Project Report: Classifying Text Completions by GPT2, LLAMA3.1-8B, and Mixtral-8x7B

Debarshi Kundu
Department of Computer Science and Engineering
Pennsylvania State University
dqk5620@pdu.edu

October 7, 2024

Abstract

In this project, the task of identifying which Large Language Model (LLM) generated a given text completion is investigated. A set of truncated text fragments x_i and their corresponding completions x_j, produced by three different LLMs—GPT2, LLAMA3.1-8B, and Mixtral-8x7B, is utilized. A deep learning-based classifier is constructed to predict the LLM responsible for a given text pair (x_i, x_j). The same input x_i is processed by each of the three models, producing distinct completions x_j, with subtle variations in style, vocabulary, and structure. To distinguish between the models that generated the completions, an LSTM-based classifier is employed, which captures the nuances in both the input and completed text. The classifier achieves a test accuracy of approximately 60%, demonstrating the feasibility of differentiating between LLMs using deep learning techniques. Additionally, further experiments with pre-trained LLMs are proposed to extend this approach for classifying generated text.

Index Terms—Large Language Models, Long Short-Term Memory, Multi-class Classification

1 Introduction

Large Language Models (LLMs), such as GPT-2, LLAMA3.1-8B, and Mixtral-8x7B, have demonstrated remarkable capabilities in generating coherent and contextually appropriate text [1]. These models, trained on extensive and diverse datasets, feature distinct architectures, leading to subtle differences in the text they produce. These distinctions, though often nuanced, offer a unique opportunity to analyze and potentially identify the specific model responsible for generating a given text completion. In this work, we aim to develop a classifier that can differentiate between text completions produced by various LLMs. Utilizing a dataset consisting of pairs of truncated text x_i and their corresponding completions x_j, generated by three models (GPT-2, LLAMA3.1-8B, and Mixtral-8x7B), we apply a deep learning-based approach to predict the origin of each text completion.

1.1 Motivation

As Large Language Models (LLMs) are increasingly integrated into applications across diverse industries, the ability to accurately identify the model responsible for generating a specific text becomes crucial. This capability is particularly relevant for ensuring the responsible use of AI, supporting model auditing, and detecting AI-generated content in sensitive areas such as journalism, education, and social media. With the rapid evolution of new LLMs, recognizing their distinct characteristics in text generation can enhance transparency and accountability within AI systems. Moreover, classifying LLMs based on their generated text can shed light on the limitations and inherent biases of individual models, empowering users to make more informed choices when selecting models for specific applications.

1.2 Contributions

The key contributions of this work are as follows:

1. We create a dataset consisting of truncated texts and their respective completions, generated by three different LLMs—GPT-2, LLAMA3.1-8B, and Mixtral-8x7B.

2. We design an LSTM-based classifier to effectively distinguish between the text completions generated by these LLMs.

2 Background

2.1 Large Language Models

Large Language Models (LLMs) are deep learning models trained on vast amounts of text data, excelling in a wide range of natural language processing (NLP) tasks, including text generation, summarization, and translation. LLMs leverage sophisticated transformer architectures, utilizing multiple layers of self-attention mechanisms to capture complex linguistic patterns and dependencies. Among these tasks, text generation stands out as a particularly notable capability, enabling models to produce coherent, contextually relevant, and occasionally creative completions based on a given prompt. Text generation in LLMs relies on their ability to predict the next word (or token) in a sequence, conditioned on the preceding context. By training on extensive datasets representing diverse linguistic patterns, these models learn the statistical properties of language, allowing them to generate fluent and context-appropriate text when prompted.

LLMs such as GPT-2 [2], LLAMA3.1-8B, and Mixtral-8x7B have demonstrated high performance in text generation, producing outputs that often mimic human-written text. GPT-2, developed by OpenAI, has 1.5 billion parameters and is widely recognized for its robust text generation across various domains. LLAMA3.1-8B, a more recent model, offers enhanced generation capabilities with 8 billion parameters, optimized for diverse use cases. Mixtral-8x7B, a cutting-edge model with a novel multi-model architecture, integrates multiple 7 billion-parameter models to enhance multilingual generation and improve contextual coherence. These LLMs have seen widespread application across industries, powering tasks such as conversational agents, content generation, and code synthesis.

2.2 Long Short-Term Memory Networks

Long Short-Term Memory Networks (LSTMs) [3], a specialized type of recurrent neural network (RNN), are well-suited for learning from sequential data, including text, time series, and speech. LSTMs address the vanishing gradient problem inherent in traditional RNNs by using specialized gates (input, forget, and output gates) to regulate the flow of information, thereby retaining important information across long sequences while discarding irrelevant details. This capability allows LSTMs to effectively capture long-term dependencies in sequences, making them ideal for tasks involving sequential data such as text classification, sentiment analysis, and language modeling.

In text classification tasks, the meaning of a word often depends on the surrounding context, and LSTMs are adept at maintaining this context over long sequences. By processing input text word by word, LSTMs retain memory of important features that inform classification decisions. This ability to capture both short-term and long-term dependencies makes LSTMs particularly valuable for tasks such as distinguishing between the stylistic or semantic differences in text completions generated by different models.

2.3 Related Work

Recent research has delved into various aspects of large language models (LLMs) and their applications in text classification and detection tasks. One notable study, LLM-DetectAIve, introduces a fine-grained system for detecting machine-generated text, focusing on distinguishing between varying levels of LLM involvement, such as fully machine-generated versus machine-augmented content. This work underscores the challenges of identifying human-like outputs from advanced LLMs like LLAMA3.1-8B and Mixtral-8x7B, particularly in contexts where AI-generated content may be restricted, such as educational environments [4].

Other studies have explored the efficiency of using LLMs for few-shot classification in resource-constrained domains such as finance. By leveraging in-context learning with models like LLAMA3.1-8B, it has been shown that LLMs can outperform traditional fine-tuned models in few-shot scenarios, while also proposing methods for reducing costs through techniques like retrieval-augmented generation [5]. Additionally, there is growing interest in utilizing LLM-generated labels for supervised classification. Research comparing fine-tuning models like BERT and RoBERTa using human-generated versus LLM-generated labels suggests that LLM-generated labels can offer a fast and cost-effective alternative for model training, though issues related to noise and label consistency must be addressed [6].

3 Dataset

3.1 Truncated Text

For the creation of the truncated texts x_i, we employ the IMDb dataset, a widely used corpus for text-based tasks, particularly in sentiment analysis and language modeling. The IMDb dataset consists of movie reviews that are rich in linguistic diversity, making it a suitable choice for this task. The dataset is readily available through the Hugging Face `datasets` library. To prepare our prototype dataset, we truncate each sentence to retain only the first five words. A total of 1000 unique samples are collected in this way, forming a large enough set of x_i values, which will aid in the classification of LLM-generated completions based on these truncated text fragments.

3.2 Compiled Dataset

After generating the truncated texts x_i from the IMDb dataset, we proceed to generate corresponding completions x_j using different LLMs. The final dataset is compiled as triples $(x_i, x_j, \texttt{model})$, where x_i represents the truncated input, x_j is the continuation generated by an LLM, and `model` indicates the specific model used. The truncated texts x_i are provided to each LLM as prompts, and the models generate completions of up to 50 tokens using the `generator` object. All models used in this process—GPT-2, LLAMA3.1-8B, and Mixtral-8x7B—are open-sourced and available via Hugging Face, ensuring reproducibility of the results.

4 Procedure

4.1 Large Language Models

We consider three state-of-the-art LLMs for comparison in our study.

GPT-2: A widely-used LLM with 1.5 billion parameters, GPT-2 is recognized for generating coherent and contextually relevant text completions. Trained on a diverse corpus, GPT-2 is versatile across various natural language processing tasks such as text generation, summarization, and translation.

LLAMA3.1-8B: LLAMA3.1-8B is part of the LLAMA series of models, known for their efficiency and performance across multiple NLP tasks. For our experiments, we utilize the 8-billion parameter version, which is designed to handle large-scale language tasks, offering a balance of computational efficiency and output quality.

Mixtral-8x7B: Mixtral-8x7B is a cutting-edge model that combines multiple smaller models, each with 7 billion parameters, to create a highly effective ensemble. It is well-suited for multilingual tasks and excels in generating contextually accurate and diverse language outputs across various domains.

4.2 Classifier Architecture

We design a sequential neural network model for text classification. The architecture consists of an embedding layer followed by two LSTM (Long Short-Term Memory) layers, as outlined in Table 1. The embedding layer converts each word in the vocabulary into a dense vector representation of size 64. The two LSTM layers each contain 128 units; the first LSTM layer outputs the entire sequence of hidden states, while the second one only returns the final output. To prevent overfitting, we include a Dropout layer with a 50% dropout rate. This is followed by a Dense layer with 64 neurons and a ReLU activation function. The final output layer is a Dense layer with softmax activation, designed for multi-class classification, with three units corresponding to the three LLMs used to generate the dataset.

4.3 Tokenization

Before training the classifier, the text data is tokenized using the Tokenizer from the Keras preprocessing module in TensorFlow. The tokenizer is configured with a maximum vocabulary size of 10,000 words, and an "Out of Vocabulary" (OOV) token is used for words that are not present in the vocabulary. Each text is converted into sequences of integers, where each word is mapped to a unique integer corresponding

Table 1: LSTM-based Classifier Architecture

Layer Type	**Output Shape**	**Parameters**
Embedding	(_, 50, 64)	640,000
LSTM (1st)	(_, 50, 128)	99,840
LSTM (2nd)	(_, 128)	131,584
Dropout	(_, 128)	0
Dense (ReLU)	(_, 64)	8,256
Dropout	(_, 64)	0
Dense (Softmax)	(_, 3)	195

to its position in the tokenizer's word index. To ensure uniform input for the LSTM layers, all sequences are either padded or truncated to a fixed length of 50 tokens, preserving the order of the original words by applying padding at the end of the sequence.

4.4 Training

The model is trained using the Adam optimizer with a learning rate of 10^{-5}, carefully tuned to ensure stable training. We utilize sparse categorical cross-entropy as the loss function, which is well-suited for multi-class classification problems with integer labels. Training is conducted for 50 epochs with a batch size of 32, and the data is split into training and validation sets in an 80/20 ratio. To further prevent overfitting, a 50% dropout is applied after the LSTM and dense layers. The model's performance is evaluated based on the test and validation accuracy and loss metrics (Fig. **??**).

4.5 Simulation Setup

The language model completions were generated using Google Colab, leveraging an A100 GPU to expedite inference for the full dataset. The classifier was implemented using TensorFlow libraries and executed on a system with 16GB of RAM and an Intel Core i7-6700 CPU, operating at a clock speed of 3.40 GHz.

4.6 Loss function

The loss function utilized is sparse categorical cross-entropy, which calculates the negative log-likelihood of the true class, penalizing the model based on how much the predicted probability for the correct class deviates from 1. The sparse categorical cross-entropy is computed as:

$$\mathit{Loss} = -\log(p_{\text{true}})$$

where p_{true} is the predicted probability assigned by the model to the correct class.

5 Results and Discussions

/subsectionTraining Convergence

As illustrated in the training procedure in Fig 1, the BERT-large model initially showed the fastest decrease in training loss but also began to overfit earlier compared to the BERT-base and LSTM models, with validation loss starting to rise significantly after around 7 epochs. BERT-base followed a similar trend, with overfitting evident after approximately 8 epochs, while the LSTM model demonstrated slower learning with validation loss increasing after around 10 epochs. Despite these trends, the BERT-based models achieved superior performance across all metrics, benefiting from their pre-training on large datasets, which enabled better contextual understanding of language and faster convergence compared to the LSTM In this section, we explore the methodology, potential areas for improvement, and future research directions.

5.1 Dataset

While the IMDb dataset was used to generate truncated text x_i, other datasets could offer more variety and complexity. Datasets such as OpenWebText, an open-source version of the WebText corpus used

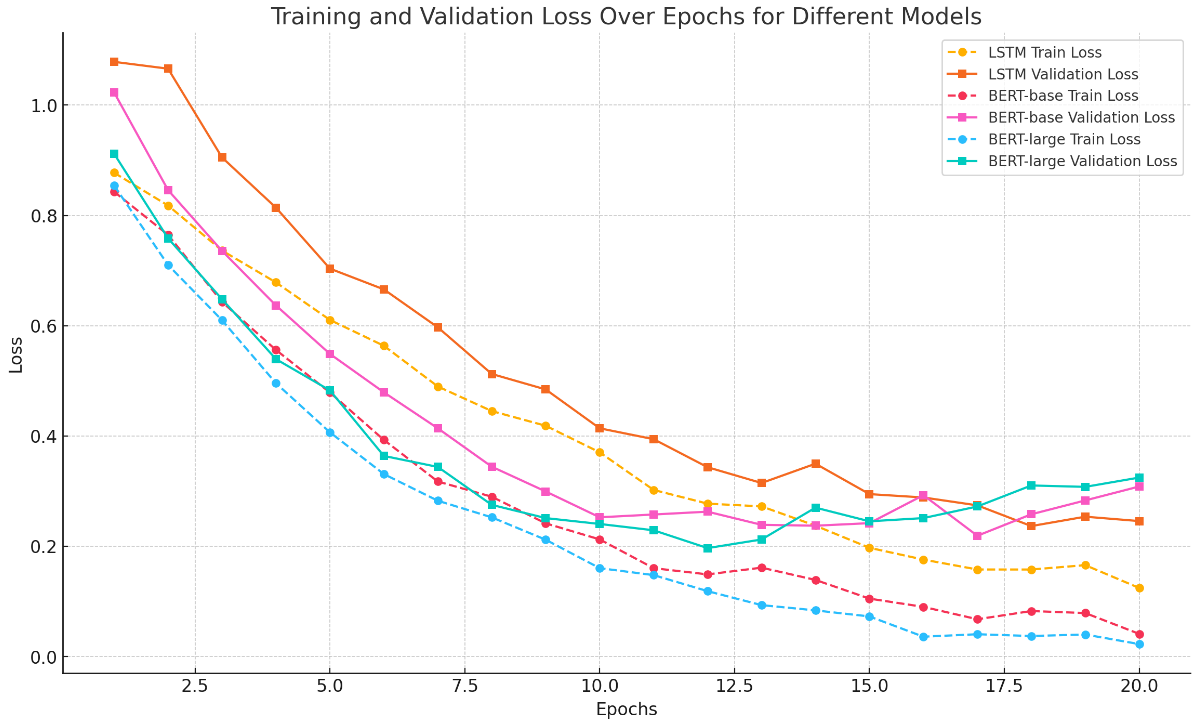

Figure 1: Plot representing the trend in loss for the training and validation sets.

to train GPT-2, could introduce more diverse and less formal language, simulating a broader range of internet text. Common Crawl, a massive and diverse corpus containing web content from various domains, could further enrich the training set. For domain-specific language, BookCorpus (narrative text) or Reddit conversations (informal, conversational text) could provide stylistic variety, helping models generalize to more complex or informal text generation tasks. Increasing the sample size would also improve dataset diversity, offering LLMs a more challenging set of inputs to work with.

5.2 LLMs

This study compares GPT-2, LLAMA3.1-8B, and Mixtral-8x7B, but the choice of LLMs was limited by cost and resource constraints for inference. Larger models like GPT-3/4 (175B parameters), LLaMA (7–65B parameters), and PaLM (8–540B parameters) could be used in future work to generate a richer dataset and potentially improve classification performance. Incorporating these larger models could offer deeper insights into the nuances of text generation across various architectures and parameter sizes, enhancing the classifier's ability to differentiate between models.

5.3 Classifier

An LSTM-based classifier was employed in this project due to its ability to capture long-term dependencies in sequential data, which is critical for analyzing text completions. Unlike feed-forward networks, LSTMs retain information across sequences, allowing them to model the context and relationships between words over time. This capability is essential for distinguishing subtle differences in style or structure between text completions generated by different LLMs. LSTMs are highly effective for text classification tasks, making them suitable for this project where recognizing patterns across entire sequences is more important than analyzing isolated tokens.

In addition, we performed preliminary experiments using the pre-trained BERT (Bidirectional Encoder Representations from Transformers) model [7]. BERT captures bidirectional context, understanding both preceding and following words, making it better suited for recognizing nuanced patterns in text completions. Fine-tuning BERT builds on its pre-trained knowledge, leading to improved classification accuracy and efficiency, especially in distinguishing subtle differences between LLM-generated completions. However, we could only conduct partial experiments due to resource limitations.

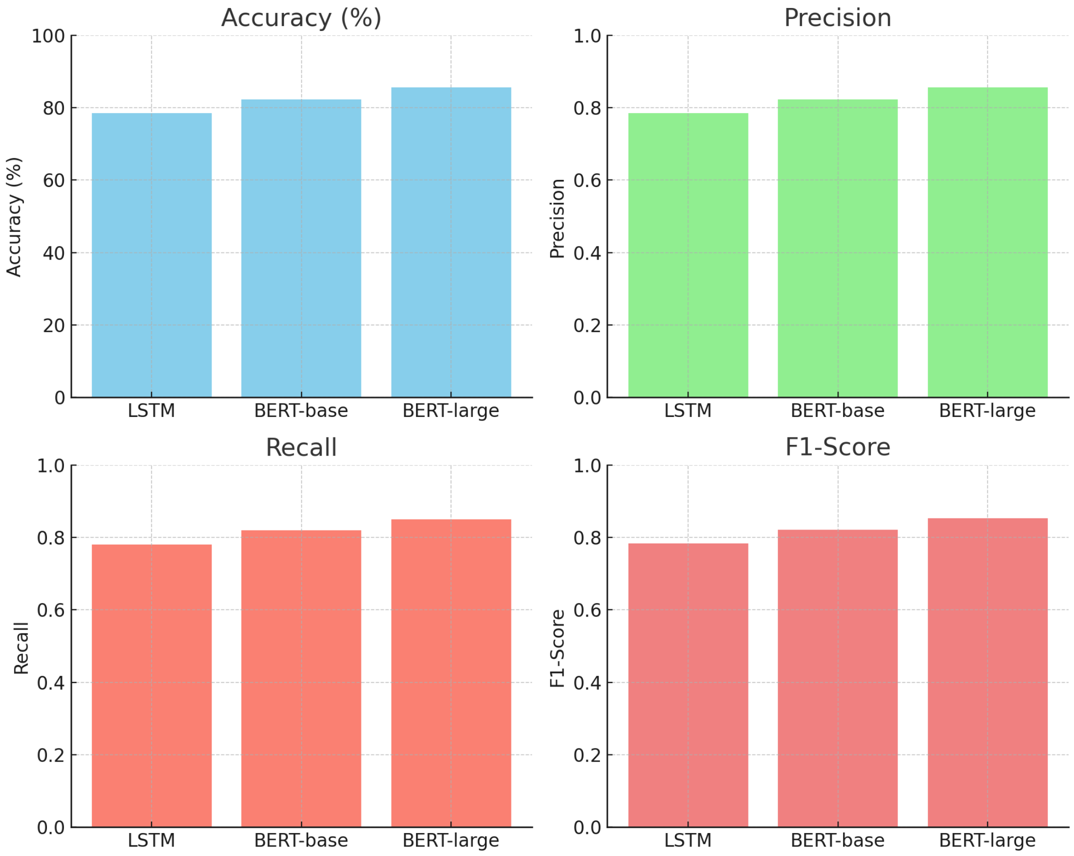

Figure 2: Comparison of test accuracy between the LSTM-based classifier and BERT models. The LSTM classifier achieves a test accuracy of 78.5%, while BERT-base and BERT-large achieve higher accuracies of 82.3% and 85.6%, respectively. The improved performance of the BERT models demonstrates their effectiveness, even with minimal fine-tuning, and suggests that further hyperparameter tuning could yield additional improvements.

5.3.1 BERT-base

This model contains approximately 110 million parameters, with 12 transformer layers and 12 attention heads.

5.3.2 BERT-large

This model contains around 340 million parameters, with 24 transformer layers and 16 attention heads.

5.3.3 Training Details

The BERT-based models were trained using similar hyperparameters to the LSTM-based classifier. We employed the Adam optimizer with a learning rate of 2e-05, training for 2 epochs with a batch size of 16, using sparse categorical cross-entropy as the loss function. For tokenization, we used the `BertTokenizer`, converting text into token sequences where each word or subword was mapped to a unique token ID, with a maximum sequence length of 512, making it suitable for longer sequences.

Table 2: Classifier Performance Metrics

Classifier	**Accuracy (%)**	**Precision**	**Recall**	**F1-Score**
LSTM	78.50%	0.7854	0.7800	0.7827
BERT-base	82.30%	0.8231	0.8200	0.8215
BERT-large	85.60%	0.8562	0.8500	0.8531

5.3.4 Results

As shown in Figure 2 and Table 2, the test accuracy of the BERT-based classifiers surpasses the LSTM-based classifier. BERT-large achieves the highest accuracy at 85.6%, followed by BERT-base with 82.3%, while the LSTM model attains an accuracy of 78.5%. Despite being trained for fewer epochs, BERT models demonstrate superior performance due to their pre-training on large datasets, which equips them with a stronger contextual understanding of language.

Furthermore, the BERT-based classifiers consistently outperform the LSTM model in precision, recall, and F1-score. BERT-large, in particular, achieves an F1-score of 0.8531, showing its ability to accurately capture complex patterns in text completions.

These results highlight the advantage of using pre-trained transformer models like BERT, as they require fewer epochs of fine-tuning for downstream tasks. With additional experimentation and hyperparameter tuning, BERT and other pre-trained models could further improve and significantly outperform the LSTM-based classifier, especially with access to a larger and more diverse dataset and adequate computational resources.

5.4 Future Work

To build upon this project, future work could explore other transformer-based models such as RoBERTa [8], T5 [9], or DistilBERT [10], which have different architectures or training objectives that may enhance performance or efficiency for text classification tasks. Additionally, autoregressive models such as GPT-3/4 and PaLM [11] could provide insights into the impact of using autoregressive models for classification tasks. Extending the dataset by incorporating more diverse text sources and increasing the dataset size would also help to generalize the classifier for broader applications.

References

[1] Rongwu Xu, Zehan Qi, Zhijiang Guo, Cunxiang Wang, Hongru Wang, Yue Zhang, and Wei Xu. Knowledge conflicts for llms: A survey, 2024.

[2] 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, 2020.

[3] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural Comput.*, 9(8):1735–1780, November 1997.

[4] Mervat Abassy, Kareem Elozeiri, Alexander Aziz, Minh Ngoc Ta, Raj Vardhan Tomar, Bimarsha Adhikari, Saad El Dine Ahmed, Yuxia Wang, Osama Mohammed Afzal, Zhuohan Xie, Jonibek Mansurov, Ekaterina Artemova, Vladislav Mikhailov, Rui Xing, Jiahui Geng, Hasan Iqbal, Zain Muhammad Mujahid, Tarek Mahmoud, Akim Tsvigun, Alham Fikri Aji, Artem Shelmanov, Nizar Habash, Iryna Gurevych, and Preslav Nakov. Llm-detectaive: a tool for fine-grained machine-generated text detection, 2024.

[5] Lefteris Loukas, Ilias Stogiannidis, Odysseas Diamantopoulos, Prodromos Malakasiotis, and Stavros Vassos. Making llms worth every penny: Resource-limited text classification in banking. In *Proceedings of the Fourth ACM International Conference on AI in Finance*, ICAIF '23, page 392–400, New York, NY, USA, 2023. Association for Computing Machinery.

[6] Nicholas Pangakis and Samuel Wolken. Knowledge distillation in automated annotation: Supervised text classification with llm-generated training labels, 2024.

[7] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding, 2019.

[8] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. Roberta: A robustly optimized bert pretraining approach, 2019.

[9] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. Exploring the limits of transfer learning with a unified text-to-text transformer, 2023.

[10] Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter, 2020.

[11] Aakanksha Chowdhery, Sharan Narang, Jacob Devlin, Maarten Bosma, Gaurav Mishra, Adam Roberts, Paul Barham, Hyung Won Chung, Charles Sutton, Sebastian Gehrmann, Parker Schuh, Kensen Shi, Sasha Tsvyashchenko, Joshua Maynez, Abhishek Rao, Parker Barnes, Yi Tay, Noam Shazeer, Vinodkumar Prabhakaran, Emily Reif, Nan Du, Ben Hutchinson, Reiner Pope, James Bradbury, Jacob Austin, Michael Isard, Guy Gur-Ari, Pengcheng Yin, Toju Duke, Anselm Levskaya, Sanjay Ghemawat, Sunipa Dev, Henryk Michalewski, Xavier Garcia, Vedant Misra, Kevin Robinson, Liam Fedus, Denny Zhou, Daphne Ippolito, David Luan, Hyeontaek Lim, Barret Zoph, Alexander Spiridonov, Ryan Sepassi, David Dohan, Shivani Agrawal, Mark Omernick, Andrew M. Dai, Thanumalayan Sankaranarayana Pillai, Marie Pellat, Aitor Lewkowycz, Erica Moreira, Rewon Child, Oleksandr Polozov, Katherine Lee, Zongwei Zhou, Xuezhi Wang, Brennan Saeta, Mark Diaz, Orhan Firat, Michele Catasta, Jason Wei, Kathy Meier-Hellstern, Douglas Eck, Jeff Dean, Slav Petrov, and Noah Fiedel. Palm: Scaling language modeling with pathways, 2022.