ tokens, where $j < k$. The compressed representation can then be used as a substitute for the $k$ token sub-sequence within the original context. This allows for a reduction of the original context length by $k - j$ tokens. Although the technique requires an initial training that uses gradient

Preprint. Under review.

updates, at inference, no gradients are required for the compression. This technique allows for LLMs to create new tokens on the fly that can act as compact memories. This also opens the possibility of creating entirely new token classes for existing LLMs, allowing the LLMs to adapt to natural changes in language without gradient updates.

Our results are preliminary but show promise. To evaluate performance of the compression, we look at model perplexity on the task of generating the remaining $n$ length uncompressed context that follows the compressed $k$ length sub-sequence. As a baseline, we examine the LLM's perplexity starting from the first few words in the non-compressed sequence, in the absence of the $k$ length compressed representation. As a performance upper bound, we look at the model's forward perplexity when it uses the full, uncompressed $k$ length sub-sequence (N. Goodman, personal communication, February 2023). We find that models using compressed context can recover as much as 80% of the performance upper bound.

## 2 Related Work

A number of approaches have been proposed to address the quadratic complexity with sequence length of transformers. In a review from Tay et. al. 2020, they broke up the methods into six categories: Recurrence, Memory/Downsampling, Learnable Patterns, Sparsity, Fixed/Factorized/Random Patterns, Low Rank/Kernels [19]. Our approach mostly falls under Memory/Downsampling. To the best of our knowledge, our approach is most similar to the Compressive Transformer [17], which applies convolutions over portions of the context window in order to create compressed representations. Their method includes training the transformer itself.

Other approaches to addressing the quadratic complexity have used Locality Sensitive Hashing as a way to only attend to a subset of relevant tokens in context [11]; spaced out the attention mask in various patterns [3]; or performed various approximations on the attention calculations [26].

Our method introduces new, untrained tokens to pre-trained LLMs with frozen weights and then backpropagates error signals through the LLM into the new tokens without updating the LLM itself. This is very similar to Prompt Tuning [13, 9]. Furthermore, we are creating a vector representation (generalizable to multi-vector representations) of a larger chunk of text which is similar to creating a sentence embedding. This has been done in a number of ways such as SBERT, which produces single vector representations of complete sentences using cosine similarity on similar and dissimilar sentence pairs as their training signal [18]. OpenAI has developed sentence embeddings using GPT-3 as well [14]. They use contrastive learning to create the embedding representations, using text that appears next to each other as positive samples. General BERT models are also related to our work in that the CLS token is architecturally similar to the CMPR token introduced in our work [6].

## 3 Dataset and Features

For all of our experiments we use the Openwebtext dataset through the datasets package from Huggingface [1, 24]. For ease of development, we shuffled the dataset and selected a subset of 1 million rows of text. We split these 1 million samples into 80% training and 20% validation samples. From each sample, we took the first 30 tokens as the complete context and split this into $k = 10$ tokens for the sub-sequence that will be compressed (the compression context), and the remaining $n = 20$ tokens were used as the remaining context for causal language modeling (the forward context). See Table 1 for examples.

| Compression Context | Forward Context |
|---|---|
| Officials in High Springs said they were working with deputies | to determine what happened and to help the surviving siblings\n\nWith help from her younger sister, a |
| Update: The World Health Organization declared Zika a global | emergency on Monday. The declaration by the UN agency likely will increase funding and research efforts to control the |

Table 1: Data Examples

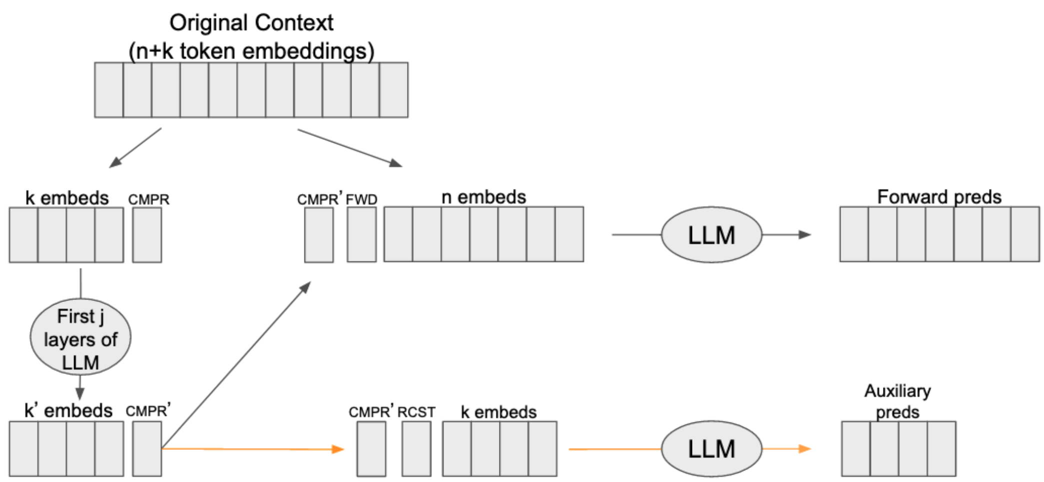


Figure 1: The model architecture. Rectangles represent token embeddings and latent vector representations. Small spaces between embeddings represent a concatenation. Arrows without circles are meant to be visual aids, indicating the order in which the operations occurred. The $k'$ embeds in the lower left portion of the figure are discarded in practice. They are only shown in the diagram to indicate the way in which the pre-trained transformer is used. The orange pathway represents the reconstruction auxiliary task.

# 4 Methods

## 4.1 Compression and Forward Prediction

Our approach is best understood as a variant of the autoregressive task in which a decoder transformer simply learns to predict the next token in a sequence [16]. Our method first introduces $c + 1$ new tokens to the pre-trained LLM. These tokens are the $c$ compression tokens (CMPR) and the forward prediction task token (FWD). The number of compression tokens, $c$, is a choice left up to the experimenter. In all of the experiments in this paper, we set $c = 1$.

During training, an initial sequence of $n + k$ tokens are first converted to their embedding representations and then separated into two subsequences. One is of length $k$, called the compression sequence. The other is of length $n$, called the forward sequence. The $c$ CMPR token embeddings are appended to the end of the compression sequence. It is important that the attention mask allows the CMPR representations to attend to all $k$ embeddings in the compression sequence. Otherwise the CMPR representations do not have the necessary information to encode the entirety of the compression sequence. This sequence of $k + c$ embeddings is then processed through the transformer up to some intermediate layer within the transformer. This intermediate layer is selected before training. We take the representations at the CMPR locations from that intermediate layer as the representation of the compressed sequence. We call this representation CMPR'. We discard the other $k$ vectors.

The CMPR' and FWD embeddings are then prepended to the $n$ length forward context, and they are processed by the transformer. Starting with the location of the FWD token, the outputs are trained using PyTorch's CrossEntropyLoss to predict the next token in the sequence. We use teacher forcing during training, and sample tokens from the softmax over logits using a temperature of 1 during validation. The order of embeddings—CMPR' then FWD—was chosen so that the CMPR' representation doesn't have an additional task of predicting the first token in the forward sequence. Gradients are only backpropagated into the newly introduced tokens.

## 4.2 Reconstruction Auxiliary Task

We experimented with including an auxiliary reconstruction task in an attempt to improve the CMPR' representations. At a high level, this reconstruction auxiliary task is essentially a text-based, autoencoding task [2]. The task consists of first introducing an additional new token to the transformer known as the reconstruction task token (RCST). Then, in addition to the Forward Prediction task with the FWD token described in the last section, the CMPR' is likewise used with the RCST representation to predict the compressed context. The CMPR' and RCST embeddings are prepended to the compression context of $k$ token embeddings, and then the $k + 2$ embedding sequence is processed by the LLM with the task of predicting the next token. The loss from this auxiliary task is simply summed with the loss from the forward prediction task before computing gradients. Gradients are backpropagated into both the RCST token and the CMPR tokens.

### 4.3 Baselines

To gain a sense of how well our method is performing, we note that the model using the compressed representation CMPR' to predict the forward context should be able to perform at least as well as the same model would perform in the complete absence of the compression context. We therefore use this as our performance lower bound. Concretely, we use the transformer prediction loss on the forward context using the first two tokens of the forward context as the initial seed for validation cases (where we do not teacher force).

On the other hand, we cannot expect the transformer's performance using CMPR' to be better than using the full, uncompressed, $k$-length compression context. Thus, we use this as an upper-bound on performance. Concretely, we use the transformer prediction on the $n$ length forward context using the complete $k + n$ length context when teacher forcing. In non-teacher forced cases, we use the $k$-length compression context as the initial seed for predictions. The ideal result would be that the transformer performs equally well using CMPR' as it does using the full compression context.

### 4.4 Training Details

Unless otherwise stated, we used a compression sequence length of $k = 10$ and a forward sequence length of $n = 20$. We used the second to last layer of the LLM for the CMPR' representation. For the transformer architecture, the majority of our experiments used the 560 million parameter Bloom model [25]. In our scaling experiment, we compare this model to the 1.1, 1.7, and 3 billion parameter Bloom models. All baselines shown were taken from the Bloom-560m model. We used 32bit precision. The precision was chosen for backpropagation stability. We used a batch size of 64 which was selected mainly due to memory constraints/processing speeds. We used a starting learning rate of 5e-4 with annealing on loss plateaus, and we used the default dropout from the PyTorch Huggingface pretrained model framework. We used PyTorch [15] for auto-differentiation and NVIDIA Titan Xp GPUs for accelerated computation.

For all reported training values, we used teacher forcing. For all validation values, we did not use teacher forcing, sampling outputs using their softmax over logits with a temperature of 1. Theis applies for both the baselines and the experimental results.

## 5 Experiments / Results / Discussion

Our most important experiment was to see if the technique would perform better than the performance lower bound. We can see from Figure 2 that the Bloom-560m model manages to perform better than the lower bound in both validation loss and accuracy. The Bloom-560m model has a final validation perplexity of 2.75 which recovers 84.6% of the difference between lower and upper bound performance. The lower and upper bounds have validation perplexities of 2.95 and 2.72 respectively. Furthermore, the final validation accuracy of the Bloom-560m model was 1.46% which recovered 80.1% of the performance gap. The lower and upper bounds had 0.79% and 1.62% validation accuracies respectively.

The training loss for the Bloom-560m model is lower than the performance upper bound. We believe this is caused by an overfitting of the model to the training set. We only used 800k samples for training as a way to iterate faster. Additionally, the training set likely has different properties than the dataset that the LLM was originally trained on. For future work, we believe expanding the training set will reduce the overfitting. We could also use the data that the LLM was originally trained on.

The auxiliary reconstruction task did not help performance on the forward task. We note, however, that performance on the reconstruction objective was significantly better than on the forward objective. The final validation perplexity was 2.49 and the top 1 validation accuracy was 2.73%.

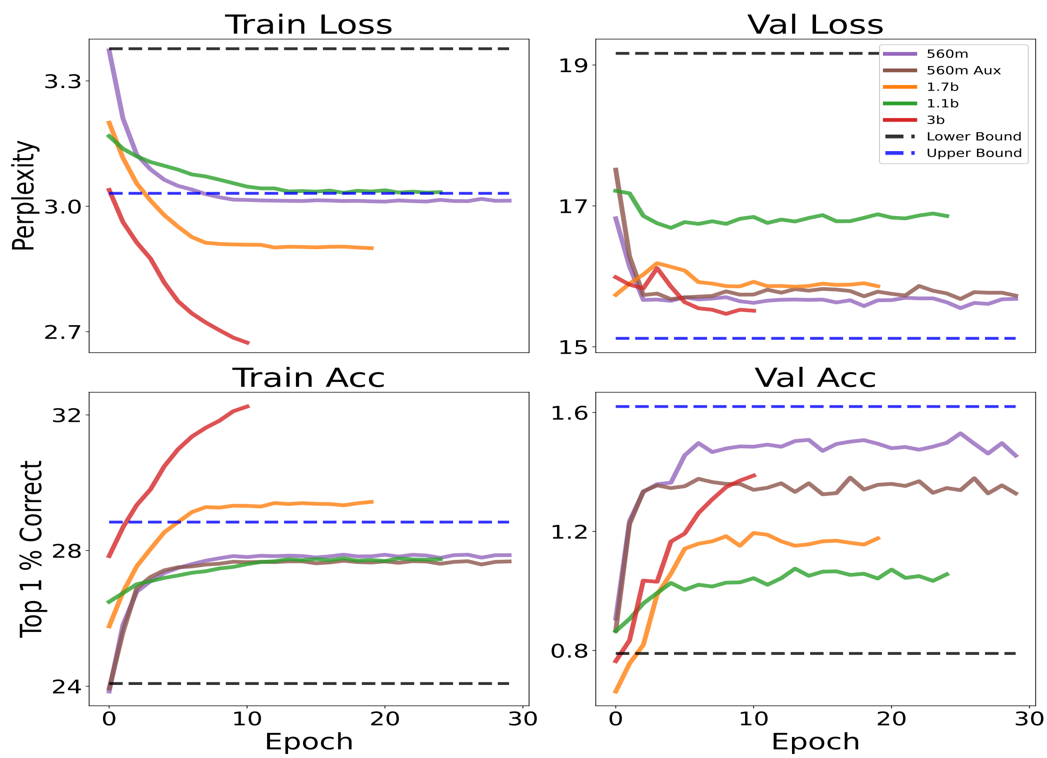


Figure 2: Perplexity of various Bloom models and their top 1 accuracy. The lower (black) and upper (blue) bounds were taken from the Bloom-560m model. "Train" results used teacher forcing, "Val" did not. The 560m Aux model includes an sequence reconstruction auxiliary loss. The training perplexity for the 560m Aux model was too high to include in the figure. We suspect a larger training set with more diverse positional sampling will likely reduce model's ability to overfit to the training set.

We wanted to look at how model scale affects the performance of the compression method. We can see from Figure 2 that increasing scale led to increased overfitting to the training set. In addition to what we've already mentioned, this overfitting may be in part due to picking the learning rate based on a course search with the Bloom-560m (smallest) model.

The validation accuracy for all models, including the performance upper bound, is noticeably low. We suspect this is caused by a few factors. First, we only use data from the start of each text entry. This may bias the data away from the distribution that the Bloom models were trained on. Furthermore, we had no padding on the left side of the inputs which may be a poor performance location due to the fixed positional encodings. Lastly, we were using the top 1 true positive rate for our accuracy metric. This is a limited choice for language modeling, as many words are often equally valid. To gain a better understanding of the models' accuracy, we looked at the models' predicted outputs. We noticed that the compression models would often make incomprehensible predictions that were noticeably worse than the lower and upper bounds. These poor predictions were often of tokens that have a high likelihood of appearing in any context. See Appendix Table 3 for a breakdown of false positive predictions. This is a dramatic shortcoming of our work. See Appendix Table **??** for example outputs.

## 6 Conclusion

In conclusion, we introduced a new context compression method that leverages the abilities of pre-trained LLMs. Despite its shortcomings, we showed that the method has promise, recovering much of the maximum possible performance. There are many next steps to improve upon this work. We find the direction of using summaries generated directly from the LLM as a form of compression a particularly interesting direction.

## Acknowledgments and Disclosure of Funding

Sahil helped brainstorm the initial ideas, and gave feedback and refinements to the ideas throughout the term. Satchel did everything else. We would like to thank Jay McClelland and the PDP Lab for funding and access to the CCN Cluster for trainings. Thanks to Noah Goodman for the suggestion on lower and upper bound performance metrics. And thank you to the TA's and Professors of CS324. It was a great class!

## References

[1] Ellie Pavlick Stefanie Tellex Aaron Gokaslan*, Vanya Cohen*. Openwebtext corpus. `http://Skylion007.github.io/OpenWebTextCorpus`, 2019.

[2] Dor Bank, Noam Koenigstein, and Raja Giryes. Autoencoders, 2020.

[3] Iz Beltagy, Matthew E. Peters, and Arman Cohan. Longformer: The long-document transformer, 2020.

[4] Rishi Bommasani, Drew A. Hudson, Ehsan Adeli, Russ B. Altman, Simran Arora, Sydney von Arx, Michael S. Bernstein, Jeannette Bohg, Antoine Bosselut, Emma Brunskill, Erik Brynjolfsson, Shyamal Buch, Dallas Card, Rodrigo Castellon, Niladri S. Chatterji, Annie S. Chen, Kathleen Creel, Jared Quincy Davis, Dorottya Demszky, Chris Donahue, Moussa Doumbouya, Esin Durmus, Stefano Ermon, John Etchemendy, Kawin Ethayarajh, Li Fei-Fei, Chelsea Finn, Trevor Gale, Lauren Gillespie, Karan Goel, Noah D. Goodman, Shelby Grossman, Neel Guha, Tatsunori Hashimoto, Peter Henderson, John Hewitt, Daniel E. Ho, Jenny Hong, Kyle Hsu, Jing Huang, Thomas Icard, Saahil Jain, Dan Jurafsky, Pratyusha Kalluri, Siddharth Karamcheti, Geoff Keeling, Fereshte Khani, Omar Khattab, Pang Wei Koh, Mark S. Krass, Ranjay Krishna, Rohith Kuditipudi, and et al. On the opportunities and risks of foundation models. *CoRR*, abs/2108.07258, 2021.

[5] 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. Language models are few-shot learners. *CoRR*, abs/2005.14165, 2020.

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

[7] Linxi Fan, Guanzhi Wang, Yunfan Jiang, Ajay Mandlekar, Yuncong Yang, Haoyi Zhu, Andrew Tang, De-An Huang, Yuke Zhu, and Anima Anandkumar. Minedojo: Building open-ended embodied agents with internet-scale knowledge, 2022.

[8] Shivam Garg, Dimitris Tsipras, Percy Liang, and Gregory Valiant. What can transformers learn in-context? a case study of simple function classes, 2022.

[9] Yun He, Huaixiu Steven Zheng, Yi Tay, Jai Gupta, Yu Du, Vamsi Aribandi, Zhe Zhao, YaGuang Li, Zhao Chen, Donald Metzler, Heng-Tze Cheng, and Ed H. Chi. Hyperprompt: Prompt-based task-conditioning of transformers, 2022.

[10] Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B. Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. Scaling laws for neural language models. *CoRR*, abs/2001.08361, 2020.

[11] Nikita Kitaev, Łukasz Kaiser, and Anselm Levskaya. Reformer: The efficient transformer, 2020.

[12] Takeshi Kojima, Shixiang Shane Gu, Machel Reid, Yutaka Matsuo, and Yusuke Iwasawa. Large language models are zero-shot reasoners, 2022.

[13] Brian Lester, Rami Al-Rfou, and Noah Constant. The power of scale for parameter-efficient prompt tuning, 2021.

[14] Arvind Neelakantan, Tao Xu, Raul Puri, Alec Radford, Jesse Michael Han, Jerry Tworek, Qiming Yuan, Nikolas Tezak, Jong Wook Kim, Chris Hallacy, Johannes Heidecke, Pranav Shyam, Boris Power, Tyna Eloundou Nekoul, Girish Sastry, Gretchen Krueger, David Schnurr, Felipe Petroski Such, Kenny Hsu, Madeleine Thompson, Tabarak Khan, Toki Sherbakov, Joanne Jang, Peter Welinder, and Lilian Weng. Text and code embeddings by contrastive pre-training, 2022.

[15] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopt, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. pages 8024–8035, 2019.

[16] Alec Radford, Karthik Narasimhan, Tim Salimans, Ilya Sutskever, et al. Improving language understanding by generative pre-training. 2018.

[17] Jack W. Rae, Anna Potapenko, Siddhant M. Jayakumar, and Timothy P. Lillicrap. Compressive transformers for long-range sequence modelling, 2019.

[18] Nils Reimers and Iryna Gurevych. Sentence-bert: Sentence embeddings using siamese bert-networks, 2019.

[19] Yi Tay, Mostafa Dehghani, Dara Bahri, and Donald Metzler. Efficient transformers: A survey, 2020.

[20] Minh Tran and Mohammad Soleymani. A pre-trained audio-visual transformer for emotion recognition. *CoRR*, abs/2201.09165, 2022.

[21] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. Attention is all you need. In I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017.

[22] Rui Wang, Dongdong Chen, Zuxuan Wu, Yinpeng Chen, Xiyang Dai, Mengchen Liu, Yu-Gang Jiang, Luowei Zhou, and Lu Yuan. BEVT: BERT pretraining of video transformers. *CoRR*, abs/2112.01529, 2021.

[23] Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Ed H. Chi, Quoc Le, and Denny Zhou. Chain of thought prompting elicits reasoning in large language models. *CoRR*, abs/2201.11903, 2022.

[24] 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. Huggingface's transformers: State-of-the-art natural language processing, 2019.

[25] 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 Ife-oluwa 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, Sanchit 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. Bloom: A 176b-parameter open-access multilingual language model, 2022.

[26] Uladzislau Yorsh and Alexander Kovalenko. Linear self-attention approximation via trainable feedforward kernel. In *Lecture Notes in Computer Science*, pages 807–810. Springer Nature Switzerland, 2022.

[27] Denny Zhou, Nathanael Schärli, Le Hou, Jason Wei, Nathan Scales, Xuezhi Wang, Dale Schuurmans, Claire Cui, Olivier Bousquet, Quoc Le, and Ed Chi. Least-to-most prompting enables complex reasoning in large language models, 2022.

# A Appendix

| Word | Avg Loss | Top 1 | Top 5 | p(Word) | False Pos | p(Pred) | Loss P | False Pos P |
|---|---|---|---|---|---|---|---|---|
| the | 6.289 | 0.133 | 0.267 | 0.034 | 0.96 | 0.116 | 0.214 | 0.112 |
| , | 5.951 | 0.12 | 0.334 | 0.033 | 0.963 | 0.108 | 0.194 | 0.104 |
| a | 6.41 | 0.063 | 0.238 | 0.019 | 0.973 | 0.046 | 0.124 | 0.045 |
| of | 7.307 | 0.063 | 0.136 | 0.021 | 0.97 | 0.044 | 0.151 | 0.042 |
| to | 6.937 | 0.034 | 0.103 | 0.018 | 0.972 | 0.023 | 0.128 | 0.022 |
| \n\n | 7.334 | 0.056 | 0.174 | 0.012 | 0.97 | 0.023 | 0.088 | 0.022 |
| and | 6.198 | 0.023 | 0.216 | 0.014 | 0.984 | 0.02 | 0.084 | 0.02 |

Table 2: Metrics for words with the most False Positive predictions from the Bloom-560m model using the CMPR method without teacher forcing. Avg Loss is the negative log likelihood, Top 1 and 5 are whether the correct label appeared in the top 1 or 5 model predictions, p(Word) is the probability of the word occurring in the evaluation data. False Pos is the number of times the model predicted the word incorrectly out of the total number of times it predicted the word, p(Pred) is the portion of times the model predicted a word out of all predictions, Loss P is Avg Loss multiplied by p(Word), False Pos P is False Pos multiplied by p(Pred). Values were selected and ordered based on the largest False Pos P values.

| | |
|---|---|
| **Sample 1** | |
| Cmpr Context: | The Lakers have gone from the leadership of a zen |
| Fwd Target: | master to the guidance of a dragon master. The new coach of the Lakers recently revealed his |
| Lower Bound: | master hood</s>Debussy whistle nogo must be smooth, posee seductive, |
| Predictions: | in conference, years, the NBANBA, the the the impressive- farign team, Kaep |
| Upper Bound: | opera drama to a limelight</s>cwoller down to a modern-day Alabama</s>Yes, |
| **Sample 2** | |
| Cmpr Context: | Blake is the content manager for DailyMTG |
| Fwd Target: | .com, making him the one you should email if you have thoughts on the website, good or |
| Lower Bound: | .com, Google Blogger, Hobby and Facebook are currently the most popular blogs. These have |
| Predictions: | is is clothingillette Glass is the United game, a videoetime, which, which time reports that |
| Upper Bound: | Blake is responsible for building and RSS Whole Mind postings.</s>It is responsible for building and |

Table 3: Examples of the data, the model's predictions, and the upper and lower bound predictions.