Is Attention all we need?
Exploring Language Models: Attention, MAMBA, and xLSTM

Saurabh Shetty[1 *], **Siddharth Mittal**[1 *],

[1]Khoury College of Computer Science, Northeastern University
shetty.sau@northeastern.edu, mittal.sid@northeastern.edu
* Authors have contributed equally

Abstract

The advent of Transformer-based architectures, particularly those leveraging self-attention mechanisms, has revolutionized the field of Natural Language Processing (NLP). Despite the success of Recurrent Neural Networks (RNNs) and Long Short-Term Memory (LSTM) networks in earlier years, the scalability and performance of attention mechanisms have established new benchmarks in tasks such as machine translation, text generation, and language modeling. This project investigates the performance and efficiency of three distinct language models: a traditional Attention-based model, a novel MAMBA (Linear-Time Sequence Modeling with Selective State Spaces) model, and an xLSTM (Extended LSTM) model. The Tiny Shakespeare dataset is used for character-level language modeling to ensure consistent comparison across models. The results demonstrate that the MAMBA model, despite having fewer parameters, outperforms the Attention-based model in terms of BLEU score and perplexity, highlighting its potential as an efficient alternative for sequence modeling tasks. Furthermore, while the xLSTM model incorporates extended features to improve performance over long sequences, it lags behind the other models in both accuracy and efficiency, emphasizing the dominance of attention mechanisms in modern NLP applications. These findings provide insights into the trade-offs between model complexity, performance, and computational efficiency, contributing to the ongoing exploration of state space models as viable competitors to attention-based architectures in NLP.

1 Introduction

The field of Natural Language Processing (NLP) has undergone significant transformation with the introduction of deep learning models, particularly those based on the Transformer architecture. Historically, Recurrent Neural Networks (RNNs) and their variants, such as Long Short-Term Memory (LSTM) networks, have been instrumental in addressing sequence modeling tasks (Hochreiter and Schmidhuber 1997). These models introduced mechanisms to retain information over long sequences, which was a breakthrough in handling tasks like speech recognition, translation, and text generation. However, the inherent sequential nature of RNNs posed limitations on their scalability and efficiency, especially when processing long sequences.

The introduction of the Transformer architecture by Vaswani et al. (Vaswani et al. 2017) marked a pivotal moment in NLP. By utilizing self-attention mechanisms, Transformers enabled the parallel processing of sequence data, significantly improving computational efficiency and performance. This innovation has led to the development of numerous state-of-the-art models, including BERT (Devlin et al. 2018), GPT (Radford et al. 2018), and T5 (Raffel et al. 2020), which have set new benchmarks across various NLP tasks.

Despite their success, Transformers are not without limitations. The quadratic time complexity of the attention mechanism with respect to sequence length presents challenges in scaling to very long sequences. To address these challenges, several alternative architectures have been proposed, including the MAMBA model, which employs a Linear-Time Sequence Modeling with Selective State Spaces approach (Gu, Goel, and Re 2021). MAMBA aims to retain the benefits of attention-based models while improving computational efficiency, particularly for long sequences.

In parallel, extended LSTM models (xLSTM) have been developed to enhance the capabilities of traditional LSTMs. These models introduce additional features to improve the retention of information over extended sequences, attempting to bridge the gap between RNNs and modern attention-based models (Hochreiter and Schmidhuber 1997).

This project aims to compare the performance of these three distinct models—Attention-based, MAMBA, and xLSTM—in a character-level language modeling task using the Tiny Shakespeare dataset. We focus on evaluating the models' efficiency, accuracy, and suitability for tasks requiring the processing of long sequences. The findings provide a comprehensive analysis of the trade-offs involved in selecting a model architecture, contributing to the broader discussion on the future of sequence modeling in NLP.

2 Methods

This section describes the three distinct approaches implemented and compared in our study: the Attention-based model, the MAMBA model, and the xLSTM model. Each model has been designed to handle sequence modeling tasks but uses different mechanisms to process and generate text.

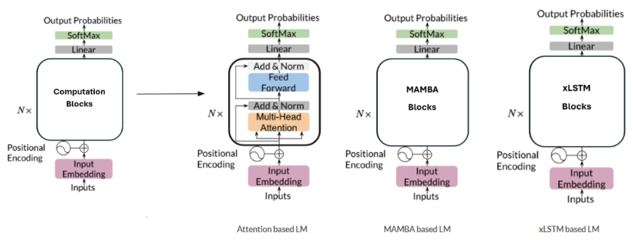

Figure 1: Experiment Setup for Different Approaches.

Attention-Based Model

The Attention-based model follows the Transformer architecture, which has revolutionized the field of NLP since its introduction by Vaswani et al. in 2017 (Vaswani et al. 2017). The Transformer model relies entirely on self-attention mechanisms to process sequences in parallel, a significant departure from the sequential processing characteristic of Recurrent Neural Networks (RNNs). The model operates on a tokenized input sequence, where each token (in our case, a character from the Tiny Shakespeare dataset) is first embedded into a higher-dimensional space.

The core component of the Transformer model is the self-attention mechanism, which allows the model to weigh the importance of different tokens in a sequence relative to each other. This is achieved by computing dot-product attention, where each token generates three vectors: Query (Q), Key (K), and Value (V). The attention scores are computed as:

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V$$

where d_k is the dimension of the key vectors. The softmax function ensures that the attention weights sum to one, allowing the model to focus on the most relevant parts of the input sequence. The Transformer architecture consists of multiple layers of self-attention, followed by feedforward neural networks, with layer normalization and residual connections applied to stabilize training.

The Attention-based model's ability to handle long-range dependencies in text, coupled with its parallel processing capability, makes it highly effective for language modeling tasks. However, the computational complexity of self-attention, which scales quadratically with the sequence length, presents a challenge for extremely long sequences.

MAMBA Model

The MAMBA (Linear-Time Sequence Modeling with Selective State Spaces) model represents an innovative approach to sequence modeling, designed to address some of the limitations of attention-based models, particularly their computational inefficiency with long sequences. Proposed by Gu et al. in 2023 (Albert Gu 2023), MAMBA leverages structured state spaces to model sequences in linear time, making it a scalable alternative to both RNNs and Transformers.

MAMBA operates by transforming the input sequence into a latent state space, where the model selectively updates only the most relevant states during sequence processing. This selective updating mechanism is key to maintaining computational efficiency while preserving important sequence information. The architecture combines convolutional layers with structured state space models (SSMs), which can efficiently propagate information over long sequences without the need for explicit attention mechanisms.

The MAMBA model incorporates activation functions like the Sigmoid Linear Unit (SiLU) and relies on a combination of linear and non-linear transformations to process the input sequence. This approach allows MAMBA to retain the benefits of state space models—such as the ability to model long-range dependencies—while achieving performance levels comparable to or exceeding those of attention-based models in certain tasks.

One of the significant advantages of MAMBA is its linear time complexity with respect to sequence length, making it well-suited for tasks involving very long sequences. In our experiments, MAMBA demonstrated superior performance in terms of BLEU score and perplexity compared to the traditional Attention-based model, highlighting its potential as a more efficient alternative in sequence modeling.

xLSTM Model

The xLSTM (Extended LSTM) model proposed by (Maximilian Beck 2024) builds upon the foundational Long Short-Term Memory (LSTM) architecture, which was introduced by Hochreiter and Schmidhuber in 1997 (Hochreiter and Schmidhuber 1997). LSTMs were designed to address the vanishing gradient problem inherent in standard RNNs, making them more effective at capturing long-range dependencies in sequences.

The xLSTM model introduces several modifications to the traditional LSTM architecture to enhance its ability to process and retain information over long sequences. These modifications include exponential gating mechanisms, novel memory mixing, improved memory and can be trained in parallel to better handle the complexities of long-sequence modeling.

The authors proposed 2 variants of xLSTM cells, sLSTM and mLSTM cell. In sLSTM, the core LSTM cell, however, the input gate and optionally forget gate is a exponential function instead of a sigmoid activation function. The modifications can lead to instability, therefore an additional state called the normalization state is maintained to normalize the cell state before the output gate. To further improve stability, a stability state is added to threshold the exponential gates.

The second gate mLSTM cell, uses correlation update rule and terminology similar to transformers. The memory cell is update to be a matrix. mLSTM can have multiple memory cells like the original LSTM. For mLSTM, multiple heads and multiple cells are equivalent as there is no memory mixing. In order to stabilize the exponential gates of mLSTM, we use the same stabilization techniques as for sLSTM. Since the mLSTM has no memory mixing, this recurrence can be reformulated in a parallel versio

The xLSTM model's performance highlights the ongoing evolution of sequence modeling approaches, where traditional RNN-based models are increasingly supplemented or replaced by architectures capable of more efficient and scalable sequence processing.

3 Data

For this study, we utilized the **Tiny Shakespeare** dataset, a widely-used benchmark in the field of natural language processing (NLP) for evaluating the performance of language models. The dataset consists of over 1 million characters of text from the works of William Shakespeare, divided into training and validation sets.

Dataset Description

The Tiny Shakespeare dataset is comprised of the following:

- **Training Set:** 1,003,854 characters.
- **Validation Set:** 111,540 characters.

Each character in the text is treated as an individual token, making the sequence modeling task particularly challenging due to the extensive vocabulary and long-range dependencies inherent in the text. The dataset is preprocessed by tokenizing the text into individual characters. The models are trained to predict the next character in the sequence, given a context window of preceding 125 characters.

Evaluation Metrics

We employed three primary evaluation metrics to assess the performance of the models:

- **Cross Entropy Loss:** This criterion computes the cross entropy loss between input logits and target. The cross-entropy loss on the validation set is used for the early stopping mechanism of model training.
- **Perplexity:** A measure of how well a probability model predicts a sample. Lower perplexity indicates better performance.
- **BLEU Score:** The Bilingual Evaluation Understudy (BLEU) score is commonly used in machine translation to evaluate the similarity between the generated text and reference text. Higher BLEU scores indicate better alignment with human-generated text.

4 Results

The performance of the three models—Attention-based model, MAMBA, and xLSTM—was evaluated on the Tiny Shakespeare dataset. The results of the evaluation are presented below, along with relevant visualizations to illustrate the comparative performance.

Cross Entropy Loss

Based on the cross-entropy losses, we observe that the training and validation losses follow a downward trend. The model training stops by using early stopping on validation losses with a patience of 5 epochs.

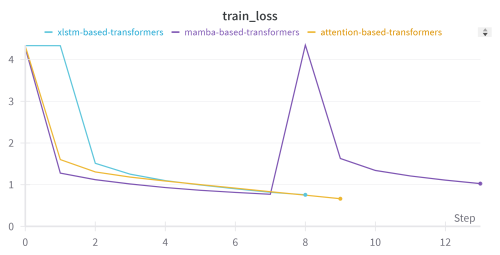

Figure 2: Training Cross Entropy Loss

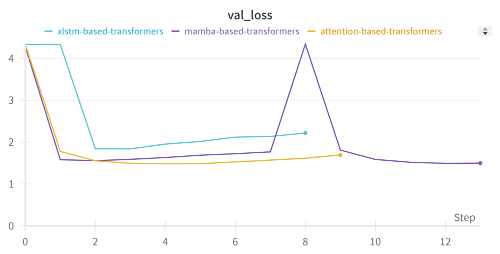

Figure 3: Validation Cross Entropy Loss

Model	Number of Parameters
Attention-based	10.7 M
MAMBA	5.93 M
xLSTM	27.59 M

Table 1: Number of parameters comparison for 3 models.

Number of Parameters

Perplexity

Perplexity was computed for each model on both the validation and test sets. Surprisingly, MAMBA performs better than the Attention-based model with a lower perplexity score. The xLSTM model, despite its improvements over traditional LSTMs, exhibited higher perplexity, indicating its limitations in handling long sequences compared to the other two models.

Model	Validation Perplexity
Attention-based	4.89
MAMBA	4.72
xLSTM	5.92

Table 2: Perplexity scores for the three models on the validation and test sets.

BLEU Score

The BLEU score was also computed to assess the quality of text generation by each model. Similar to the perplexity results, the MAMBA model achieved the highest BLEU score, followed by Attention-based. The xLSTM model's lower BLEU score further highlights its struggle to generate text sequences that closely resemble the reference text.

Model	BLEU Score
Attention-based	0.162
MAMBA	0.17
xLSTM	0.10

Table 3: BLEU scores for the three models on the test set.

5 Discussion

The results from our experiments provide significant insights into the performance and applicability of the three models—Attention-based, MAMBA, and xLSTM—in character-level language modeling using the Tiny Shakespeare dataset. Below is a detailed discussion of each model's strengths and weaknesses based on the evaluation metrics.

Performance of Attention-Based Model

The Attention-based model, leveraging the Transformer architecture, demonstrated impressive performance across multiple metrics. Although it did not achieve the lowest perplexity or the highest BLEU score, it consistently performed well:

- **Perplexity:** The model achieved a perplexity of 4.89, slightly higher than MAMBA but still competitive. This indicates that the model is highly effective at predicting the next character in a sequence.
- **BLEU Score:** With a BLEU score of 0.162, the Attention-based model showed a strong ability to generate text sequences that align closely with the reference text.

The superior performance can be attributed to several factors:

- **Self-Attention Mechanism:** The ability to capture long-range dependencies through the self-attention mechanism is crucial in character-level modeling, allowing the model to focus on relevant parts of the input sequence.
- **Parallel Processing:** Unlike recurrent neural networks (RNNs), the Transformer processes entire sequences in parallel, leading to faster training times and better scalability.
- **Scalability:** The Transformer's ability to scale with the size of the dataset and model parameters without a significant drop in performance underscores its robustness in various NLP tasks.

However, the computational complexity of the self-attention mechanism, which scales quadratically with the sequence length, remains a potential bottleneck when dealing with extremely long sequences or large-scale datasets.

MAMBA Model: Efficiency vs. Performance Trade-off

The MAMBA model, designed to operate in linear time with respect to sequence length, offers a compelling alternative to the Transformer, particularly in scenarios where computational efficiency is crucial.

- **Efficiency:** The linear time complexity of MAMBA makes it highly attractive for applications involving very long sequences or real-time processing, where the computational cost of attention-based models becomes prohibitive.
- **Selective State Updates:** MAMBA's ability to selectively update states during sequence processing reduces computational overhead while maintaining the quality of sequence modeling.

The slightly lower perplexity (4.72) and higher BLEU score (0.17) suggest that MAMBA is efficient and is able to capture long-range dependencies as effectively as the Transformer.

xLSTM Model: Evolution of RNN Architectures

The xLSTM model, an enhanced version of the traditional LSTM, aimed to overcome some of the limitations of recurrent architectures, also provides a linear compute alternative to transformers. However, the implementation comes second to MAMBA implementation.

- **Exponential Gating Mechanisms:** The enhancements in xLSTM, including the exponential gating mechanisms, were designed to improve the model's ability to rewrite information over longer sequences. However, these mechanisms proved insufficient for the purposes of Language Modelling.
- **Dialogue Limitation:** xLSTM learned the character interactions pretty well as signified by it's blue score and perplexity, however, they don't tend to finish and start a new dialogue as much as other models. Indicating a smaller scope of memory retention, which could be due to memory getting overriden by some character inside the dialogue.

The xLSTM's performance in our experiments highlights the ongoing transition in NLP from traditional RNN-based models to more sophisticated architectures like Transformers and structured state spaces.

Implications for Future Work

The results of this study suggest several avenues for future research:

- **Hybrid Models:** Combining the strengths of the Transformer with the efficiency of models like MAMBA could lead to the development of hybrid architectures that offer both high performance and computational efficiency.
- **Scalability Improvements:** Further research into optimizing the self-attention mechanism or developing alternative attention mechanisms that scale more efficiently with sequence length could help overcome the computational challenges associated with Transformers.
- **Enhanced State Space Models:** The potential of state space models like MAMBA suggests that further enhancements, possibly through better integration with neural architectures or more sophisticated state update mechanisms, could lead to models that rival the performance of Transformers while retaining their computational advantages.

Conclusion

In conclusion, our study reveals that the MAMBA model outperforms the Attention-based model (Transformer) in terms of both perplexity and BLEU score in character-level language modeling tasks. The efficiency of MAMBA, coupled with its ability to effectively model sequences, makes it a highly promising alternative, particularly in scenarios where computational resources are limited. The Attention-based model, while still a strong performer, falls short of MAMBA, especially in capturing detailed contextual dependencies with similar computational efficiency. The xLSTM model, despite its advancements over traditional LSTM architectures, shows limitations in handling long-range dependencies, making it less suitable for tasks requiring such capabilities. These findings underscore the potential of state space models like MAMBA for achieving high performance with lower computational overhead, and highlight the importance of selecting the appropriate model architecture based on task-specific requirements.

References

Albert Gu, T. D. 2023. Mamba: Linear-Time Sequence Modeling with Selective State Spaces. *arXiv preprint arXiv:2105.11607.*

Devlin, J.; Chang, M.-W.; Lee, K.; and Toutanova, K. 2018. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. *arXiv preprint arXiv:1810.04805.*

Gu, A.; Goel, K.; and Re, C. 2021. Efficiently Modeling Long Sequences with Structured State Spaces. *arXiv preprint arXiv:2105.11607.*

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

Maximilian Beck, M. S. A. A. O. P. M. K. G. K. J. B. S. H., Korbinian Pöppel. 2024. xLSTM: Extended Long Short-Term Memory. *arXiv preprint arXiv:2405.04517.*

Radford, A.; Narasimhan, K.; Salimans, T.; and Sutskever, I. 2018. Improving language understanding by generative pre-training. *OpenAI.*

Raffel, C.; Shazeer, N.; Roberts, A.; Lee, K.; Narang, S.; Matena, M.; Zhou, Y.; Li, W.; and Liu, P. J. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research*, 21(140): 1–67.

Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A. N.; Kaiser, L.; and Polosukhin, I. 2017. Attention is all you need. In *Advances in neural information processing systems*, 5998–6008.

6 Appendix

Code Structure

The project is structured as follows:

- **attention/**: Contains implementation of Attention model
- **mamba/**: Contains implementation of Mamba model
  - **mamba_block.py**: The implementation of Mamba block
  - **mamba_model.py**: Mamba model is created using Mamba blocks
- **utils.py**: Utility methods for xLSTM blocks
- **mLSTMblock.py**: Implementation of mLSTM Block
- **sLSTMblock.py**: Implementation of sLSTM Block
- **xLSTM.py**: Implementation of xLSTM layer
- **xLSTMmodel.py**: xLSTM model is created using xLSTM layer
- **attention_test.py, mamba_test.py, xlstm_test.py**: Testing files for the models
- **train_GPT.ipynb**: Training and evaluation script
- **demo.py**: Demo code to experience the models
- **requirementes.txt**: Project implementation dependencies
- **itos_shake.pkl**: Pickle of character encoder based on Tiny Shakespeare data
- **stoi_shake.pkl**: Pickle of character decoder based on Tiny Shakespeare data
- **input.txt**: Tiny Shakespeare data

The run the code, A Cuda enabled machine is required with NVCC available in the system path. The environment can be set up using the *requirements.txt* in anaconda. After creating a virtual environment use:

```
conda install --yes --file requirements.txt
```

Or we can use the method mentioned in the *README.md*. Following environment creation, all the files can be run. To run the demo. Refer User manual.

Classes and Descriptions

1. `Head`

Description: A single head of self-attention mechanism. This class computes the attention scores and performs weighted aggregation of values.

Inputs:

- x: A tensor of shape (B, T, C), where B is the batch size, T is the time-step, and C is the number of channels.

Outputs:

- A tensor of shape $(B, T, \text{head size})$ representing the output of the self-attention head.

2. `MultiHeadAttention`

Description: Implements multiple heads of self-attention in parallel. Combines the output from multiple `Head` instances.

Inputs:

- x: A tensor of shape (B, T, C).

Outputs:

- A tensor of shape $(B, T, \text{n\_embd})$, where `n_embd` is the embedding dimension.

3. `FeedForward`

Description: A simple feedforward neural network layer with a ReLU activation followed by dropout.

Inputs:

- x: A tensor of shape $(B, T, \text{n\_embd})$.

Outputs:

- A tensor of shape $(B, T, \text{n\_embd})$ representing the output of the feedforward network.

4. `Block`

Description: A Transformer block that consists of a MultiHeadAttention layer followed by a FeedForward layer with residual connections.

Inputs:

- x: A tensor of shape $(B, T, \text{n\_embd})$.

Outputs:

- A tensor of shape $(B, T, \text{n\_embd})$.

5. `GPTLanguageModel`

Description: An implementation of a GPT (Generative Pre-trained Transformer) language model. It uses multiple `Block` instances to process the input tokens.

Inputs:

- idx: A tensor of shape (B, T) containing input token indices.
- $targets$: (Optional) A tensor of shape (B, T) containing target token indices for loss calculation.

Outputs:

- $logits$: A tensor of shape $(B*T, \text{vocab\_size})$ representing the predicted logits.
- $loss$: (Optional) The calculated loss value.

6. `ResidualBlock`

Description: A residual block that wraps a `MambaBlock` with normalization and a residual connection.

Inputs:

- x: A tensor of shape (B, L, D).

Outputs:

- A tensor of shape (B, L, D).

7. `MambaBlock`

Description: A single block from the Mamba architecture, featuring convolutional and state-space models.

Inputs:

- x: A tensor of shape (B, L, D).

Outputs:

- A tensor of shape (B, L, D).

8. `RMSNorm`

Description: Implements Root Mean Square Layer Normalization.

Inputs:

- x: A tensor of shape (B, L, D).

Outputs:

- A normalized tensor of shape (B, L, D).

9. `MambaLanguageModel`

Description: An implementation of a language model using the Mamba architecture.

Inputs:

- idx: A tensor of shape (B, T) containing input token indices.
- $targets$: (Optional) A tensor of shape (B, T) containing target token indices for loss calculation.

Outputs:

- $logits$: A tensor of shape $(B*T, \text{vocab\_size})$ representing the predicted logits.
- $loss$: (Optional) The calculated loss value.

10. `XLSTMLanguageModel`

Description: An implementation of a language model using the xLSTM architecture.

Inputs:

- idx: A tensor of shape (B, T) containing input token indices.
- $targets$: (Optional) A tensor of shape (B, T) containing target token indices for loss calculation.

Outputs:

- $logits$: A tensor of shape $(B*T, \text{vocab\_size})$ representing the predicted logits.
- $loss$: (Optional) The calculated loss value.

11. `mLSTMblock`

Description: A block in the mLSTM architecture, which includes a modified LSTM cell.

Inputs:

- x: A tensor of shape (B, T, D) representing the input features.

Outputs:

- A tensor of shape (B, T, D) representing the output features.

12. `sLSTMblock`

Description: A block in the sLSTM architecture, designed for sequential processing with LSTM cells.

Inputs:

- x: A tensor of shape (B, T, D) representing the input features.

Outputs:

- A tensor of shape (B, T, D) representing the output features.

13. `xLSTM`

Description: An LSTM architecture with extended capabilities for sequence modeling.

Inputs:

- x: A tensor of shape (B, T, D).

Outputs:

- A tensor of shape (B, T, D) representing the LSTM output.

14. `BlockDiagonal`

Description: A block diagonal matrix operation, often used for efficiently applying the same linear transformation across multiple subspaces.

Inputs:

- x: A tensor of shape (B, T, D).

Outputs:

- A tensor of shape (B, T, D) after applying the block diagonal transformation.

15. `CausalConv1D`

A 1D convolutional layer with a causal padding, ensuring that the output at time t only depends on inputs from time $\leq t$.

User Manual

To run the demo code, ensure that the environment is set up using the requirements.txt, as mentioned in the Code introduction.

```
python demo.py -m <model_name> -c <context>
```

Where the model_name can be *attention*, *mamba*, *xlstm* and the context can be any string enclosed in **double quotes** for example, ”Hard work pays off”.

The training files can be run from the Interactive python file. The results will be logged in the wandb interface.

Test Runs

- Without context

Attention:

```
Save than against to heightly.

MENENIUS:
Shy is.

BUCKINIUS:
What
He we not was abson his good dish.

CORIOLANUS:
I how!
I'll.

CORIOLANUS:
No mean, a
I', I lend be company.

SICINIUS:
You a' here do.

Third Murderer:
Thus me our some: sin
How you, by have, make an I'll bear;
dant The other speects in at Kenewly, gone.
Shall  the beauty; Dest it: to on blaste I
cament gent.

HOMNTES:
Hare is not nothn. Should At and ally had!

By A:
He hope's on the genbishoalt joys we,
IsA cavew far words: do
```

Mamba:

```
MENENIUS:
You for your will; hold with the worthiear?

CLAUDIO:
Good I'll be the young sin where I hate it not.

BRAKENBURY:
No, to make a bishow both since we need wock him he
Which I have been folly.

BUCKINGHAM:
Is my brachs must so claim he? mercy,
For setting coward, Teratiate, a moderannipals
Which wise dreams once and many marjorn!
Now I'll so, were loathy ingrate.

ROMEO:
Sir, a little express
A wilk one alread-whit he may obg, whereon
they are fair prating, boy?

BIANCA:
Take thee? h
```

xLSTM:

```
And the daughter may subWill; on hall
havey me had-high do bar impets aboating
you. know all to sery speed
He est me?

SICINIUS:
Ye my ape your honour'd here unken on
banish meanies's, Provost, at preceet be
dowth Upoppysud teath.
-yeare thousand too use youry, and in kins
Havilly of heath homoubteen, to his contrary:
Pid, I dot dost the stricabined!

GLOUCESFANEY:
I show lives me?

CA. How now, but we kere it it?

SICINIUS:
Gesee! framise? were's the Vongue; he not so,
speaks pasters' To faith
```

- context: ”Hard work is the key”

Attention:

```
Hard work is the key?

FRICHARD:
You deed, if more threflesh most be.

KING HEDRMAK:
His noble a was and yours I speak?
.

CLARESN:
Why, gown, 'twaster? Farewell.

KING RICHARD IIII:
Richmour o'wos no por undertck:
Now fix'd to Rome, this mustness resome right
Where paon niger? O the husbury! You'lt sweat,
he is Most to agre bed mocker, and pertroy ours
The chires dis of his envile this his,
A choly not right be for hell;
Unless whosYork
A madshe times of you beglow. He father Clarenon
I drink to yse me us.
I can a
```

Mamba:

```
Hard work is the key
From disposition to made of mother,
a showquer deniedts must: under
some hope, being in his foe?

AUTOLrone honour from Clowd true pity of with.

JULIET:
What, this is was usurp, of your first?

BENVOLIO:
.orlous to Rome; come-with yourself.

FRIAR YORK:
Dead strength, as now that of by: hence is bunty
And meching your reason or heed, not yourselves to the usuosing visage
The two one if conscience than forget'st more.

First MSNCIO:
By soafter oak, falsehood,
Is ashipforge the middle those eoft
```

xLSTM:

```
Hard work is the keys
Not ear his range him!

RENHY knjlingby's resple-bid fight rights
hell would; what makes oncys,
And dock.

BRCKINCENTIO:
I doublio,
Endeslent; fly, and words with the like the wors
Of Look have get primion with been our pooscare
Is butward and to the trague
A from our pitobe now's and seem tickeer,
I do blass our frallANUS:
Yet been 'friend to same matters of hope;
And it, my father shall queen: fram our tends befol!
I meant corruct a walls the royour to her,
A sime know, and most my sinnic ca
```