Exploring the New Horizon of Sequence Modeling: Unveiling the Potentials and Challenges of Mamba

Saaduddin Mahmud smahmud@umass.edu
Md Ashraful Islam mdashrafuli@umass.edu
Sabrina Zaman Ishita sishita@umass.edu
Amit Sarker asarker@umass.edu
Sarmistha Sarna Gomasta sgomasta@umass.edu

1 Problem statement

Our research project proposes a comprehensive evaluation across a spectrum of NLP and sequence modeling benchmarks of the newly proposed sequence model called Mamba (Gu and Dao, 2023), a recent addition to the generalized state space models (GSSM) (Gu et al., 2021). Unlike its predecessors, Mamba has already distinguished itself within the medium-scale parameter sphere (3-6 billion parameters), showcasing superior performance over the traditional Transformer models (Vaswani et al., 2017) in various data modalities such as language, audio, and genomics. This exceptional performance, coupled with a nearly 5x increase in inference speed, positions Mamba as a potentially transformative approach in the realm of sequence modeling.

Despite its promising start, the comprehensive capabilities of Mamba remain underexplored, particularly in comparison to the broad spectrum of tasks and challenges customarily tackled by its transformer and LSTM (Hochreiter and Schmidhuber, 1997) counterparts. Critical dimensions such as model interpretability, fairness, and bias—essential for the responsible development and deployment of AI technologies—have yet to be thoroughly examined for Mamba models. To bridge these gaps, our project will focus on the following key contributions:

1. **Extensive Evaluation and Analysis of Capabilities:** Systematically evaluate Mamba against state-of-the-art transformer models like LLAMA 2 (Touvron et al., 2023) and Mixtral (Jiang et al., 2024), as well as LSTM models, across a comprehensive range of NLP and sequence modeling benchmarks. In particular, we want to explore four different areas:
   - Arithmetic Sequence Generalization (Jelassi et al., 2023).
   - CoT augmented Fine-Tuning.
   - Temporal Structured Data.
   - In-context Learning.
2. **Exploration of Model Dimensions:** Assess Mamba's performance across crucial dimensions such as fairness, and bias through quantitative analyses.

By systematically addressing these areas, our project aspires to provide the research community with a thorough understanding of Mamba's potential and practical applications. Our goal is to illuminate whether the development of Mamba models represents a worthwhile investment for future research endeavors in the rapidly evolving field of AI and sequence modeling.

2 What you proposed vs. what you accomplished

Overall, we successfully met our primary objectives of evaluating Mamba's capabilities and analyzing its performance on various benchmarks. However, due to constraints in computational resources, we could not fully explore architectural innovations within the Mamba framework. This remains an area for future exploration. Notably, the current version of the project incurred over $200 in computational resource expenses (Google Colab, Vast.ai).

3 Paper Structure

The rest of the paper is structured as follows — We begin with a concise literature review of the evaluation areas we are focusing on. Following that, Sections 5 to 11 each provide a complete analysis and comparison in one of the areas more precisely:

- **Section 5:** Investigating Arithmetic Capability of Language Models
- **Section 6:** Chain-of-Thought Fine-tuning for Arithmetic
- **Section 7:** Evaluating In-Context Learning (ICL) Performance of LMs on Arithmetic and Sentiment Analysis Tasks
- **Section 8:** Investigating Machine Translation of Language Models
- **Section 9:** Mamba-Based Model Updater for M-TGNNs
- **Section 10:** Bias and Fairness Analysis
- **Section 11:** Evaluating Fairness of Pretrained Mamba Models using Shapley Attribution

Finally, we conclude with a discussion, summarizing our contributions.

4 Literature Review

In this section, we commence with an overview of the contemporary landscape of sequence modeling, focusing on Generalized State Space Models (GSSMs) and Mamba. Subsequently, we will introduce the various domains and benchmarks that will be utilized for evaluation.

4.1 Sequence Models

The journey of sequence modeling has transitioned from the simplicity of recurrent layers through the complexity of Transformers, to the efficiency of state space models, culminating in the innovative Mamba model. Initially, Recurrent Neural Networks (RNNs) (Pineda, 1987) and their variants like LSTMs (Hochreiter and Schmidhuber, 1997) dominated the scene with their ability to process sequences by maintaining a hidden state across steps. Despite their early promise, they struggled with long-range dependencies and computational scalability. The Transformer(Vaswani et al., 2017) model revolutionized this landscape with its self-attention mechanism, dynamically focusing on different parts of a sequence to capture complex dependencies, albeit at the cost of quadratic computational complexity for long sequences.

To surmount the limitations of Transformers in handling very long sequences, structured state-space sequence models (SSMs)(Gu et al., 2021) were introduced, blending the strengths of RNNs, CNNs (LeCun et al., 1995), and classical state-space models for efficient sequence processing. Enter the Mamba(Gu and Dao, 2023) model, which refines this approach with selective SSMs, allowing for content-based selective information propagation or omission. Mamba stands out for its hardware-aware computation, achieving unparalleled efficiency and scalability across diverse data modalities like language, audio, and genomics. This can beginning of a potentially significant leap in the evolution of sequence modeling. However, more evaluation is required as the development is fairly new and there are several areas where Mamba has not been compared with Transformers and LSTMs.

4.2 Fine-Tuning

LoRA, proposed by Hu et al. (Hu et al., 2021), is an efficient technique for fine-tuning large language models on specific tasks by updating only a small subset of the model's parameters. This reduces computational costs while maintaining high performance, making it feasible to adapt pre-trained models to specialized tasks such as arithmetic problem-solving. We particularly chose this due to computational resource limitations and full SFT was not possible.

4.3 Arithmetic Capability of Language Models

The investigation of arithmetic capabilities in large language models (LLMs) has become an important area of research, particularly for understanding the models' ability to handle numerical and logical reasoning tasks. One key focus is the generalization of arithmetic skills to inputs that vary in length and complexity, which is crucial for real-world applications.

Recent work by Jelassi et al. (Jelassi et al., 2023) explores the challenges that transformers face in learning basic integer arithmetic and generalizing to longer sequences than those seen during training. Their study highlights the use of relative position embeddings (RPE) to enable length generalization in simple tasks such as addition. The authors found that transformers trained on 5-digit numbers could perform 15-digit sums

when RPE was used. However, this approach failed for multiplication tasks. To address this, they proposed a method called train set priming, where a few long sequences are added to the training set. This priming allowed models trained on 5-digit by 3-digit multiplications to generalize to much longer sequences. Previous studies have examined the arithmetic capabilities of LLMs, noting both strengths and limitations. Brown et al. (Brown et al., 2020) showed that GPT-3 can handle simple arithmetic operations but struggles with more complex calculations involving large numbers. This limitation is often due to the models being trained primarily on textual data, which includes limited examples of large-number arithmetic. With similar motivation in this paper, we compare the arithmetic capability of Mamba and Transformer architecture.

4.4 Chain-of-Thought (CoT) Fine-tuning

The use of Chain-of-Thought (CoT) (Wei et al., 2022) fine-tuning in natural language processing has garnered significant attention in recent years. CoT fine-tuning involves training language models on datasets that include intermediate steps, making it easier for them to process and generate accurate outputs (Kim et al., 2023). This approach has been particularly effective in tasks requiring logical reasoning and arithmetic problem-solving. Recent work on rational augmentation demonstrates that providing intermediate steps during training significantly improves the performance of language models on arithmetic tasks (Zelikman et al., 2023). This study showed that CoT fine-tuning significantly improved model performance on a variety of reasoning tasks, including mathematical reasoning, commonsense reasoning, and multi-hop question answering. In the context of our experiment, combining CoT fine-tuning with LoRA aims to address these limitations by providing structured reasoning paths and efficiently adapting the model to arithmetic tasks. Our results indicate that CoT fine-tuning significantly enhances the model's ability to solve complex multiplication problems, aligning with findings from previous studies (Zelikman et al., 2023). However, the observed decrease in out-of-distribution (OOD) accuracy suggests a need for further research into generalization techniques to ensure robust performance across diverse inputs.

4.5 In-context learning

In-context learning (ICL) has demonstrated the ability of transformers to perform tasks based on a few examples provided in their input. Unlike traditional approaches that rely on explicit training or fine-tuning on a specific task, ICL enables models to infer how to perform a task from the input examples alone, suggesting a form of learning that is more flexible and adaptable (Grazzi et al., 2024; Park et al., 2024). Grazzi et al. (Grazzi et al., 2024) investigate the performance of a structured state space model named Mamba in ICL tasks. Their findings indicate that Mamba matches or even surpasses the ICL performance of transformers in certain scenarios, particularly when handling longer input sequences or tasks that involve simple function approximation and natural language processing problems. Furthermore, (Grazzi et al., 2024) extends the understanding of how models like Mamba and transformers approach ICL tasks. They propose that these models incrementally refine their internal representations in a way that resembles iterative optimization. Performance evaluations across various NLP tasks demonstrated that Mamba models, especially when pre-trained and potentially fine-tuned on large datasets, can achieve competitive ICL performance. Comparison with other models, such as RWKV, LLaMA, and Pythia, across a range of NLP tasks, including algorithmic, translation, and knowledge-based tasks, showed that Mamba models are capable of delivering high accuracy, with their performance improving as the number of model parameters increases (Park et al., 2024). Vilar et al. have explored few-shot prompting strategies for machine translation task on PaLM (Chowdhery et al., 2023) suggesting that the downstream machine translation performance is largely correlated with the quality of in-context examples. Subsequently, several recent works have also explored the ICL capabilities of LLMs for Machine translation tasks (Robinson et al., 2023; Zhang and Toral, 2019).

4.6 Memory-based Temporal Graph Neural Networks MTGNN

Learning on temporal graphs is an area that warrants further exploration, especially concerning the potential of Mamba. Among temporal learning algorithms, memory-based temporal graph neural networks (Rossi et al., 2020; Wang et al.,

2021) consistently outperform others in accuracy for tasks such as node classification or link prediction. However, performance degrades with increase of batch size for memory based TGNNs as message aggregator of the memory updated fails to capture temporal depency fully when performing parallel aggregation and LSTM/GRUCell updates the memory once every batch making the memory stale with missing information for large batches. TGL (Zhou et al., 2022) addresses this issue through random-chunking of the batches which showed marginal improvements for large batch sizes. And, for distributed processing as demonstrated in Dist-TGL (Zhou et al., 2023) and GNNFlow (Zhong et al., 2023), random-chunking alone fails to account for the missing intra-batch dependencies. In this part of the project, we have used a modified M-TGNN model by replacing Aggregator-LSTM combo of memory updater with Mamba block.

4.7 Fairness and Bias

The project proposal provided an extensive overview of our literature review in the realm of fairness in language models (Blodgett et al., 2020; Talat et al., 2022), drawing on works, which have greatly influenced our fairness analysis of Mamba. Here, we also incorporate additional literature reviews conducted after the proposal. Fairness in language models is broadly divided into two main domains: fairness/bias evaluation and bias mitigation. In this study, we assess Mamba's gender fairness and compare the results with multiple baselines. Drawing on seminal works by (Bolukbasi et al., 2016; Matthews et al., 2021), which employ word analogy tests on word embeddings to measure bias in language models, several metrics have been devised, including WEAT (Word Embedding Association Test) (Caliskan et al., 2017), SEAT (Sentence Embedding Association Test) (May et al., 2019), and CEAT (Contextual Embedding Association Test) (Guo and Caliskan, 2021; Zhao et al., 2019; Kurita et al., 2019; Nadeem et al., 2020) etc. Probability-based (Webster et al., 2020) and generated text-based (Liang et al., 2022) metrics have also been proposed to gauge bias in language models. Surveys on fairness evaluation metrics in language models (Gallegos et al., 2023; Li et al., 2023; Silva et al., 2021) indicate a scarcity of robust metrics suitable for LLMs, highlighting most post-word analogy test works (Bolukbasi et al., 2016) being somewhat similar and building upon it. Since models trained through Mamba have not yet, to the best of our knowledge, been evaluated for gender bias, we employ the word analogy test for this purpose.

4.8 Shapley Attribution

Several techniques have been developed to evaluate and mitigate bias in language models. One prominent method involves using SHapley Additive exPlanations (SHAP) to attribute the contributions of different features to the model's predictions. This method has been widely adopted for its ability to provide consistent and accurate feature attribution. Notably, Ghorbani and Zou (2019) extended this concept to measure algorithmic fairness, demonstrating how Shapley values can be used to identify and mitigate biases in machine learning algorithms (Ghorbani and Zou, 2019).

5 Investigating Arithmetic Capability of Language Models

5.1 Overview

In this experiment, we aim to investigate the arithmetic capabilities of various large language models (LLMs) using synthetic datasets designed to test both addition and multiplication operations. We created four distinct datasets, each focusing on different ranges of digits and types of arithmetic operations. Our approach involves fine-tuning pre-trained LLMs using Low-Rank Adaptation (LoRA) to adapt them to these specific arithmetic tasks. We evaluated the performance of the models on in-distribution (ID) and out-of-distribution (OOD) test sets, measuring their accuracy and analyzing their errors to understand their limitations and capabilities in performing precise arithmetic operations. The code of the project can be found in `https://github.com/Saad-Mahmud/CS685`. All the fine-tuned models can be found in `https://huggingface.co/saaduddinM`.

5.2 Dataset

We investigated the arithmetic capabilities of language models using four distinct datasets, each focusing on either addition or multiplication tasks. The datasets are synthetically generated to present arithmetic problems formatted as texts eac consisting of a prompt and followed by an answer. Each

dataset has a training set and an out-of-distribution (OOD) test set with different ranges of numbers.

5.2.1 Dataset Descriptions

- **Addition Small**:
  - **Training Set**: Numbers with 1-5 digits
  - **OOD Test Set**: Numbers with 6-10 digits
- **Addition Large**:
  - **Training Set**: Numbers with 1-25 digits
  - **OOD Test Set**: Numbers with 26-40 digits
- **Multiplication Small**:
  - **Training Set**: Numbers with 1-5 digits
  - **OOD Test Set**: Numbers with 6-10 digits
- **Multiplication Large**:
  - **Training Set**: Numbers with 1-10 digits
  - **OOD Test Set**: Numbers with 11-20 digits

Here are examples from the datasets to illustrate the format:

- **Addition**:
  - Prompt: `<prompt> 00045 + 00032 </prompt>`
  - Answer: `<ans> 00077 </ans>`
- **Multiplication**:
  - Prompt: `<prompt> 00003 * 00002 </prompt>`
  - Answer: `<ans> 00006 </ans>`

Note that the task is posed as a masked causal text generation process. So during training, we concatenated the texts. The datasets are synthetically generated, ensuring controlled and diverse arithmetic challenges. Basic statistics include:

- **Addition Small**: 10000 examples
- **Addition Large**: 10000 examples
- **Multiplication Small**: 10000 examples
- **Multiplication Large**: 10000 examples

5.2.2 Data preprocessing

The preprocessing steps included:

- **Normalization**: Standardizing the length of numbers with leading zeros.
- **Tokenization**: Converting text into tokens suitable for the language model.
- **Splitting**: Dividing the training data into training, testing sets and OOD data to OOD test sets.
- **Padding/Truncation**: Ensuring uniform input lengths for batch processing.

5.3 Baselines

The chosen baselines include various large language models with different architectures and sizes. The baselines used in this experiment are 11 different language models (LMs). These models are:

- Mamba1.4B
- Phi1.5B
- Gemma2B
- RGemma2B
- Mamba2.8B
- Zephyr3B
- Mamba7B
- Llama7B
- Llama8B
- Mistral7B
- Gemma7B

All the models were taken from Huggingface.

5.4 Our approach

We employed LoRA (Low-Rank Adaptation) for fine-tuning the models on the arithmetic datasets. The primary goal was to adapt the pre-trained language models to generate accurate arithmetic answers. The fine-tuning process involved:

- **Library**: HuggingFace Transformers
- **Implementation**: Utilized pre-existing implementations for model fine-tuning with slight modifications.

- **Computing Resources**: Experiments were conducted on NVIDIA 4090 24GB GPUs and 64GB RAM rented from VAST.AI

The dataset was split as follows:

- **Training set**: 80% i.e. 8000 examples
- **Test set**: 20% i.e. 2000 examples
- **OOD set**: 2000 examples.

Here are the training hyperparameters used:

- **Batch size**: 8
- **Learning rate**: 2e-4
- **Epochs**: 8
- **Optimizer**: AdamW

Here is the LoRA config:

- **r**: 8
- **lora_alpha**: 16
- **lora_dropout**: 0.1
- **target_modules Mamba**: ["x_proj", "embeddings", "in_proj", "out_proj"]
- **target_modules Transformer**: ["q_proj", "up_proj", "k_proj", "down_proj", "v_proj"]
- textbftarget_module RecurrentGemma: ["q_proj", "o_proj", "k_proj","v_proj"]
- **task_type**: "CAUSAL_LM"

5.5 Results and Analysis

The results show that:

- ***Mambas Perform Better at Multiplication:** The Mamba models consistently outperformed other models of comparable size in multiplication tasks. For instance, 'Mamba-7B' achieved a training accuracy of 0.53 and a test accuracy of 0.28 in the Multiplication Small dataset compared to Llama 8B a SOTA transformer model. Similarly, in the Multiplication Large dataset, Mamba shows equal performance to Llama 8B.
- ***Poor Performance in Addition Tasks:** The same Mamba models performed poorly on addition tasks. For example, 'Mamba-7B' had a training accuracy of only 0.12 and a test accuracy of 0.07 in the Addition Large dataset compared to the near-perfect accuracy of transformer models.
- ***No Zero-shot Training Effect**: The zero-shot accuracies were negligible across all models in the multiplication datasets, indicating that the pre-trained models did not have inherent arithmetic multiplication capabilities that could be leveraged without fine-tuning. Even in the case of addition, the effect is small. This indicates that the performance of Mamba in multiplication is not an artifact of pre-training but rather its inherent capability.
- ***Out-of-Distribution (OOD) Performance**: The Mamba models showed no capability to generalize to OOD examples. Their OOD accuracies remained at 0, highlighting a significant limitation in handling numbers outside the training range.
- **Gemma Models Show Superior Generalization**: The small Gemma model demonstrated good performance in both addition and multiplication tasks.

5.5.1 Error Analysis

We consider 0-1 Metric for accuracy i.e. an answer is either correct or wrong. Most wrong answers follow the correct text generation structure but ultimately provide wrong answer to the question. For example:

- Prompt: `<prompt> 00045 + 00032 </prompt>`
- Answer: `<ans> 00087 </ans>`

In certain cases, the model produces answers in the wrong format which is also considered wrong:

- Prompt: `<prompt> 00045 + 00032 </prompt>`
- Answer: `<ans>00077<and><ans>00077</ans>`

Finally, in OOD case sometimes the model generates the correct answer up to a length and prints 0 for rests:

- `<prompt>085944181*095363251 </prompt>`

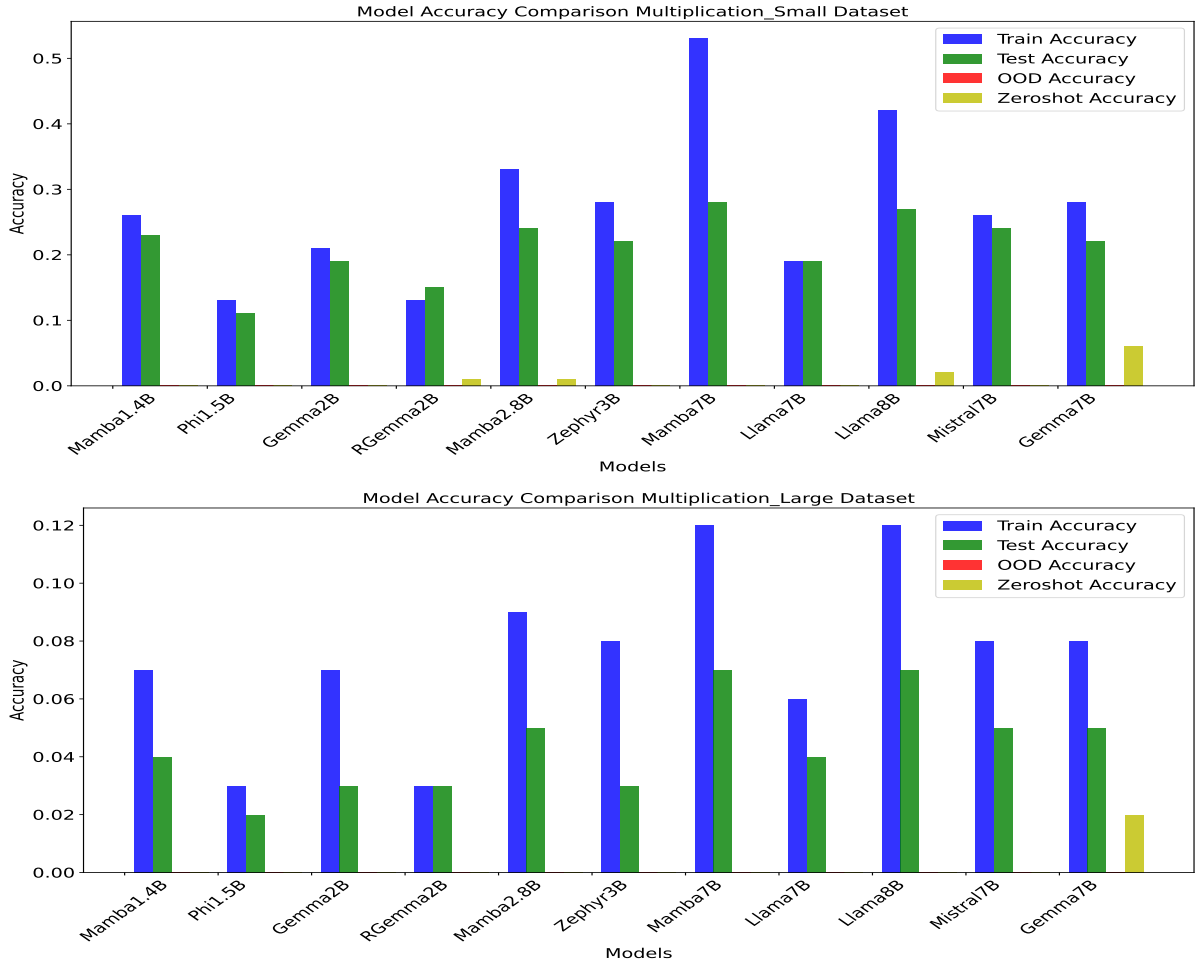

Figure 1: Model Accuracy Comparison for Multiplication Small, and Multiplication Large Datasets

- `<ans> 000307432 </ans>`

Most of the errors occur due to a lack of generalization.

5.6 Conclusion

The results of this experiment demonstrate that Mamba models exhibit a potential advantage in performing certain sequence generation tasks, particularly multiplication, compared to transformer-based models. Despite their superior performance in multiplication tasks, it is noteworthy that multiplication is theoretically more complex than addition due to the need for carrying over digits and handling larger intermediate values. This makes the Mamba models' success in multiplication tasks even more significant. However, the Mamba models consistently underperformed in addition tasks, suggesting a limitation in their adaptability to simpler arithmetic operations.

A critical area for improvement for Mamba models is their out-of-distribution (OOD) performance. The OOD accuracies remained at zero, highlighting a significant limitation in handling numbers outside the training range. Enhancing the generalization capabilities of Mamba models to perform well on OOD examples is an essential next step.

In summary, while Mamba models show promise in specific sequence generation tasks and complex arithmetic operations, there is a clear need for further research and development to improve their performance in simpler tasks and OOD scenarios.

6 Chain-of-Thought Fine-tuning for Arithmetic

6.1 Overview

In this experiment, we investigated the arithmetic Chain-of-Thought (CoT) capability of language models (LMs). We fine-tuned the models using LoRA (Low-Rank Adaptation) on an arithmetic CoT dataset, which comprised two types

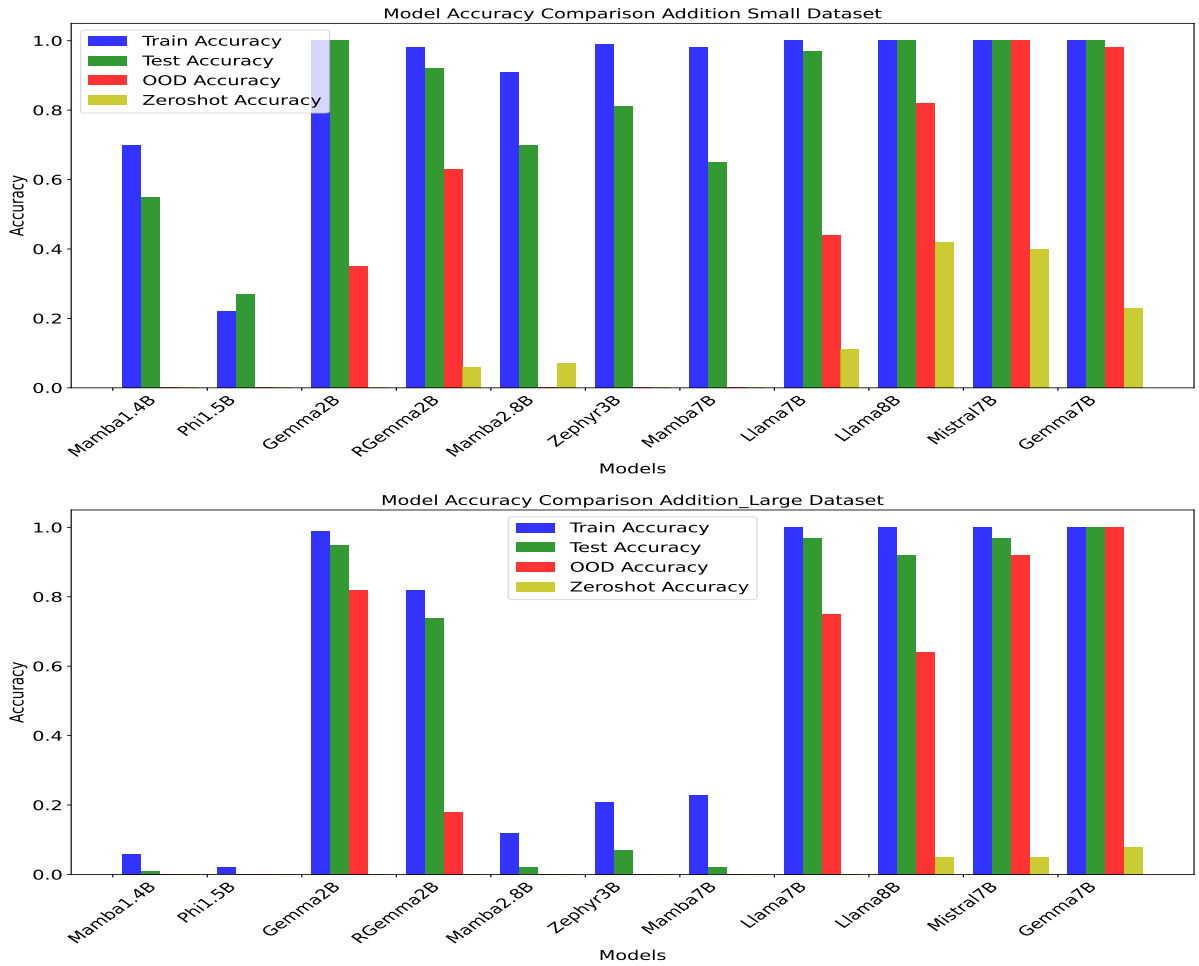

Figure 2: Model Accuracy Comparison for Addition Small, Addition Large Datasets

of texts: one with step-by-step CoT explanations and one without. The CoT texts provided detailed intermediate steps for solving multiplication problems, whereas the without-CoT texts only included the final answers. We created two datasets for comparison: the CoT multiplication dataset, containing 5000 CoT-augmented texts and 5000 normal texts, and the without-CoT multiplication dataset, containing 10000 normal multiplication texts. Several Mamba and transformer LMs were trained on both datasets, and their performance was evaluated on in-distribution (ID) and out-of-distribution (OOD) data. The results demonstrated that models trained on the CoT dataset generally outperformed those trained without CoT. However, differences between the Mamba and Transformer models were negligible indicating they have similar CoT reasoning capabilities. The code of the project can be found in https://github.com/Saad-Mahmud/CS685 . All the fine-tuned models can be found in https://huggingface.co/saaduddinM.

6.2 dataset

The dataset for this experiment consists of arithmetic problems, specifically multiplication tasks. The dataset is divided into two types of texts: one with a step-by-step Chain-of-Thought (CoT) and one without CoT. The CoT texts provide intermediate steps for solving multiplication problems, while the without-CoT texts provide only the final answer.

6.3 Examples from the dataset

Without CoT:

```
<prompt>000085944181*000095363251
</prompt> <ans> 000181307432 </ans>
```

With CoT:

```
<prompt> 0000097321 * 05567
</prompt>
<Chain of Thought>
0000097321 * 7 = 000000000681247
```

```
0000097321 * 6 = 000000005839260
0000097321 * 5 = 000000048660500
0000097321 * 5 = 000000486605000
0000097321 * 0 = 000000000000000
7+0+0+0+0 = 7
4+6+0+0+0 = 0
2+2+5+0+0 = 0
1+9+0+5+0 = 6
8+3+6+0+0 = 8
6+8+6+6+0 = 7
0+5+8+6+0 = 1
0+0+4+8+0 = 4
0+0+0+4+0 = 5
0+0+0+0+0 = 0
0+0+0+0+0 = 0
0+0+0+0+0 = 0
0+0+0+0+0 = 0
0+0+0+0+0 = 0
</Chain of Thought>
<ans> 000000541786007 </ans>
```

The data preprocessing involved normalizing the numbers to a fixed length by padding with zeros. Additionally, the Chain-of-Thought texts were structured to clearly delineate each multiplication step and subsequent addition. No manual annotation was required for this dataset as it is automatically generated based on arithmetic rules.

6.4 Baselines

The baselines for this experiment are:

- `Mamba 2.8B`
- `Stablelm Zephyr 3B`
- `Mamba 7B`
- `Llama-3 8B`

These models were chosen for their varying sizes, providing a broad comparison spectrum. All the models were taken from Huggingface.

6.5 Our approach

Our approach involves fine-tuning the language models using LoRA (Low-Rank Adaptation) on two datasets: one with CoT texts and one without CoT texts. The models were trained to generate the correct answer given the prompt.

- **Library**: HuggingFace Transformers
- **Implementation**: Utilized pre-existing implementations for model fine-tuning with slight modifications.
- **Computing Resources**: Experiments were conducted on NVIDIA 4090 24GB GPUs and 64GB RAM rented from VAST.AI

The dataset was split as follows:

- **Training set**: 80% i.e. 8000 examples
- **Test set**: 20% i.e. 2000 examples
- **OOD set**: 2000 examples.

Here are the training hyperparameters used:

- **Batch size**: 2
- **Gradient accumulation**: 4
- **Learning rate**: 2e-4
- **Epochs**: 8
- **Optimizer**: AdamW

Here is the LoRA config:

- **r**: 8
- **lora_alpha**: 16
- **lora_dropout**: 0.1
- **target_modules Mamba**: [”x_proj”, ”embeddings”, ”in_proj”, ”out_proj”]
- **target_modules Transformer**: [”q_proj”, ”up_proj”, ”k_proj”, ”down_proj”, ”v_proj”]
- textbftarget_module RecurrentGemma: [”q_proj”, ”o_proj”, ”k_proj”,”v_proj”]
- **task_type**: ”CAUSAL_LM”

6.6 Results

Models trained with CoT consistently demonstrated higher test accuracy compared to those trained without CoT. For example, the `Mamba 2.BB` model achieved a 0.17 test accuracy with CoT compared to 0.09 without CoT, representing an 88.9% improvement. Similar improvements were observed across other models. This suggests that the step-by-step guidance provided by CoT helps models better understand and solve arithmetic problems.

However, a notable decrease in OOD accuracy was observed for models trained with CoT. For instance, the OOD accuracy of the `Mamba 2.8B`

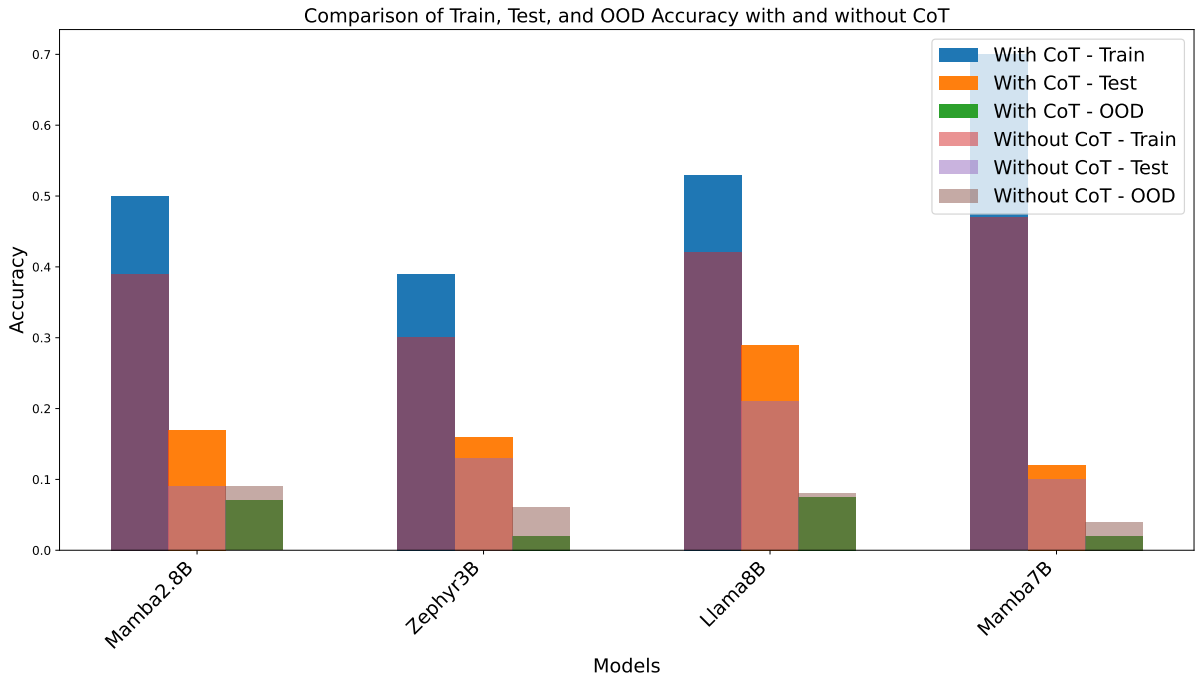

Figure 3: Performance comparison of models on CoT and without-CoT datasets

model was 0.07 with CoT compared to 0.09 without CoT. This decrease can be attributed to the potential overfitting on the CoT examples, where the model relies heavily on the intermediate steps provided during training and struggles to generalize to unseen data where such steps are not available.

6.6.1 Error Analysis

Similar to the previous section, most wrong answers follow the correct text generation structure but ultimately provide wrong answers to math. Additionally, in the OOD case sometimes the model generates the correct answer up to a length and prints 0 for rests Finally, the model produces answers in the wrong format which is also considered wrong:

- `<prompt> 00045 + 00032 </prompt>`
- `<ans> 00077 <ans>`
  `<ans> 00077 </ans>`
  `<Chain of Thought>`

Most of the errors occur due to a lack of generalization and overfitting to reasoning.

6.7 Conclusions

Comparing the Mamba models with the transformer-based models reveals mixed results. The Mamba-2.8B model outperformed the Zephyr-3B model in train, test, and OOD accuracy. Conversely, the Llama-8B model outperformed the Mamba-7B model on the test and OOD sets but underperformed on the train set. Therefore, it is not conclusive whether Mamba or transformer-based models have superior reasoning capabilities. From our experiment, we can only conclude that neither model type has a clear advantage over the other in CoT tasks.

7 Evaluating In-Context Learning (ICL) Performance of LMs on Arithmetic and Sentiment Analysis Tasks

7.1 Overview

This evaluation investigates the In-context learning (ICL) capabilities of pre-trained language models on *arithmetic tasks* and *sentiment analysis*. To perform the experiments, we created synthetic datasets. The main goal is to design prompts using different strategies (zero-shot, few-shot, chain-of-thought) and evaluate the performance of the language models on this task. We performed two types of arithmetic tasks. **(1) Regular Arithmetic:** Involves simple arithmetic operations (addition, subtraction, multiplication, and division[1]). **(2) Jumbled Arithmetic:** Involves arithmetic operations; however, we have introduced three new arithmetic symbols (\$, #, and @). The interpreta-

[1]We have used integer division

tions of these symbols are:

- $: This is addition. So, $a\$b = a + b$.
- #: This is subtraction. So, $a\#b = a - b$.
- @: This is the multiplication of $a+b$ and $a-b$. So, $a@b = (a + b) * (a - b)$.

The motivation for using jumbled arithmetic is to check whether the model is actually learning from the demonstrations or predicting from its memory. For sentiment analysis, the task involves classifying the sentiment (Positive, Negative, or Neutral) of a given statement. The code used for evaluation in this Section 7 is publicly available at https://github.com/amit-sarker/ICL-Analysis-NLP-685.

7.2 Dataset

7.2.1 Regular Arithmetic

For this task, we have created **seven** synthetic datasets using Python scripts with 100 prompts. We have changed the number of demonstrations in each dataset (zero-shot and few-shot prompting). We have created a separate dataset for the zero-shot and few-shot with 0, 2, 4, 6, 8, and 10 demonstrations. We also created another dataset that includes 5 demonstrations with random labels. The prompts include demonstrations for simple arithmetic operations followed by a task that the models need to complete. The operations include two variables, and the values are randomly chosen between (1, 100). Here are two examples of zero-shot and few-shot (with 2 demonstrations):

- Calculate the next problem and provide the answer:
  Calculate 70 * 58? The correct answer is:
- 1. Calculate 8 + 67. The answer is 75.
  2. Calculate 81 + 5. The answer is 86.
  —
  Calculate the next problem and provide the answer:
  Calculate 24 - 47? The correct answer is:

Here is an example prompt of the random label dataset:

- 1. Calculate 11 - 3. The answer is 58.
  2. Calculate 81 + 66. The answer is 66.
  3. Calculate 72 - 63. The answer is 61.
  4. Calculate 10 * 3. The answer is 14.
  5. Calculate 29 + 28. The answer is 38.
  —
  Calculate the next problem and provide the answer:
  Calculate 65 + 60? The correct answer is:

7.2.2 Jumbled Arithmetic

For this task, we have created **six** synthetic datasets with 50 prompts using a Python script. In this task, we have created separate datasets for few-shot (2, 4, 6, 8, 10 demonstrations) and chain-of-thought (CoT) prompting with 3 demonstrations. Unlike regular arithmetic prompts, we have created each prompt with the same arithmetic operations for the few-shot prompting. Otherwise, the models get everything wrong most of the time. For CoT, as we have provided the reasoning, we did not keep this restriction. Moreover, we have used the operation @ = $a * b$ for simplicity in this task. Three demonstrations have been provided with reasoning for three different operations in each prompt. Here are two examples of few-shot (with 2 demonstrations) and CoT prompting:

- 1. Calculate 58 $ 50. The answer is 108.
  2. Calculate 58 $ 35. The answer is 93.
  —
  Calculate the next problem and provide the answer:
  Calculate 97 $ 19? The correct answer is:
- 1. Calculate 45 # 11. The answer is 34.
  Reason: # is subtraction. a # b = a - b. So, 45 # 11 = 34
  2. Calculate 14 $ 9. The answer is 23.
  Reason: $ is addition. a $ b = a + b. So, 14 $ 9 = 23
  3. Calculate 54 @ 3. The answer is 162.
  Reason: @ is multiplication. a @ b = a * b. So, 54 @ 3 = 162
  —
  Calculate the next problem and provide the answer:
  Calculate 79 $ 9? The correct answer is:

7.2.3 Sentiment Analysis

To evaluate the models on this task, **four** synthetic datasets were created (dataset with true label, dataset with random label, dataset with no demonstration, and dataset with CoT prompting) with 50 prompts in each of them. All the datasets contain 3 demonstrations in each prompt, except the dataset with no demonstrations. Each of the

demonstrations is a (Statement, Sentiment) pair. All the demonstrations and final tasks are created using GPT-4o. Here are two examples of the dataset with true label and CoT prompting:

- 1. Statement: The meal was delicious and the service was excellent. Sentiment: Positive.
  2. Statement: It rained all day during our outdoor event. Sentiment: Negative.
  3. Statement: She arrived at the meeting on time. Sentiment: Neutral.
  —
  Classify the sentiment of the following statement:
  Statement: The sunset over the ocean was breathtaking. Sentiment:

- 1. Statement: The meal was delicious and the service was excellent. Sentiment: Positive.
  Reason: The statement describes enjoyment and good service.
  2. Statement: It rained all day during our outdoor event. Sentiment: Negative.
  Reason: The statement describes an unfavorable situation.
  3. Statement: She arrived at the meeting on time. Sentiment: Neutral.
  Reason: The statement provides a factual detail without emotional context.
  —
  Classify the sentiment of the following statement:
  Statement: The sunset over the ocean was breathtaking. Sentiment:

7.3 Baselines

For the evaluation of this task, the following language models are used:

- Mamba-2.8b
- Cerebras-btlm-3b
- Mamba-7b
- Llama2-7b
- Mistral-7b

The Cerebras-btlm-3b was used for a fair comparison with Mamba-2.8b. Llama2-7b and Mistral-7b were used for comparison with Mamba-7b. All the models were taken from Huggingface.

7.4 Our approach

The main goal of this analysis was to evaluate the performance of the pre-trained language models on various ICL tasks. Therefore, the pre-trained models are taken Huggingface without modifying the models.

- **Library:** Huggingface transformers
- **Implementation:** Used existing implementations of the LMs
- **Computing Resource:** Google Colab Pro. T4 15 GB GPUs are mainly used with 51 GB RAM.

The following configurations are used to generate output from each model:

- max_new_tokens=64
- early_stopping=True
- num_return_sequences=1

The model's responses for each dataset were saved to a text file. We have used Python scripts to parse the answers from the responses. We used Python's regex library to extract the numeric answers for the arithmetic tasks. To evaluate the models, standard accuracy is calculated for the arithmetic tasks. For sentiment analysis tasks, we used the Macro-F1 score. The Macro-F1 score is calculated by taking the F1 score for each sentiment class and then averaging them. This means each class contributes equally to the final score, regardless of the number of samples in each class.

7.5 Results and Analysis

This section describes the performance evaluation of the five Language Models (LMs) on three different tasks: regular arithmetic, jumbled arithmetic, and sentiment analysis. We also analyzed each model's performance for a particular task by changing the prompt strategies.

7.5.1 Regular Arithmetic

Figure 4 shows the accuracy of the models in predicting the correct answers to simple arithmetic tasks. Key observations are as follows:

- **Mistral-7b** consistently outperforms the other models, achieving the highest accuracy across all numbers of demonstrations. Its accuracy ranges from around 80% to 90%, indicating robust performance regardless of the demonstration counts.

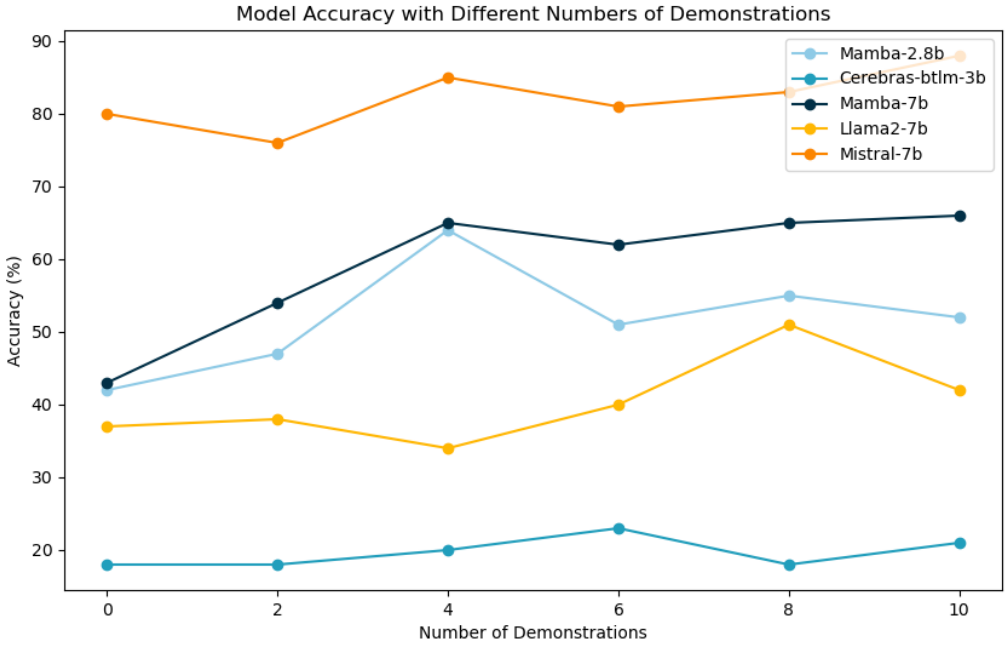

Figure 4: Model Accuracy Comparison for Regular Arithmetic Tasks Using Different Number of Demonstrations

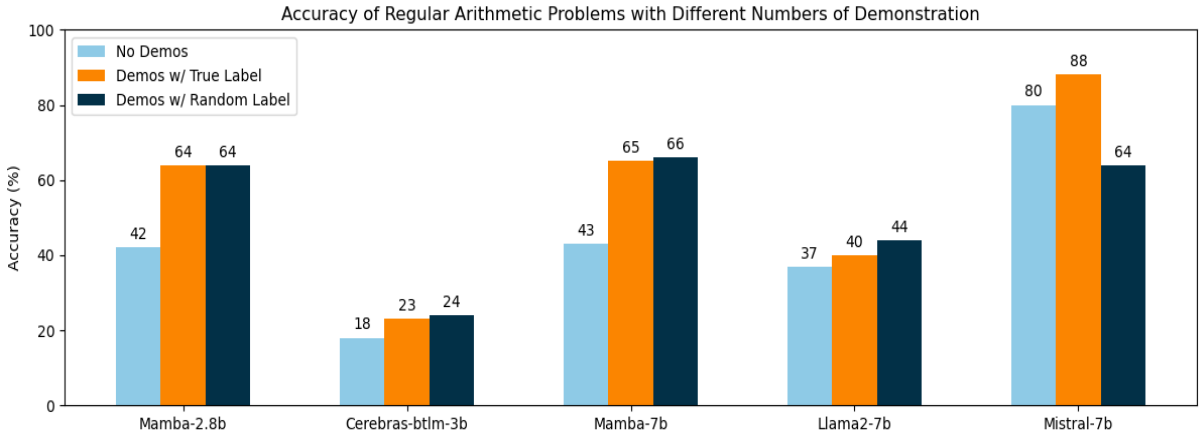

Figure 5: Model Accuracy Comparison for Regular Arithmetic Tasks on Dataset with no Demonstrations, Demonstrations with True Labels, and Demonstrations with Random Labels

- **Mamba-7b** shows an improvement in accuracy as the number of demonstrations increases, peaking at around 65% with 6 demonstrations. This indicates that Mamba-7b benefits from additional demonstrations up to a point. A similar trend can be observed for **Mamba-2.8b**, however, its accuracy drops later.
- **Llama2-7b** demonstrates moderate variability, with accuracy fluctuating between 30% and 50%. It shows the highest improvement at 8 demonstrations, reaching an accuracy of 51%.
- **Cerebras-btlm-3b** remains consistently low in accuracy, hovering around 18% to 24%. This model shows minimal improvement with increasing demonstrations, indicating limited benefit from additional context.

Figure 5 shows the accuracy of the model across various demonstration types. The three categories are zero demonstrations, demonstrations with true labels, and demonstrations with random labels. The analysis reveals that providing demonstrations

enhances the performance of language models on arithmetic tasks, with most models showing improved accuracy. The extent of this improvement varies across models. Overall, the findings highlight the importance of context in enhancing model performance and the varying degrees to which different models can utilize this context.

- **Mamba-2.8b** and **Mamba-7b** benefit the most from demonstrations, displaying robust performance regardless of the accuracy of the labels.
- **Mistral-7b** achieves the highest accuracy with no demonstrations; however, it shows the highest sensitivity to the correctness of the demonstrations. The model achieves exceptional performance with true labels but drops with random labels.
- On the other hand, **Cerebras-btlm-3b** shows limited improvement, which suggests potential limitations in its ability to use contextual information.
- **Llama2-7b** shows the highest accuracy when the labels of the demonstrations are random. The overall performance shows that the model does better when demonstrations are provided.

7.5.2 Jumbled Arithmetic

We changed the arithmetic operations as described in Section 7 to better understand how the models predict the answers to the arithmetic tasks. We planned to change the signs of the operations to see whether the models were predicting based on their pre-training data or if they were actually learning from the demonstrations. Figure 6 shows the models' performance on the jumbled arithmetic tasks.

- **Mamba-2.8b**, **Mamba-7b**, and **Cerebras-btlm-3b** all of these models were poor in terms of accuracy in this task. Mamba-7b shows 2% and 6% accuracy and shows a slight improvement at 6 demonstrations before dropping again. Mamba-2.8b and Cerebras-btlm-3b exhibit the weakest performance; Mamba-2.8b's accuracy ranges between 2% and 8% without a clear trend, while Cerebras-btlm-3b remains consistently at 0% accuracy across all numbers of demonstrations
- **Mistral-7b** consistently outperforms the other models. It shows a clear upward trend in accuracy as the number of demonstrations increases. It starts at 40% accuracy with 2 demonstrations and peaks at 68% with 10 demonstrations, indicating a strong capability to leverage additional context effectively.
- **Llama2-7b** also shows a positive response to increasing demonstrations. Its accuracy increases from 18% at 2 demonstrations to 32% at 4 and 10 demonstrations with some fluctuations.

After getting this accuracy, we decided to use Chain-of-Thought (CoT) prompting to see if we got a better result. Figure 7 shows the performance of the model using CoT prompting.

- **Mistral-7b** stands out with a significantly higher accuracy of 66%, demonstrating its strong ability to leverage the chain of thought approach effectively.
- **Mamba-7b** shows a moderate accuracy of 18%, indicating some benefit from the chain of thought prompting but far less effectively than Mistral-7b. **Llama2-7b** achieves 16% accuracy, showing a slight improvement. But still underperforming relative to the Mistral-7b.
- **Cerebras-btlm-3b** and **Mamba-2.8b** display the lowest accuracies, at 8% and 6%, respectively. This suggests the limited ability to benefit from the chain of thought prompting.

7.5.3 Sentiment Analysis

For sentiment analysis, we utilized four datasets for each model: dataset with no demonstrations, demonstrations with true labels, demonstrations with random labels, and demonstrations with Chain-of-Thought (CoT) prompting. We have calculated the Macro-F1 scores for each model. Figure 8 shows the performance of the models. The analysis reveals that all models benefit from demonstrations, particularly with true labels. CoT prompting significantly enhances performance, especially for Cerebras-btlm-3b and Llama2-7b. Random label demonstrations also contribute to improved performance but to a lesser extent than true labels. Mistral-7b consistently outperforms other models across all conditions,

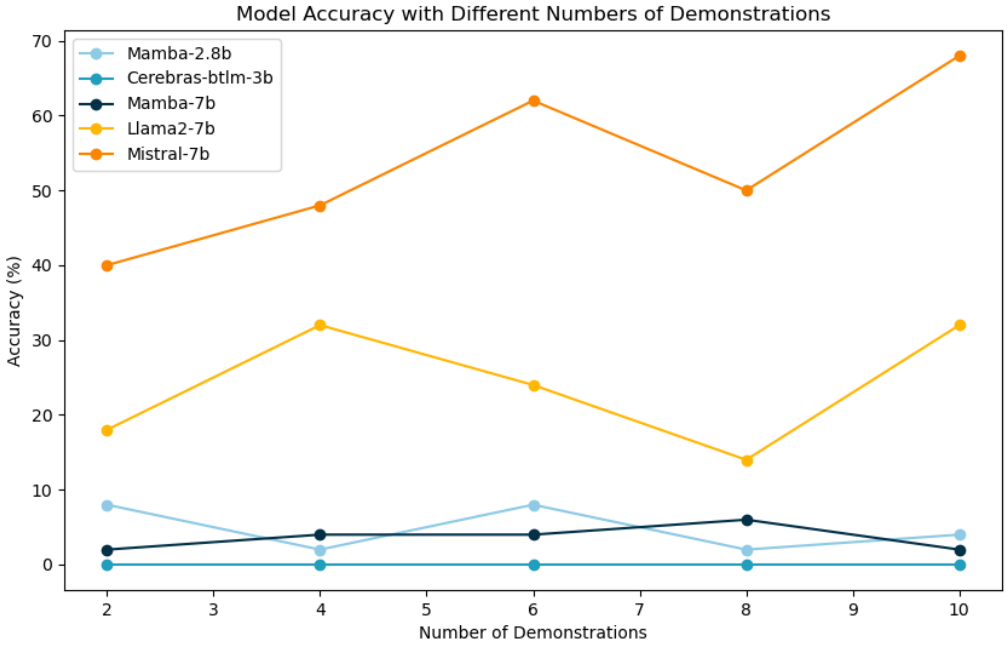

Figure 6: Model Accuracy Comparison for Jumbled Arithmetic Tasks Using Different Number of Demonstrations

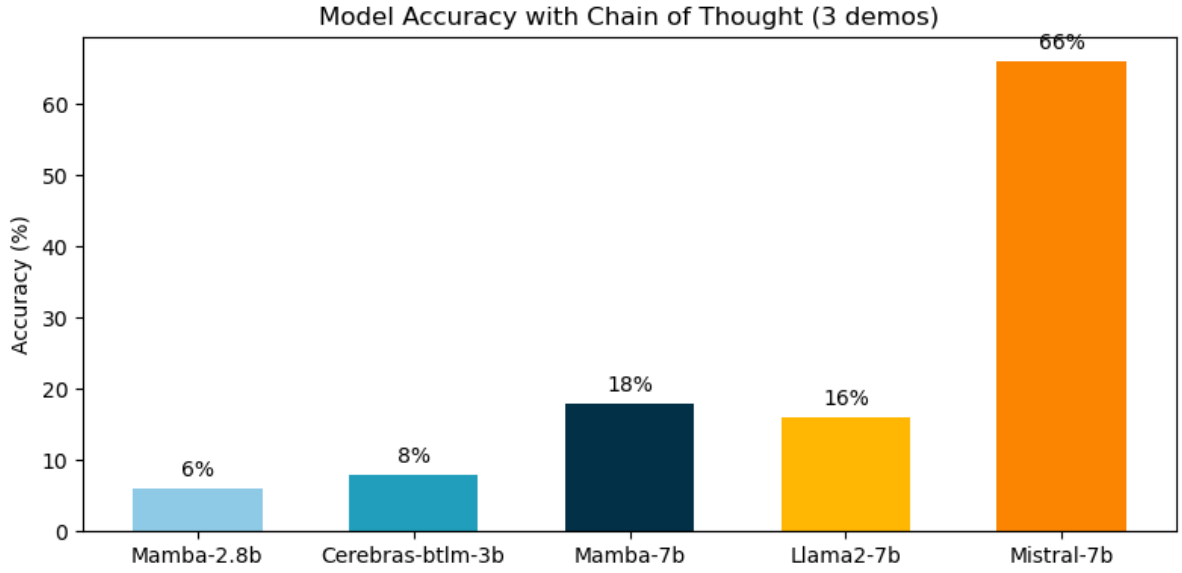

Figure 7: Model Accuracy Comparison for Jumbled Arithmetic Tasks Using Chain-of-Thought (CoT) Prompting

demonstrating its capability to use contextual information for sentiment analysis tasks.

- **Mamba-2.8b** Shows improvement with demonstrations. Its highest Macro-F1 is 76.81% with true label demonstrations. However, CoT prompting results in moderate results (50.16%). The model benefits significantly from accurate context, with a drop when using random labels (56.12%).
- **Mamba-7b** shows moderate performance (52.94%) with no demonstration and shows good improvements with true label (66.3%) and CoT prompting (62.27%). The use of random labels also improves accuracy (59.35%). Therefore, Mamba-7b gains better Macro-F1 when more context is provided.
- **Cerebras-btlm-3b** starts with a low Macro-

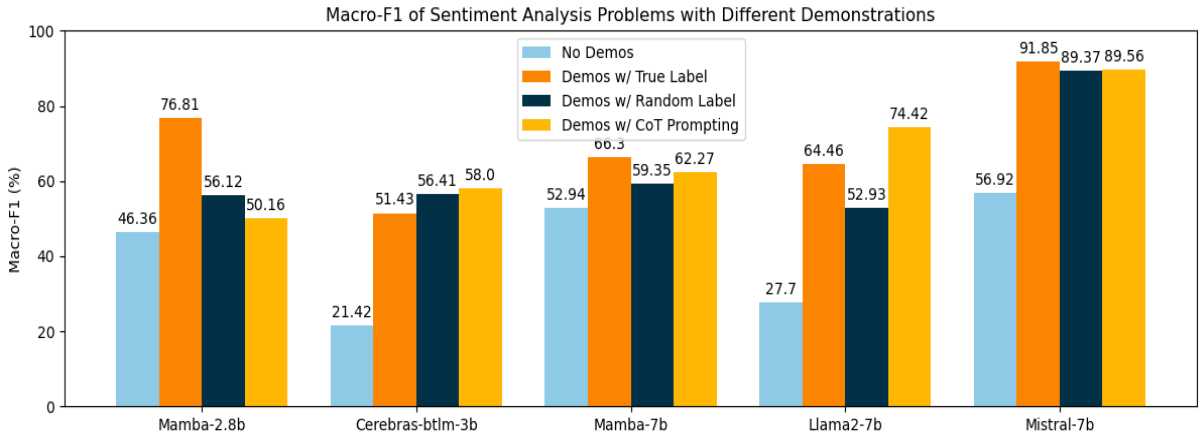

Figure 8: Model Macro-F1 Comparison for Sentiment Analysis Tasks on Dataset with no Demonstrations, Demonstrations with True Labels, Demonstrations with Random Labels, and Demonstrations with Chain-of-Thought (CoT) Prompting

F1 (21.42%) for no demonstrations but benefits the most from CoT prompting (58.0%). It also surpassed its performance with true (51.43%) and random label demonstrations (56.41%). This indicates CoT prompting's effectiveness in enhancing its performance.

- **Llama2-7b** shows the weakest performance without demonstrations (27.7%) but shows significant improvement with CoT prompting (74.42%), followed by true labels (64.46%). Random labels (52.93%) also improve performance, which tells the model's sensitivity to contextual accuracy.

- **Mistral-7b** consistently outperforms all other models, with a high Macro-F1 (56.92%) with no demonstrations and exceptional performance with true label demonstrations (91.85%). It maintains high Macro-F1 with both random labels (89.37%) and CoT prompting (89.56%). This indicates its superior ability to utilize additional context effectively.

7.6 Error analysis

7.6.1 Arithmetic Tasks

In this section, we describe the failures of the five models on arithmetic and sentiment analysis task. Here, we provide the types of arithmetic and sentiment tasks the models could not solve correctly and provide some of the failed tasks as examples. The complete list of errors by each model can be found in the following github links:

- Regular Arithmetic: https://github.com/amit-sarker/ICL-Analysis-NLP-685/Regular Arithmetic/Error Analysis

- Jumbled Arithmetic: https://github.com/amit-sarker/ICL-Analysis-NLP-685/Jumbled Arithmetic/Error Analysis

- Sentiment Analysis: https://github.com/amit-sarker/ICL-Analysis-NLP-685/Sentiment Anlysis/Error Analysis

Mamba-2.8b

The Mamba-2.8b model encounters a variety of errors when attempting to solve arithmetic problems. It fails to solve arithmetic problems due to errors in division and multiplication, handling negative results, and generating responses for complex or larger numerical operations. The model frequently miscalculates division (e.g., 2200 / 40, resulting in 56 instead of 55) and multiplication problems (e.g., 93 * 83, yielding 7723 instead of 7719). It also struggles significantly with negative results, often providing positive counterparts (e.g., 28 - 38 resulting in 10 instead of -10). Additionally, the model often fails to generate any response for more complex calculations (e.g., 5612 / 61), and occasionally produces completely unrelated outputs (e.g., 1 - 97 resulting in 3871). These patterns indicate a need for targeted improvements in handling arithmetic complexity and numerical ranges.

Mamba-7b

The Mamba-7b model exhibits a range of errors when attempting to solve arithmetic problems, particularly struggling with multiplication and division operations, handling negative results, and maintaining precision with larger numbers. The model frequently provides incorrect results for multiplication (e.g., 3237 instead of 3267 for 83 * 39) and division (e.g., 64 instead of 65 for 455 / 7). It also shows a notable pattern of errors in handling negative results, often responding with positive counterparts (e.g., 10 instead of -10 for 55 - 65). Additionally, the model often generates responses that are close but not accurate, suggesting issues with precision (e.g., 5275 instead of 5325 for 75 * 71). These errors indicate the model's difficulty in performing precise calculations and handling more complex numerical operations, highlighting areas where further fine-tuning and training on diverse arithmetic problems could improve its performance.

Cerebras-Btlm-3b

The Cerebras-btlm-3b model shows several types of failures when attempting to solve arithmetic problems. The model exhibits a range of semantic and syntactic errors when solving arithmetic tasks. It particularly struggles with multiplication and division operations, negative results, and problems involving larger numbers. These patterns indicate areas where the model's arithmetic reasoning could be improved. Particularly through fine-tuning on diverse arithmetic problems and enhancing its ability to handle complex calculations and larger numerical ranges. Here are some example problems that the model fails to solve:

In Question: Calculate 24 - 47?, the model responded with 27 instead of -23, indicating a misunderstanding of the subtraction operation. Similarly, in Question: Calculate 43 - 95?, the model responded with 48 instead of -52. In Question: Calculate 54 - 66?, the model responded nothing. Similarly, in Question: Calculate 48 - 98?, the model responded with 50 instead of -50. There are many instances where the model fails to produce any response, such as in Question: Calculate 61 * 70? and Question: Calculate 6478 / 79?. This pattern repeats frequently, suggesting the model may be unable to handle certain numerical ranges or complex operations.

Some responses are incomplete or entirely absent. This indicates potential issues with the model's ability to handle certain calculations or operations. For example, in Question: Calculate 612 / 17?, the response was a series of digits (3.833333333333333) that do not correspond to any reasonable answer. Oftentimes, the model generates responses that are totally out of context. For example, Calculate 89 + 58?, the model's response is "A: You can use the following formula: frac{a+b}{c}=frac{a}{c}+frac{b}{c} A: frac{a+b}{c}=frac{a}{c}+frac".

Sometimes, the model writes code. For example, for Calculate 24 - 47?, the response is: You can use the following code:

```
public static void main(String[] args) {
Scanner sc = new Scanner(System.in);
int n = sc.nextInt();
int m = sc.nextInt();
```

The model fails consistently with multiplication and division problems, either by providing incorrect results or no response at all. For instance, Question: Calculate 58 * 22? resulted in 1038 instead of 1276, and Question: Calculate 54 / 18? resulted in 2 instead of 3. The model struggles more with larger numbers and operations involving large multipliers or divisors. This is evident in problems like Question: Calculate 94 * 79? and Question: Calculate 2703 / 51?.

Llama2-7b

The Llama2-7b model exhibits a variety of errors when solving arithmetic problems, particularly struggling with subtraction resulting in negative numbers, division, and multiplication. The model frequently provides incorrect positive counterparts for negative results (e.g., -53 instead of 53 for 29 - 82, and -76 instead of 76 for 13 - 89). It also demonstrates significant issues with division, often failing to generate any response (e.g., 2336 / 73 and 49 * 78), or providing incorrect results that are close but not accurate (e.g., 4212 / 81 resulting in 51 instead of 52). Multiplication errors are also common, with the model producing responses that are either entirely off (e.g., 56 * 61 yielding 13160 instead of 3416) or slightly incorrect (e.g., 97 * 60 resulting in 5920 instead of 5820). Additionally, the model sometimes fails to generate any response for simple addition problems (e.g., 51 + 2 and 36 + 15), indicating issues with handling even basic arithmetic operations.

Mistral-7b

The Mistral-7b model demonstrates a pattern of errors primarily characterized by minor miscalculations in multiplication and division, and a consistent failure to handle negative results correctly. The model often provides answers that are close to the correct result but not accurate, such as 3436 instead of 3416 for 56 * 61, and 5395 instead of 5415 for 95 * 57. This suggests a tendency to make small numerical errors in calculations. Additionally, the model frequently fails to recognize negative results, as seen in responses like 29 instead of -29 for 39 - 68, and 42 instead of -42 for 2 - 44. These types of errors indicate that while the model has a generally strong understanding of arithmetic operations, it struggles with precision and the correct handling of negative values, suggesting a need for further fine-tuning to improve accuracy in these areas.

Jumbled Arithmetic

The overall performance of the models on this task is very poor, except for the Mistral-7b model. The other models consistently provide very low accuracy on this task. The worst is the Cerebras-Btlm-3b, which usually generates unrelated responses with no sign of solving the task. We did not find any common patterns for the failures; the models failed to learn from the prompts and were confused between the arithmetic operations. For example, Calculate 37 $ 6. The answer is 43. But oftentimes, the models either performed subtraction or multiplication.

Sentiment Analysis

The most common failure for the sentiment analysis tasks is characterizing the "Neutral" labels. The models either predicted "Positive" or "negative" for these tasks. For example, Statement: The train arrives at 8 PM. It should have been a neutral sentiment task, but the models failed.

7.7 Conclusion

In summary, the evaluation of the language models on regular arithmetic tasks reveals significant differences in their ability to use demonstrations to enhance performance. Mistral-7b consistently outperforms the other models, demonstrating robust performance regardless of the number of demonstrations. Mamba-7b shows improvement with increased demonstrations, though its performance fluctuates. Mamba-2.8b follows a similar trend but declines at higher demonstration counts. Llama2-7b exhibits moderate variability, with the highest performance at a mid-range number of demonstrations. In contrast, Cerebras-btlm-3b shows minimal improvement, indicating limited benefit from additional context.

In the jumbled arithmetic tasks, where the operations were altered, Mistral-7b is the best in performance again. It shows a clear upward trend with increased demonstrations. Mamba-2.8b, Mamba-7b, and Cerebras-btlm-3b exhibit weak performance overall. Mamba-7b shows slight improvement at a certain point before declining again, while Llama2-7b responds positively to increased demonstrations despite some fluctuations. The introduction of Chain-of-Thought (CoT) prompting significantly enhances the performance of Mistral-7b. This highlights its strong ability to leverage this approach. Mamba-7b and Llama2-7b show moderate improvements with CoT prompting. On the other hand, Cerebras-btlm-3b and Mamba-2.8b display limited benefits, which indicates their challenges in adapting to this strategy.

In sentiment analysis tasks, all models benefit from demonstrations, particularly when true labels are provided. Mistral-7b has exceptional performance, maintaining high scores across all conditions. Mamba-2.8b and Mamba-7b show substantial improvements with true labels and moderate gains with CoT prompting. Cerebras-btlm-3b starts with lower performance but benefits significantly from CoT prompting. Llama2-7b was the weakest without demonstrations but shows significant improvement with CoT prompting and true labels. These findings tell the effectiveness of CoT prompting and accurate demonstrations in enhancing model performance across various tasks.

8 Investigating Machine Translation of Language Models

8.1 Overview the experiment

In this experiment, we aim to investigate the efficacy of machine translation through In-Context Learning (ICL) for converting English to Spanish. Using the WMT-14 dataset, which includes diverse language phenotypes and structures, our approach involves prompting pre-trained models with varying numbers of example translations, known as "shots." We explore zero-shot, two-shot, and few-shot scenarios to evaluate how well the

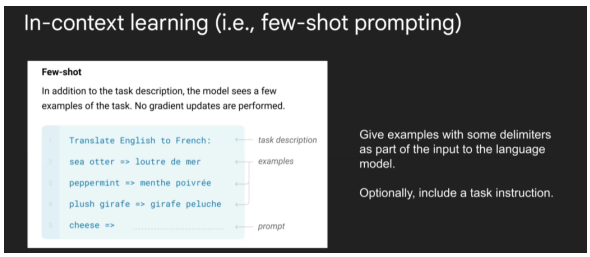

Figure 9: In Context Learning in Machine translation

models perform translations based on the provided context. We use five pre-trained models: MAMBA-2.8b, Btlm-3b, Mistral- 7B, Pythia-2.8B, and LLama-7B Testing involves translating sentences such as "The book is on the table" and "The flowers are blooming in the garden" to Spanish. We have used BLEU (Bilingual Evaluation Understudy) score to evaluate the translation accuracy. The approach is shown in figure 9.

8.2 What you proposed vs. what you accomplished

In the project proposal, we have added the English to Spanish translation and existing works have been mentioned. Here, we have done all the experiments which are mentioned in proposal in ICL machine translation techniques. We also added the few shots prompting and chain of thought in prompting.

8.3 Dataset

We used the WMT-14 (Bojar et al., 2014) training dataset, which is widely recognized in previous machine translation studies. This dataset was chosen due to its extensive range of sentence structures and vocabulary, providing a rich resource for English to Spanish translations across various contexts.

Several challenges arose in utilizing this dataset. Its large size and high average sentence length presented significant constraints for training in a low-resource environment. Consequently, we reduced the dataset to a sample of 500 English-Spanish sentence pairs. The variability in sentence contexts further complicated the task, making it difficult for models to generalize effectively. The description of dataset is given in Table 1.

In Table 2, we have added the original English sentence, equivalent original Spanish translated text, and experimental pre-trained large language model generated translation by prompting.

Table 1: Dataset Statistics

Statistic	**Value**
Training set size	953,621
Validation set size	3,000
Test set size	3,003
Sample size for testing	500
Average English sentence length	26.25
Average Spanish sentence length	23.452
No. of English Unique words	3004
No. of Spanish Unique words	3544

8.4 Data Pre-processing

The preprocessing steps included:

- **Tokenization**: Converting text into tokens suitable for the language model. Tokenization is essential because language models operate on numerical representations of text. This step breaks down sentences into smaller units (tokens) that the model can understand. Tokens can be words, subwords, or characters, depending on the tokenizer used. This process allows the model to process and learn from text data more efficiently.

- **Padding/Truncation**: Ensuring uniform input lengths up to 50 for batch processing. Padding and truncation are necessary to standardize the input sequences' lengths. Padding adds special tokens to shorter sequences to match the desired length, while truncation shortens longer sequences. This uniformity is crucial for batch processing, where multiple sequences are processed simultaneously. It ensures that all input data in a batch have the same shape, allowing for efficient computation and reducing memory overhead during training and inference.

8.5 Baselines

In our project, we evaluated the In-Context Learning (ICL) machine translation task across several new NLP benchmark models, including Mamba (Gu and Dao, 2023). Mamba has become prominent in the medium-scale parameter range (3-6 billion parameters), demonstrating superior performance over traditional Transformer models (Vaswani et al., 2017). To understand the ICL capabilities of various medium-scale language models, we compared Mamba-2.8B to models like

English Sentence	Original Spanish Sentence	LLM's Generated Translation
The sun is shining brightly.	El sol esta brillando intensamente.	El sol está brillantemente.

Table 2: In Context Learning in Machine translation (English to Spanish)

Bltm-3B, Mistral-7B, Pythia-2.8B, and LLama-7B. All models were sourced from Huggingface.

In the context of ICL tasks, we employed various prompting shots (i.e., 0, 2, 5, 8, 10) to compare the performance of these benchmark models. Notably, ICL tasks do not require parameter tuning because the models utilize pre-trained parameters to generate responses based on the examples provided in the prompt. This approach leverages the model's existing knowledge, enabling it to perform tasks without further adjustment of its internal parameters.

During preprocessing, we applied tokenization to convert text into tokens that the language models can process. We also used padding and truncation to standardize input lengths to a maximum of 50 tokens, ensuring uniformity.

- early_stopping= True,
- skip_special_tokens = True
- max_length= input_ids.shape[1]+50

The chosen baselines include various large language models with different architectures and sizes. The baselines used in this experiment are 3 different language models (LMs). All the models were taken from Huggingface. These models are:

- Mamba-2.8B
- Bltm-3b
- Mistral7B
- Pythia-2.8B
- LLama-7B

The detailed approach and results are written in the following subsection.

8.6 Our Approach

Our approach involves using different "shots" or examples within the prompts to guide the pre-trained model in translating English sentences into Spanish. The concept of "shots" in ICL refers to the number of example translations provided in the prompt before asking the model to generate a new translation. We explore the performance of the models under various shot conditions:

- Zero-Shot Learning: The model is prompted to translate sentences without any example translations provided.
- Few-Shot Learning: The model is provided with several example translations (typically between 2 to 10) before translating new sentences.

Pre-Trained Models: We compared five pre-trained language models (Mamba2.8B, BLTM 3B, Mistral 7B, Pythia-2.8B and LLama 7B) for our investigation:

8.7 Prompt Generation

The prompts consist of English sentences paired with their Spanish translations, followed by a new English sentence for which the model must generate the Spanish translation. The English and original Spanish translations, we have used WMT-14 datasets. For example, for 10 shots prompting, the prompt generation module is :

prompt = """

Translate the following sentences from English to Spanish:

1. The dog is running in the garden. El perro está corriendo en el jardín.
2. She is watching a movie. Ella está viendo una película.
3. The children are drawing pictures. Los niños están dibujando cuadros.
4. He is reading a newspaper. Él está leyendo un periódico..
5. They are traveling to Spain next month. Ellos van a viajar a España el próximo mes.
6. He is studying for the exam. Él está estudiando para el examen.

7. We are cooking dinner tonight. Nosotros estamos cocinando la cena esta noche.

8. They are playing in the park. Ellos están jugando en el parque.

9. She is going to the store. Ella va a la tienda.

10. The book is on the table. El libro está sobre la mesa.

Then we asked for another sentence to be translated into Spanish by pre-trained language models. Please translate the following sentence into Spanish:

The sun is shining brightly. The correct translation is:""""

Like this process, we have used various shot prompting in three different models.

8.8 Evaluation

To quantitatively evaluate the performance of the models, we use the BLEU score (Bilingual Evaluation Understudy), a standard metric for assessing the quality of machine-translated text compared to a reference translation. It measures the similarity between machine-translated text and reference translations. The BLEU score is a number between 0 and 1, with 0 indicating low quality and 1 indicating high quality. The BLEU algorithm compares consecutive phrases in the machine-translated text with the same phrases in the reference translation and counts the number of matches. A higher match degree indicates a higher degree of similarity with the reference translation and a higher score.

The BLEU score is calculated using the precision of n-grams in the candidate translation relative to the reference translations, along with a brevity penalty to account for short translations.

N-gram Precision First, we calculate the precision for each n-gram size (typically 1-gram, 2-gram, 3-gram, and 4-gram):

$$p_n = \frac{\sum_{C \in \text{Candidates}} \sum_{n\text{-gram} \in C} \text{Count}_{\text{clip}}(n\text{-gram})}{\sum_{C \in \text{Candidates}} \sum_{n\text{-gram} \in C} \text{Count}(n\text{-gram})} \tag{1}$$

where:

- $\text{Count}_{\text{clip}}(n\text{-gram})$ is the clipped count of n-grams in the candidate translation.
- $\text{Count}(n\text{-gram})$ is the total count of n-grams in the candidate translation.

Clipping means that the count for any n-gram in the candidate cannot exceed the count in the reference translations.

Brevity Penalty (BP) The brevity penalty is used to penalize overly short translations.

$$BP = \begin{cases} 1 & \text{if } c > r \\ e^{1-\frac{r}{c}} & \text{if } c \leq r \end{cases} \tag{2}$$

where:

- c is the length of the candidate translation.
- r is the effective reference length, which is typically the closest length of the reference translation(s) to the candidate translation length.

BLEU Score Finally, the BLEU score combines the n-gram precision and the brevity penalty:

$$\text{BLEU} = BP \cdot \exp\left(\sum_{n=1}^{N} w_n \log p_n\right) \tag{3}$$

where:

- w_n is the weight for the n-gram precision (commonly equal, e.g., $w_n = \frac{1}{N}$ for N-gram precisions).
- N is the maximum n-gram length considered.

8.9 Implementation Details:

We have completed our implementation in three steps : prompt generation, LLM's translation, and the evaluation of the performance of language models.

Library: We have used the following tasks for this library:

- `HuggingFace Transformers`
- `trl`
- `peft`
- `transformers`
- `datasets`
- `tqdm`

- `BitsAndBytesConfig`
- `torch`
- `pandas`
- `nltk.translate.bleu_score`

We have used the existing implementation of Mamba-2.8B, Mistral-7B, BTLM-3B, Pythia-2.8B, Llama-7B from huggingFace. The links are attached here :

- Mamba-2.8B: https://huggingface.co/state-spaces/mamba-2.8b-hf
- BTLM-3B: https://huggingface.co/cerebras/btlm-3b-8k-base
- Mistral-7B: https://huggingface.co/mistralai/Mistral-7B-v0.
- Pythia-2.8B: https://huggingface.co/EleutherAI/pythia-2.8b
- Llama-7B: https://huggingface.co/meta-llama/Llama-2-7b-chat-hf

Code availability: The datasets and codes are available here: https://github.com/ron352/In-Context-Learning-ICL-_MT.git.

Computing Resources: For this study, we used Google Colab Pro to leverage its enhanced computational capabilities, which are crucial for extensive training sessions. However, we faced several challenges inherent to the prolonged use of Colab Pro. The primary issue was the frequent disconnection of the runtime during extended training periods.

To address this, we had to terminate other active sessions multiple times and utilize higher-tier GPUs, such as the T4, A100, and L4 GPUs, to ensure sufficient computational power and efficiency. Additionally, we implemented JavaScript code within the browser console to keep the training session active, preventing it from timing out and thus ensuring the uninterrupted execution of our models. We couldn't run Mamba-7B because of limited memory.

8.10 Results

The LLama 7B model consistently performs the best across all prompting strategies, achieving a BLEU score of 0.9746 for 10-shot, 0.9240 for 5-shot, and 0.3757 for zero-shot prompting. This model's performance is particularly notable in high-shot scenarios, indicating its robustness in utilizing provided context. The Mamba 2.8B model also shows strong performance, especially in higher-shot prompts. It achieves a BLEU score of 0.8457 for 10-shot and 0.7416 for 5-shot, demonstrating its ability to leverage multiple examples effectively. However, its zero-shot score of 0.4210 suggests that while it can generalize well with context, it struggles more without any prompts.

BTLM 3B shows competitive performance with a 10-shot BLEU score of 0.8551, slightly higher than Mamba's. However, its zero-shot score of 0.2703 indicates significant challenges in translating without examples, highlighting its dependence on contextual information for accuracy. Mistral 7B performs similarly to Mamba 2.8B with a 10-shot BLEU score of 0.8457 but slightly lower in the 5-shot and zero-shot scenarios. This consistency suggests that Mistral can also effectively utilize multiple prompts but faces similar zero-shot translation challenges. Pythia 2.8B exhibits a consistent yet slightly lower performance across all prompting strategies compared to Mamba and Mistral. Its 10-shot BLEU score of 0.8329 and zero-shot score of 0.4169 indicate a moderate ability to use context but not as effectively as LLama or Mamba.

Overall, the LLama 7B model demonstrates superior performance across the board, particularly excelling in scenarios with more prompts. The Mamba 2.8B model shows strong performance with sufficient context but has room for improvement in zero-shot settings. Both BTLM and Mistral exhibit dependence on contextual examples for optimal performance, while Pythia consistently underperforms compared to the top models. These results underscore the importance of prompt design and contextual information in enhancing translation quality and model performance.

The model performance is attached in Figure 10.

Table 3: BLEU Scores for Different Models with Various Prompting Strategies

Model	10-shot	8-shot	5-shot	2-shot	0-shot
Mamba 2.8B	0.8457	0.8297	0.7416	0.64696	0.4210
BTLM 3B	0.8551	0.839000	0.7488	0.66433	0.2703
Mistral 7B	0.8457	0.81872	0.7346	0.630476	0.4016
Pythia 2.8B	0.8329	0.81981	0.7168	0.819480	0.4169
LLama 7B	0.974601	0.971324	0.924035	0.759362	0.37568

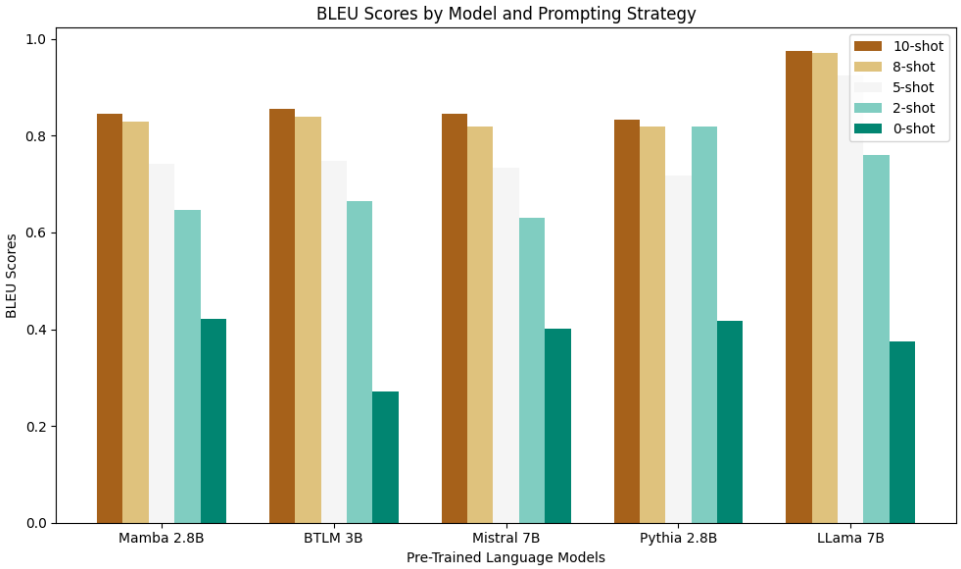

Figure 10: In context Learning in Translation work using prompting

8.11 Error Analysis

While we have used zero-shot prompting, the models fail at translating the sentences to Spanish. When we increase the prompt generation, it improves the result by translating it properly without the last words. In our prompts, we have used different contextual sentences so that the model can learn generalizability. The models fail at both semantic and syntactic commonalities in zero-shot prompting.

Types of Inputs the Baselines Fail The baseline models exhibit significant challenges with:

- **Zero-Shot Prompting**: All models fail to produce accurate translations without any examples, often retaining the original English text or producing incomplete translations.
- **Contextual Understanding**: The models frequently struggle to maintain the context of the sentence, leading to translations that are either too literal or miss the intended meaning.

For instance, the sentence ”The sun is shining brightly.” yielded the following translations under different prompting scenarios:

- **Original Sentence**: ”The sun is shining brightly.”
- **Expected Translation**: ”El sol está brillando intensamente.”

Mamba’s Responses:

- 10-shot: ”El sol está brillando brillantemente.”
- 8-shot: ”El sol brilla brillantemente.”
- 5-shot: ”El sol está brillando brillantemente.”
- 2-shot: ”El sol está brillando brillantemente.”
- 0-shot: ”El sol brilla brillantemente.”

BTLM’s Responses:

- 10-shot: ”El sol está brillando.”
- 8-shot: ”El sol está brillando.”

- 5-shot: ”El sol está brillando.”
- 2-shot: ”El sol está brillando fuerte.”
- 0-shot: ”The sun is shining brightly.”

Mistral's Responses:

- 10-shot: ”El sol está brillando brillantemente.”
- 8-shot: ”El sol está brillando brillantemente.”
- 5-shot: ”El sol está brillando brillantemente.”
- 2-shot: ”El sol está brillando brillantemente.”
- 0-shot: ”El sol brilla.”

Pythia's Responses:

- 10-shot: ”El sol brilla con fuerza.”
- 8-shot: ”El sol brilla con fuerza.”
- 5-shot: ”El sol brilla con brillo.”
- 2-shot: ”El sol brilla con fuerza.”
- 0-shot: ”The sol está brillando.”

Llama's Responses:

- 10-shot: ”El sol está brillando con fuerza.”
- 8-shot: ”El sol está brillando con fuerza.”
- 5-shot: ”La luz del sol brilla con fuerza.”
- 2-shot: ”La luz del sol brilla con fuerza.”
- 0-shot: ”El sol brilla con fuerza.”

Semantic and Syntactic Commonalities

- **Incomplete Translations**: Many translations miss key words or phrases, particularly at the end of sentences.
- **Literal Translations**: Some models translate word-by-word rather than capturing the overall meaning, leading to awkward or incorrect phrases.
- **Context Misunderstanding**: Models often fail to maintain the context provided in the examples, leading to semantically incorrect translations.
- **Inconsistent Usage**: Different models show varying levels of formality and vocabulary usage, often inconsistent with the provided context.

Hypotheses and Discussion The models likely struggle due to:

- **Limited Contextual Training**: Without sufficient context in the prompts, the models fail to generalize well.
- **Prompt Design**: The design and structure of the prompts significantly impact the model's ability to translate effectively. More contextual examples help but also highlight the need for better prompt engineering.

Chain of Thought Analysis Using the Chain of Thought (CoT) approach, we observe the models' progression in handling translation tasks:

- Initially, in zero-shot prompting, the models often fail to understand the context and produce incomplete or literal translations. For instance, Mamba's response ”El sol brilla brillantemente” fails to capture the exact meaning of ”intensamente” in the expected translation.
- As we increase the number of examples (shots), the models start to show improvements. For example, in the 10-shot scenario, Mamba translates the sentence to ”El sol está brillando brillantemente,” which, while not perfect, shows better alignment with the context.
- Despite improvements, some issues persist across models and shot scenarios. The CoT reveals that models like BTLM and Pythia struggle with maintaining the correct tense and formality, evident in their varying translations for similar prompts.
- By analyzing these chains of thought, we identify that better contextual understanding and prompt engineering can help models generalize translations more effectively, reducing errors related to semantic and syntactic inconsistencies.

8.12 Conclusion

In summary, CoT usually improves the performance of relatively larger language models than small models. It sometimes hurts the performance of smaller scale language models as the reasoning power is not well enough in these models. So, it becomes confused and performs poorly. From

our results, LLama-7B has surpassed the performance in all shots prompting except one compared to other LLMs. When we increased the prompts, the LLama-7B performed better because it can do reasoning better than other models. In contrast, Mistral-7B should perform alike Llama as it has the same number of parameters but in reality, it performed comparatively worse than Mamba 2.8B. An interesting aspect can be done by using downstream of prompting and showing the results of these models.

9 Mamba Based Model Updater for M-TGNNs

9.1 Dataset for Temporal Structured Data Learning

Edge event-based data format is a representation commonly used for temporal graph data, especially in the context of tasks like link prediction in temporal graphs. In this format, each interaction or event between nodes is recorded with a timestamp, capturing the temporal dynamics of the graph. Table 4 shows a sample event based temporal graph dataset and Figure 11 is the corresponding temporal graph.

src	dst	timestamp	feature
1	2	t_1	f(e_1)
1	3	t_2	f(e_2)
2	4	t_3	f(e_3)
3	4	t_4	f(e_4)
1	2	t_5	f(e_5)
1	5	t_6	f(e_6)
5	1	t_7	f(e_7)
6	3	t_8	f(e_8)
7	6	t_9	f(e_9)
4	7	t_{10}	f(e_{10})
2	8	t_{11}	f(e_{11})
9	10	t_{12}	f(e_{12})
8	11	t_{13}	f(e_{13})
12	6	t_{14}	f(e_{14})
11	12	t_{15}	f(e_{15})
6	9	t_{16}	f(e_{16})
10	1	t_{17}	f(e_{17})
12	9	t_{18}	f(e_{18})

Table 4: Edge Event Temporal Graph Dataset

For experimental analysis we have used the Wikipedia dataset introduced in Jodie (Kumar et al., 2019) for link prediction tasks. It is a moderately sized temporal benchmark graph database Wikipedia, consisting of 9,000 nodes and 157,000 edges. Each edge is associated with a timestamp and 172-dimensional feature.

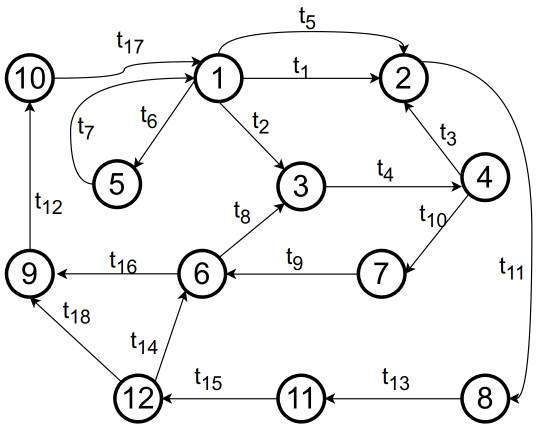

Figure 11: Temporal Graph from Table 4

9.2 Baseline for Temporal Data Learning

As baseline for temporal data learning, we have chosen TGN (Rossi et al., 2020) which is the state-of-the-art model for memory based temporal graph neural network. Details of the baseline are:

- Task: Link Prediction
- Model: TGN
- Framework: Deep Graph Library with PyTorch backend
- Memory Aggregator: Last_Message
- Memory Updater: LSTM Cell

9.3 Proposed Approach

Memory-based Temporal Graph Neural Networks (M-TGNN) are an extension of traditional Graph Neural Networks (GNNs) designed to handle temporal graph data. Temporal graphs are graphs where edges or node attributes change over time. M-TGNNs integrate memory mechanisms into GNN architectures to effectively capture temporal dynamics in such data.

The key idea behind M-TGNNs is to incorporate memory modules to store and update information about the past states of the graph. This allows the model to consider not only the current state of the graph but also its historical evolution, enabling better understanding and prediction of temporal graph dynamics.

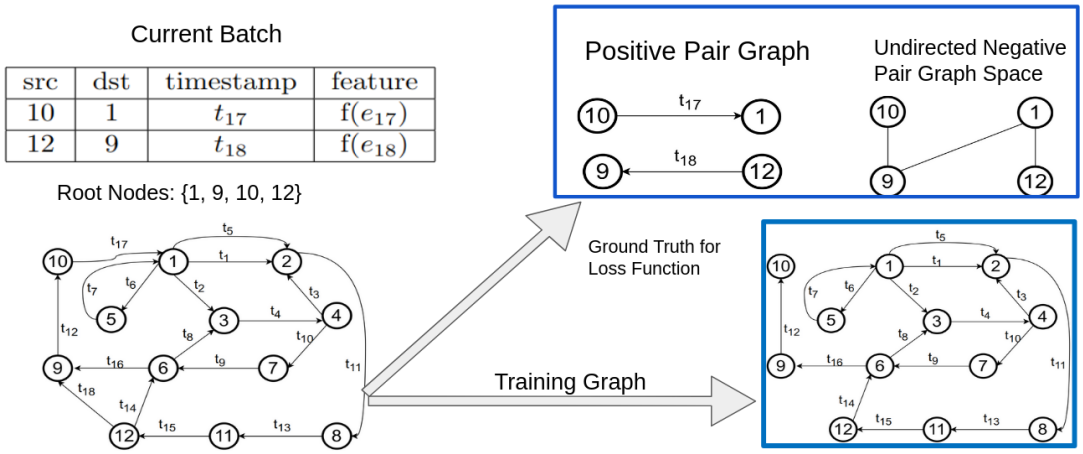

Figure 12: TGNN Training Procedure

M-TGNNs typically consist of several components:

- **Memory Mechanisms:** Memory modules store past information, such as node or edge memory vector, allowing the model to learn from historical patterns. These memories are updated over time as new information becomes available. It is typically a ike LSTM/GRU based sequential model.

- **Message Passing Module:** Message Passing Module generates message from all n-hop neighbors using node memory and feature vector from node and edge. Frameworks like DGL stores these messages in the receiving nodes mailbox.

- **Aggregator:** Aggregator combines all the messages in the mailbox of a node to generate a single node embedding vector, also known as node representation.

- **Prediction Layers:** These layers predict future states of the graph based on the learned representations and historical information.

M-TGNNs find applications in various domains where temporal graph data is prevalent, such as social networks, transportation networks, and financial markets. They have shown promising results in tasks like link prediction, node classification, and graph-level prediction in dynamic graph data. In this part of the project we are focusing on memory module of M-TGNNs for Link-Prediction as the downward prediction task.

Memory Module Design with Mamba TGNN processes events chronologically and uses the temporal graph constructed using the events prior to the current batch as the temporal training graph. If we consider batch size two for the sample graph database in Table 4, then the Figure 12 shows the training graph for the 9th batch. M-TGNNs uses memory vectors generated until the previous batch along with training graph messages for root nodes to estimate probability score for the edges present in the current batch (edge e_{17} and e_{18} in Figure 12). To make the dataset balanced, equal number of negative edges are sampled from the undirected negative pair graph sample space as show in the figure. After the prediction phase, memory vectors of the root nodes are updated using the positive pair graph constructed from the the events in the current batch.

Each edge in the positive pair-graph generates two messages. If $e_t = (u, v, t)$ is an edge where $u\&v$ are source node and destination nodes and t is timestamp, then message to node u from node v is:

$$m_{u \leftarrow v} = msgfn(f_u, f_v, f_{e_t}, timeEncode(t))$$

Similarly, message from node u to node v is:

$$m_{v \leftarrow u} = msgfn(f_v, f_u, f_{e_t}, timeEncode(t))$$

When we have a large batch size, each node will have multiple messages. For example, in the $Wikipedia$ dataset, for a batch size of 8000, some nodes have more than 650 incoming messages. In DGL, incoming messages are stored in the mailbox of the destination node. All of these messages have time information associated with it and form a sequence of messages.

$$M_u = m_1(t_1), m_2(t_2) \ldots m_n(t_n)$$

TGN (Rossi et al., 2020) uses an aggregator to combine all of these messages to a single vector (avg/last message/learnable function) and then use a GRUcell/LSTM passing previous node memory $S_u^(0)$ as hidden cell state and aggregated message as input to calculate the next cell state which is then used as the node's memory for next batches until updated. In this study we have replaced this memory update mechanism with Mamba block in two configurations.

- *Configuration 1: TGN-AGG-MAMBA* Replace LSTM cell with Mamba block.
- *Configuration 2: TGN-MAMBA* Replace Aggregator-LSTM with Mamba block.

For *Configuration 1*, we have prepared a two length sequence $S_u^{(0)} AggregatedMessage(u)$ and passed it to to mamba block; used the last output of the mamba block as updated node memory. For *Configuration 2*, we have prepended the message the mailbox of the node with the previous memory and passed this entire sequence to the Mamba block.

$$S_u^{(0)} m_1(t_1), m_2(t_2) \dots m_n(t_n)$$

The last vector of the output sequence is then used as the updated memory. *Configuration 1* is expected to be inferior to Configuration 2 in terms of accuracy as TGN-MAMBA will aggregate the messages considering the temporal dependence among the messages. The purpose of *Configuration 1* is to evaluate raw power of Mamba block with LSTM cell.

9.4 Empirical Analysis for Temporal Data Learning

Tools We have used DGL (Wang et al., 2019) with PyTorch kernel for implementation framework and Google colab Pro A100 GPU.

Experimental Results As baseline for evaluating temporal graph learning we have used TGN model with last_message aggregator as reducer and LSTM memory updater (TGN-AGG-MAMBA). We have two variant of the model to test against the baseline: TGN-AGG-MAMBA which uses Mamba block in place of LSTM cell as memory updater and TGN-MAMBA that replaces both aggregator and LSTM memory updater with Mamba block.

Average Precision We have recorded average precision on the Wikipedia Dataset for batch-size $32, 128, 256, 512, 1024, 2048, 4096, 8192$ and 16384. For the base model, as mentioned in the original paper, we have observed gradual fall in the average precision measure with the increase of the batch size. The best average precision was achieved for batch size 32.

We could not run the MAMBA models for batch size $32, 128, 256$ and 512 because of numerical instability. For batch size 1024 and larger, from Figure 13, we can see that, TGN-AGG-MAMBA and is also no exception here; in fact it performed slightly worse than base model for batch size 4096 and smaller. For batch size 8192 and 16384, TGN-AGG-MAMBA outperformed TGN-AGG-LSTM. But when we replaced last_message aggregator and LSTM memory updater with Mamba block on raw messages from DGL mailbox, average precision was significantly stable and outperformed other two model. For larger batch sizes, this performance improvement is significant.

Training Time For TGNNs, training time is mainly dominated by sampling time and loading time. so, in stead of focusing on the total time required for convergence, we have focused on per time. All three models exhibited a similar convergence trend; larger batch training required more epochs to converge. We have recorded epoch times per epoch to evaluate the computational burden introduced by my Mamba block.

Per epoch time for different batch sizes are shown in Figure 14. As expected, both configurations with Mamba takes a slightly more time compared to LSTM block, But the interesting point is, TGN-MAMBA considers perform aggregation respecting sequential intra-batch temporal dependency without significant overhead which resulted in stable average precision as shown in Figure 13.

9.5 Conclusion

Training Mamba from scratch is challenging and often subject to numerical instability. Architecturally, our experiments do not conclusively show that the Mamba block is more capable than LSTM. However, its ability to handle sequential dependencies in parallel computation makes it suitable for applications where we would not typically consider using an LSTM cell. Therefore, from the perspective of learning node representations and handling temporal data, we believe Mamba has

Figure 13: Average Precision

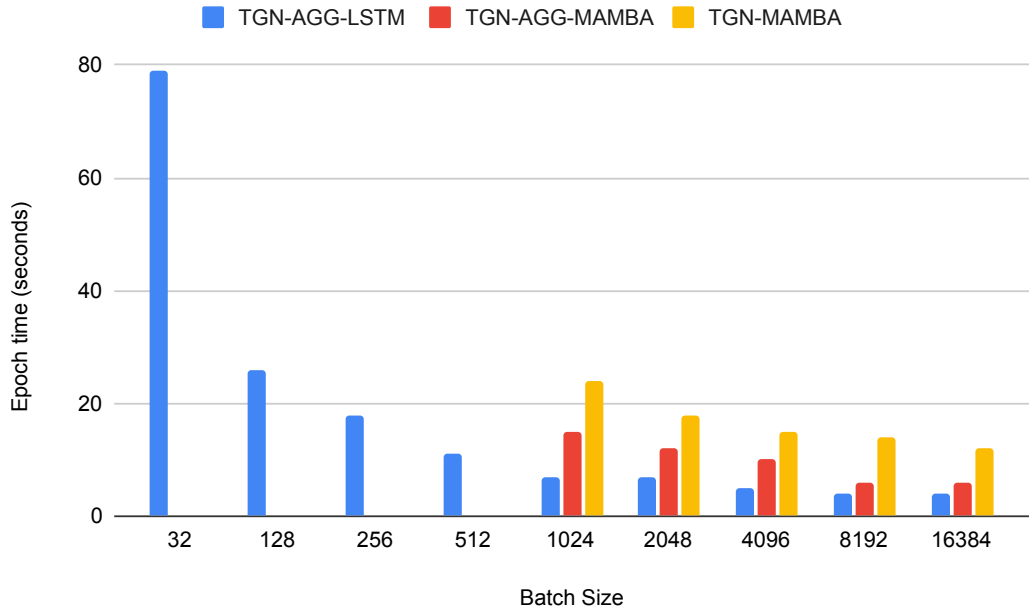

Figure 14: Epoch Time

promising potential.

Code Availability The code is available at https://github.com/cseduashraful/TGN-Mamba.git

10 Bias and Fairness Analysis

10.1 What you proposed vs. what you accomplished

In the project proposal, various evaluation metrics for measuring gender bias in language model from existing works have been mentioned and we proposed to evaluate Mamba for fairness using the metrics. We have used the word analogy test (Bolukbasi et al., 2016) metric to measure gender bias of Mamba language model (1.4b parameters) and compared the results with that of word2vec (Mikolov et al., 2013) since the metric was developed to measure bias in word embeddings.

10.2 Baselines

The following baselines are used to present a comparative performance of model regarding gender bias:

- Mamba-1.4b
- Gemma-2b
- zephyr-3b
- word2vec

10.3 Approach

10.3.1 Overview

To assess gender bias in word embeddings, we extract the last hidden layer output of the models, excluding word2vec which already contains the embeddings. Following the methodology outlined in (Bolukbasi et al., 2016), we establish a gender subspace g using a set of Definitional Word Pairs, comprising highly gendered words. Then, we evaluate a set of Professional Words, which are gender-neutral, by calculating their cosine similarity to the gender subspace g. This process illuminates the association of non-gendered words with gender, thereby highlighting any lingering gender bias in the embeddings. Below lists the definitional and professional words selected for bias evaluation in the Mamba model.

Definitional Word Pairs	*woman-man, girl-boy, she-he, mother-father, daughter-son, gal-guy, female-male, her-his, mary-john*
Professional Words	*nurse teacher writer engineer scientist manager driver banker musician artist chef filmmaker judge comedian inventor worker soldier journalist student athlete actor governor farmer person lawyer adventurer aide ambassador analyst astronaut astronomer biologist*

10.3.2 Gender sub-space *g*

Discerning the gender subspace is the first step. Here, we analyzed the difference vectors (embeddings) of definitional word pairs (e.g, *man-woman*) and calculated their principal components (PCs). We discovered a dominant direction that accounts for most of the variation in these vectors. We designate the leading PC, represented by the unit vector g as capturing the gender subspace. Further elaboration can be found in the codebase, including visualization of the principal components.

10.3.3 Measure Direct Bias

To quantify direct bias, we utilize the set of gender-neutral professional words, denoted as P, along with the gender direction determined previously, represented by g. We define the direct gender bias of an embedding (of a professional word) as follows:

$$bias\_score = \frac{1}{|P|} \sum_{w \in P} |cos(\vec{w}|g)|$$

This metric allows us to assess the gender bias of specific professional words. Initially, we present the gender bias of definitional words to assess their alignment towards male and female genders. Subsequently, we examine the gender bias of the professional words. The results are depicted in Figure 15 and Figure 16 respectively.

Further analyses were conducted on the gender bias present in the word embeddings utilized by the models (i.e., the embeddings employed as input to the model). These analyses aimed to illustrate the alterations in embeddings throughout training and their corresponding shifts in gender

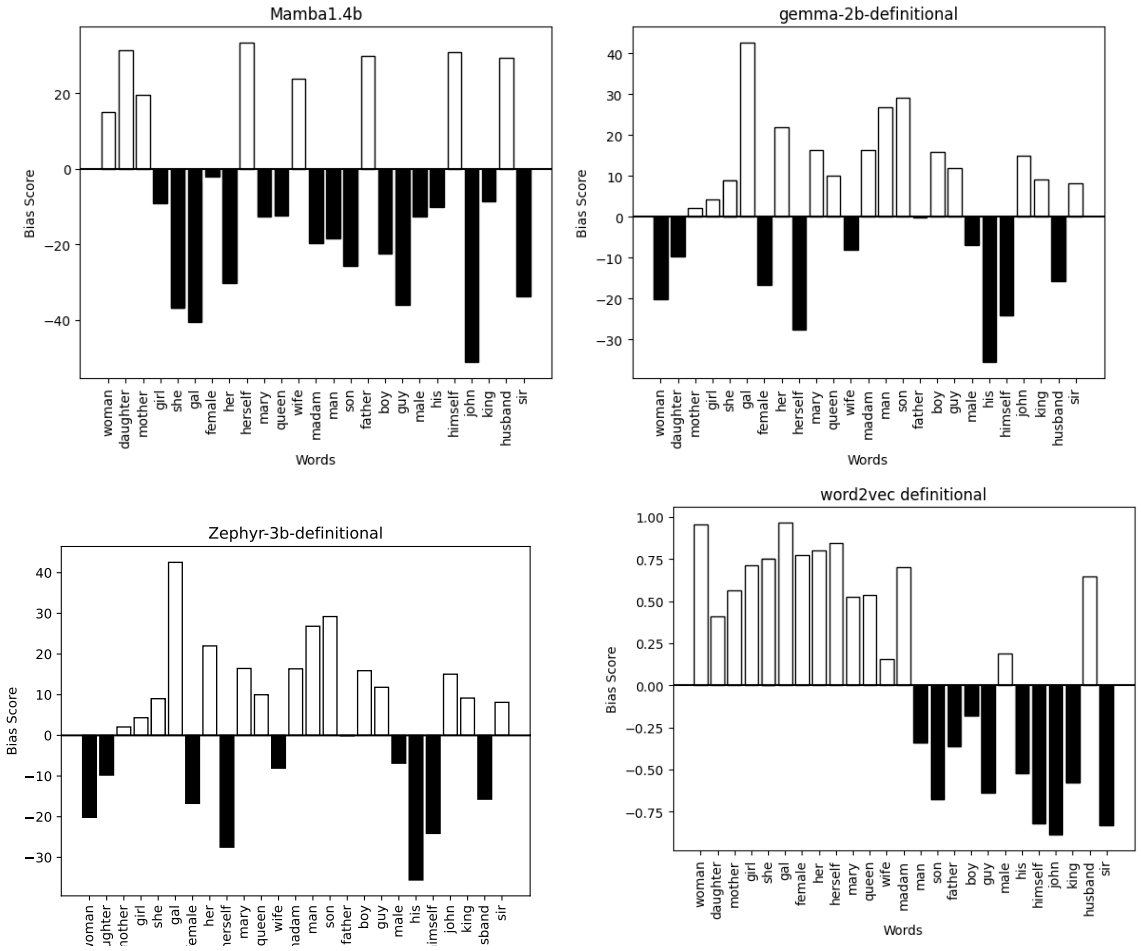

Figure 15: Gender bias of Definitional words. Female gender bias is represented as a positive number (white bar) and male gender bias is represented as a negative number (black bar).

bias. However, due to space constraint, these additional analyses are not presented here.

10.4 Dataset/Error Analysis

As we utilized word embeddings (for the word2vec model) or the last hidden layer representation of input tokens to gauge gender bias using the outlined procedure, there was no need for an explicit dataset in this application. Moreover, no explicit error analysis was necessary here, as there isn't a definitive correct output. However, the desired outcome is for gender-neutral occupational words not to exhibit stronger associations with gender space, as observed in the instances discussed. Further analysis on this point follows below.

10.5 Conclusion/Analysis

Word2vec illustrates how neutral words maintain gender information within their embeddings and exhibit bias towards particular genders for specific occupational terms. These biases fall into the category of stereotypical bias, which permeates societal norms and influences decision-making processes. However, the presence of gender bias in other language models (such as mamba-1.4b, gemma-2b, zephyr-3b) doesn't necessarily imply stereotypical bias; nevertheless, they still demonstrate bias towards certain genders (prevalently *male*), as evidenced by the bias scores for occupational terms. This finding underscores the need for more robust and pragmatic evaluation metrics to assess gender bias in word representations within LLMs. Additionally, gender assessment in various downstream tasks can provide insight into the diverse ramifications of gender bias. Furthermore, it's imperative to analyze and address other forms of bias (e.g., race, age, ethnicity), and gender bias should not be confined to binary genders.

Code availablity Gender bias evaluation code is publicly available at `https://github.`

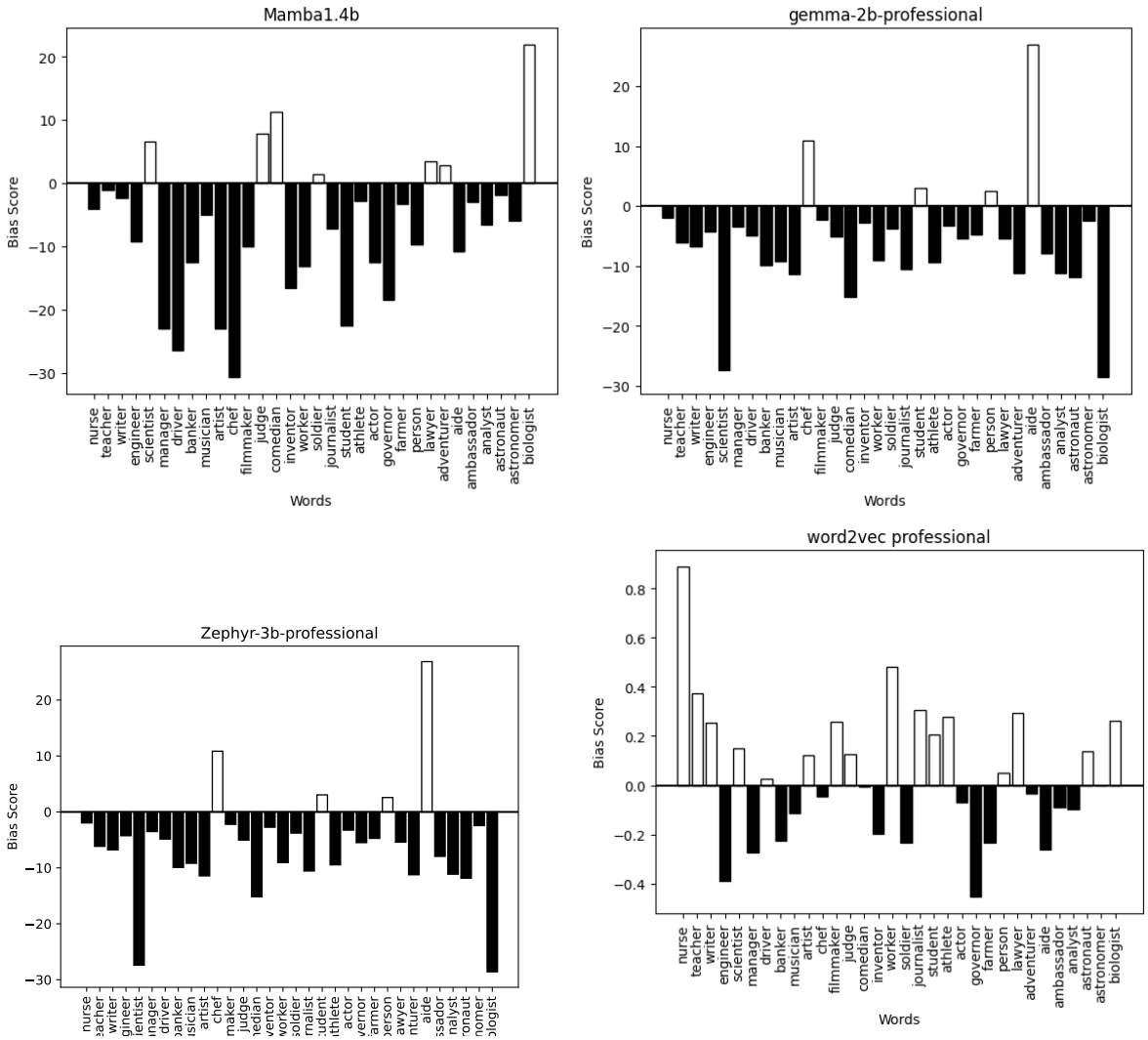

Figure 16: Gender bias of Professional words. Female gender bias is represented as a positive number (white bar) and male gender bias is represented as a negative number (black bar).

com/sz-ishita/mamba-gender-bias.

11 Evaluating Fairness of Pre-trained Mamba Models using Shapley Attribution

11.1 Overview

The experiment investigates the sensitivity of 11 language models (LMs) to demographic attributes by analyzing their responses to different profile statements. A dataset comprising 120 statements (covering general positive/negative qualities, academic/career successes/failures, and involvement in/victim of crime) and 189 profiles (combinations of sex, sexuality, age, and race) was created. Each model's sensitivity to these attributes was assessed by computing the cosine similarity between profile and statement embeddings, followed by SHAP (SHapley Additive exPlanations) attribution analysis to quantify the influence of each attribute. The results, spanning 11 LMs, revealed varying degrees of sensitivity to attributes such as sex, age, race, and sexuality, highlighting the nuanced biases inherent in these models. Results show existing pre-trained Mamba models show lower bias than transformer models. The code of the project can be found in https://github.com/Saad-Mahmud/CS685.

11.2 Dataset

The dataset consists of 120 statements and 189 profiles. The statements include general positive and negative statements, academic and career successes and failures, and involvement in or being a victim of crime. Examples of statements are:

- *This person is talented.*

- *This person is dishonest.*
- *This person graduated with honors.*
- *This person was involved in a fraud scheme.*

Profiles are created using combinations of attributes such as sex, sexuality, age, and race. Examples of profiles are:

- *He is a young Asian straight male.*
- *She is an old African American bisexual female.*

The data preprocessing involved generating profiles by combining attributes such as sex, sexuality, age, and race. The statements were generated using Chat-GPT4o.

11.3 Baselines

The baselines used in this experiment are 11 different language models (LMs). These models are:

- Mamba1.4B
- Phi1.5B
- Gemma2B
- RGemma2B
- Mamba2.8B
- Zephyr3B
- Mamba7B
- Llama7B
- Llama8B
- Mistral7B
- Gemma7B

All the models were taken from Huggingface.

11.4 Our approach

The approach involves calculating the cosine similarity of the final layer embeddings between profiles and statements. Let $\mathbf{e}_p$ and $\mathbf{e}_s$ denote the embeddings of a profile and a statement, respectively. The cosine similarity is given by:

$$\text{cosine\_similarity}(\mathbf{e}_p, \mathbf{e}_s) = \frac{\mathbf{e}_p \cdot \mathbf{e}_s}{\|\mathbf{e}_p\| \|\mathbf{e}_s\|} \quad (4)$$

SHAP (SHapley Additive exPlanations) values were then used to analyze the contribution of each attribute to the cosine similarity score. The SHAP value for a feature i is computed as follows:

$$\phi_i = \sum_{S \subseteq N \setminus \{i\}} C_S \left[f(S \cup \{i\}) - f(S) \right] \quad (5)$$

$$C_S = \frac{|S|!(|N| - |S| - 1)!}{|N|!} \quad (6)$$

where N is the set of all features, S is a subset of features, and f is the prediction function. The mean absolute SHAP values for each attribute were calculated to determine the sensitivity of each model. Let $\phi_{i,j,k}$ represent the SHAP value for attribute i in profile j, and statement k and n be the total number of profiles and m be the total number of statements. The mean absolute SHAP value for attribute i is given by:

$$\overline{|\phi_i|} = \frac{1}{n} \sum_{j=1}^{n} \left(\frac{1}{m} \sum_{k=1}^{m} |\phi_{i,j,k}| \right) \quad (7)$$

11.5 Results and Analysis

Figure 17 shows the sensitivity of various LMs to demographic attributes such as sex, age, race, and sexuality. The analysis reveals that Mamba-based models (Mamba1.4B, Mamba2.8B, and Mamba7B) generally show relatively low sensitivity across all attributes, indicating lower bias compared to other models. In particular, Mamba2.8B demonstrates the lowest sensitivity in most attributes, suggesting it handles demographic fairness better. Conversely, models like Zephyr3B and Gemma7B exhibit higher sensitivity, especially in sex and race attributes, indicating a higher level of bias. These results emphasize that existing pre-trained Mamba models do not pose additional safety issues and are relatively safe to use.

12 Contributions of group members

- Saaduddin Mahmud: Sections $1-4$ and Sections $5, 6, \& 11$.
- Md Ashraful Islam: Sections $1-4$ and Section 9.
- Sabrina Zaman Ishita: Sections $1-4$ and Sections $10, 11$.
- Amit Sarker: Sections $1-4$ and Section 7.

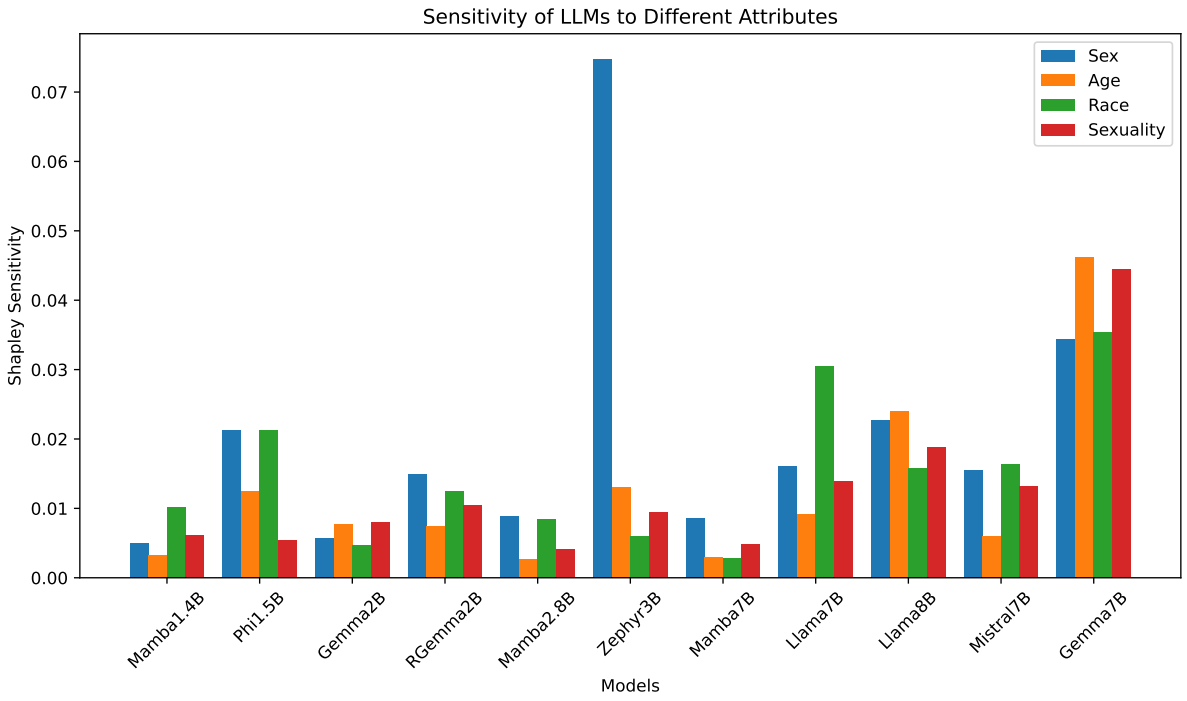

Figure 17: Sensitivity of LMs to Different Attributes

- Sarmistha Sarna Gomasta: Sections $1-4$ and Section 8.

13 Conclusion

Our experiments provide a comprehensive evaluation of Mamba models, highlighting their strengths and weaknesses across various tasks. Mamba models show promise in specific areas such as complex arithmetic and demographic fairness. However, there are clear areas for improvement, particularly in out-of-distribution (OOD) performance and simpler arithmetic tasks. In tasks involving in-context learning (ICL) and Chain-of-Thought (CoT) prompting, Mamba models did not exhibit a definitive advantage or disadvantage compared to transformers. The smaller parameter sizes of the models might not be sufficient to fully leverage CoT properties, leading to fluctuating performance. Additionally, the differences in training datasets for the pre-trained models could contribute to these inconsistencies, indicating a potential direction for in-depth future exploration.

In several experiments, such as those involving M-TGNNs and CoT fine-tuning, Mamba did not conclusively outperform competing models. However, Mamba models still offer the benefit of faster inference, making them potentially more favorable in scenarios where speed is crucial. This suggests that while Mamba models may not always lead in performance, their efficiency in inference could make them advantageous in specific applications.

Overall, while Mamba models demonstrate considerable promise in certain areas, ongoing research and development are necessary to address their limitations in OOD performance and enhance their capabilities in simpler arithmetic tasks. Further, we would suggest the community to train larger Mamba models as ICL and CoT properties might not be apparent in smaller models.

14 AI Disclosure

- Did you use any AI assistance to complete this proposal? If so, please also specify what AI you used.
  - Yes, we have used ChatGPT-4.

If you answered yes to the above question, please complete the following as well:

- If you used a large language model to assist you, please paste *all* of the prompts that you used below. Add a separate bullet for

each prompt, and specify which part of the proposal is associated with which prompt.

  - We have used prompts like write the formula of BLEU score, how to do prompt generation in ICL techniques.
  - Write a python code to read this text file, parse the questions, calculate the correct answer from the questions, parse the model response. (Section 7)
  - We got the following error: for i, line in range(0, len(lines)): TypeError: cannot unpack non-iterable int object. Describe the reasoning. (Section 7)
  - Describe the operations of strip() method in python. (Section 7)
  - How to use regex in python? (Section 7)
  - Here is the data, write a python code to generate a graph [graph type]. (Section 7
  - The code stopped working for this error: AttributeError: module 'backend_interagg' has no attribute 'FigureCanvas'. Did you mean: 'FigureCanvasAgg'? How to fix this? (Section 7)
  - rects1 = ax.bar(x - width, no_demos, width, label='No Demos', color='#8ecae6') rects2 = ax.bar(x, demos_true_label, width, label='Demos w/ True Label', color='#fb8500') rects3 = ax.bar(x + width, demos_random_label, width, label='Demos w/ Random Label', color='#023047')

    If we want to draw another bar, how do I change it? (Section 7)
  - How to change the gap between bars in this code? (Section 7)
  - Here is a graph that describes the [experiment details]. Please write a summary of the main findings for this graph. (Section 7)
  - Here are the examples of the problems the models failed to answer [examples]. Analyze which types of problems are the most difficult for each model. (Section 7)
  - Paraphrase the text and formalize it: "...." (Section 10)

- **Free response:** For each section or paragraph for which you used assistance, describe your overall experience with the AI. How helpful was it? Did it just directly give you a good output, or did you have to edit it? Was its output ever obviously wrong or irrelevant? Did you use it to generate new text, check your own ideas, or rewrite text?

  - The AI was helpful in description but it needed lots of fact checking and modification.
  - The AI was helpful in paraphrasing and formalizing some texts (that were written as dumping text in the report at the first place). But in some cases, it changed the meaning of some sentences that needed manual correction.

- Summarize the conclusions of each section: *dump of each section's conclusions. (Section 13)

- give me a brief literature review of topic **section name** with citations. (Section 4)

- Summarize the numeric results in a matplotlib bar graph. (Section 5, 6, 11)

- given experiment description write an overview. (Section 5, 6, 11)

- given written details of an experiment write math expression for that. (Section 5, 6, 11)

References

Blodgett, S. L., Barocas, S., Daumé III, H., and Wallach, H. (2020). Language (technology) is power: A critical survey of" bias" in nlp. *arXiv preprint arXiv:2005.14050*.

Bojar, O., Buck, C., Federmann, C., Haddow, B., Koehn, P., Leveling, J., Monz, C., Pecina, P., Post, M., Saint-Amand, H., Soricut, R., Specia, L., and Tamchyna, A. s. (2014). Findings of the 2014 workshop on statistical machine translation. In *Proceedings of the Ninth Workshop on Statistical Machine Translation*, pages 12–58, Baltimore, Maryland, USA. Association for Computational Linguistics.

Bolukbasi, T., Chang, K.-W., Zou, J. Y., Saligrama, V., and Kalai, A. T. (2016). Man is to computer programmer as woman is to homemaker? debiasing word embeddings. *arXiv preprint arXiv:1607.06520*.

Brown, T. B. et al. (2020). Language models are few-shot learners. *arXiv preprint arXiv:2005.14165*.

Caliskan, A., Bryson, J. J., and Narayanan, A. (2017). Semantics derived automatically from language corpora contain human-like biases. *Science*, 356(6334):183–186.

Chowdhery, A., Narang, S., Devlin, J., Bosma, M., Mishra, G., Roberts, A., Barham, P., Chung, H. W., Sutton, C., Gehrmann, S., et al. (2023). Palm: Scaling language modeling with pathways. *Journal of Machine Learning Research*, 24(240):1–113.

Gallegos, I. O., Rossi, R. A., Barrow, J., Tanjim, M. M., Kim, S., Dernoncourt, F., Yu, T., Zhang, R., and Ahmed, N. K. (2023). Bias and fairness in large language models: A survey. *arXiv preprint arXiv:2309.00770*.

Ghorbani, A. and Zou, J. (2019). Interpretation and fairness: A unified framework for evaluating bias in machine learning models. *arXiv preprint arXiv:1905.12888*.

Grazzi, R., Siems, J., Schrodi, S., Brox, T., and Hutter, F. (2024). Is mamba capable of in-context learning? *arXiv preprint arXiv:2402.03170*.

Gu, A. and Dao, T. (2023). Mamba: Linear-time sequence modeling with selective state spaces. *arXiv preprint arXiv:2312.00752*.

Gu, A., Goel, K., and Ré, C. (2021). Efficiently modeling long sequences with structured state spaces. *arXiv preprint arXiv:2111.00396*.

Guo, W. and Caliskan, A. (2021). Detecting emergent intersectional biases: Contextualized word embeddings contain a distribution of human-like biases. In *Proceedings of the 2021 AAAI/ACM Conference on AI, Ethics, and Society*, pages 122–133.

Hochreiter, S. and Schmidhuber, J. (1997). Long short-term memory. *Neural computation*, 9(8):1735–1780.

Hu, E. J. et al. (2021). Lora: Low-rank adaptation of large language models. *arXiv preprint arXiv:2106.09685*.

Jelassi, S., d'Ascoli, S., Domingo-Enrich, C., Wu, Y., Li, Y., and Charton, F. (2023). Length generalization in arithmetic transformers. *arXiv preprint arXiv:2306.15400*.

Jiang, A. Q., Sablayrolles, A., Roux, A., Mensch, A., Savary, B., Bamford, C., Chaplot, D. S., Casas, D. d. l., Hanna, E. B., Bressand, F., et al. (2024). Mixtral of experts. *arXiv preprint arXiv:2401.04088*.

Kim, S., Joo, S. J., Kim, D., Jang, J., Ye, S., Shin, J., and Seo, M. (2023). The cot collection: Improving zero-shot and few-shot learning of language models via chain-of-thought fine-tuning. *ArXiv*, abs/2305.14045.

Kumar, S., Zhang, X., and Leskovec, J. (2019). Predicting dynamic embedding trajectory in temporal interaction networks. In *Proceedings of the 25th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM.

Kurita, K., Vyas, N., Pareek, A., Black, A. W., and Tsvetkov, Y. (2019). Measuring bias in contextualized word representations. *arXiv preprint arXiv:1906.07337*.

LeCun, Y., Bengio, Y., et al. (1995). Convolutional networks for images, speech, and time series. *The handbook of brain theory and neural networks*, 3361(10):1995.

Li, Y., Du, M., Song, R., Wang, X., and Wang, Y. (2023). A survey on fairness in large language models. *arXiv preprint arXiv:2308.10149*.

Liang, P., Bommasani, R., Lee, T., Tsipras, D., Soylu, D., Yasunaga, M., Zhang, Y., Narayanan, D., Wu, Y., Kumar, A., et al. (2022). Holistic evaluation of language models. *arXiv preprint arXiv:2211.09110*.

Matthews, A., Grasso, I., Mahoney, C., Chen, Y., Wali, E., Middleton, T., Njie, M., and Matthews, J. (2021). Gender bias in natural language processing across human languages. In *Proceedings of the First Workshop on Trustworthy Natural Language Processing*, pages 45–54.

May, C., Wang, A., Bordia, S., Bowman, S. R., and Rudinger, R. (2019). On measuring social biases in sentence encoders. *arXiv preprint arXiv:1903.10561*.

Mikolov, T., Chen, K., Corrado, G., and Dean, J. (2013). Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*.

Nadeem, M., Bethke, A., and Reddy, S. (2020). Stereoset: Measuring stereotypical bias in pretrained language models. *arXiv preprint arXiv:2004.09456*.

Park, J., Park, J., Xiong, Z., Lee, N., Cho, J., Oymak, S., Lee, K., and Papailiopoulos, D. (2024). Can mamba learn how to learn? a comparative study on in-context learning tasks. *arXiv preprint arXiv:2402.04248*.

Pineda, F. J. (1987). Generalization of back-propagation to recurrent neural networks. *Physical review letters*, 59(19):2229.

Robinson, N. R., Ogayo, P., Mortensen, D. R., and Neubig, G. (2023). Chatgpt mt: Competitive for high-(but not low-) resource languages. *arXiv preprint arXiv:2309.07423*.

Rossi, E., Chamberlain, B., Frasca, F., Eynard, D., Monti, F., and Bronstein, M. (2020). Temporal graph networks for deep learning on dynamic graphs. *arXiv preprint arXiv:2006.10637*.

Silva, A., Tambwekar, P., and Gombolay, M. (2021). Towards a comprehensive understanding and accurate evaluation of societal biases in pre-trained transformers. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 2383–2389.

Talat, Z., Névéol, A., Biderman, S., Clinciu, M., Dey, M., Longpre, S., Luccioni, S., Masoud, M., Mitchell, M., Radev, D., Sharma, S., Subramonian, A., Tae, J., Tan, S., Tunuguntla, D., and Van Der Wal, O. (2022). You reap what you sow: On the challenges of bias evaluation under multilingual settings. In *Proceedings of BigScience Episode #5 – Workshop on Challenges & Perspectives in Creating Large Language Models*, pages 26–41, virtual+Dublin. Association for Computational Linguistics.

Touvron, H., Martin, L., Stone, K., Albert, P., Almahairi, A., Babaei, Y., Bashlykov, N., Batra, S., Bhargava, P., Bhosale, S., et al. (2023). Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., and Polosukhin, I. (2017). Attention is all you need. *Advances in neural information processing systems*, 30.

Wang, M., Zheng, D., Ye, Z., Gan, Q., Li, M., Song, X., Zhou, J., Ma, C., Yu, L., Gai, Y., et al. (2019). Deep graph library: A graph-centric, highly-performant package for graph neural networks. *arXiv preprint arXiv:1909.01315*.

Wang, X., Lyu, D., Li, M., Xia, Y., Yang, Q., Wang, X., Wang, X., Cui, P., Yang, Y., Sun, B., et al. (2021). Apan: Asynchronous propagation attention network for real-time temporal graph embedding. In *Proceedings of the 2021 international conference on management of data*, pages 2628–2638.

Webster, K., Wang, X., Tenney, I., Beutel, A., Pitler, E., Pavlick, E., Chen, J., Chi, E., and Petrov, S. (2020). Measuring and reducing gendered correlations in pre-trained models. *arXiv preprint arXiv:2010.06032*.

Wei, J. et al. (2022). Chain of thought prompting elicits reasoning in large language models. *arXiv preprint arXiv:2201.11903*.

Zelikman, E., Wu, J., and Goodman, N. (2023). Star: Bootstrapping reasoning with rationales. *arXiv preprint arXiv:2305.14045*.

Zhang, M. and Toral, A. (2019). The effect of translationese in machine translation test sets. *arXiv preprint arXiv:1906.08069*.

Zhao, J., Wang, T., Yatskar, M., Cotterell, R., Ordonez, V., and Chang, K.-W. (2019). Gender bias in contextualized word embeddings. *arXiv preprint arXiv:1904.03310*.

Zhong, Y., Sheng, G., Qin, T., Wang, M., Gan, Q., and Wu, C. (2023). Gnnflow: A distributed framework for continuous temporal gnn learning on dynamic graphs. *arXiv preprint arXiv:2311.17410*.

Zhou, H., Zheng, D., Nisa, I., Ioannidis, V., Song, X., and Karypis, G. (2022). Tgl: A general framework for temporal gnn training on billion-scale graphs. *arXiv preprint arXiv:2203.14883*.

Zhou, H., Zheng, D., Song, X., Karypis, G., and Prasanna, V. (2023). Disttgl: Distributed memory-based temporal graph neural network training. In *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, pages 1–12.