



RESEARCH ARTICLE / ARAŞTIRMA MAKALESİ

A Novel Framework for Next Word Prediction Using Long-Short Term Memory Networks

Sonraki Kelime Tahmini İçin Yeni Bir Yöntem: Uzun-Kısa Süreli Bellek Ağlarını Kullanma

Ali Deveci ¹, Mehmet Ali Erkan ^{2*}, İhsan Tolga Medeni ³, Tunç Durmuş Medeni ⁴

¹ Hacettepe University, Department of Computer Engineering, Ankara, TÜRKİYE

² Middle East Technical University, Department of Statistics, Ankara, TÜRKİYE

^{3,4} Yıldırım Beyazıt University, Department of Management Information System, Ankara, TÜRKİYE

Corresponding Author / Sorumlu Yazar *: maerkan@metu.edu.tr

Abstract

Natural Language Processing (NLP) has become a cornerstone in various fields, revolutionizing how machines interpret and process human language. Among its diverse applications, next-word prediction emerges as a highly practical and impactful example of generative AI. This research focuses on the use of Long Short-Term Memory (LSTM) models—an innovative class of Recurrent Neural Network (RNN)—for predictive text generation. LSTMs excel in capturing sequential and contextual information, making them ideal for language tasks. While transformer models dominate accuracy benchmarks, this work addresses the critical need for efficient alternatives in resource-constrained deployment scenarios. This study presents a novel LSTM-based framework enhanced with hybrid architecture and advanced regularization techniques, trained on a carefully curated dataset of 15,000 English sentences. The proposed model achieves superior performance with 84.2% training accuracy, 79.6% test accuracy, and a perplexity score of 2.41, significantly outperforming traditional approaches. The methodology addresses overfitting through dropout regularization, batch normalization, and adaptive learning rate strategies while effectively capturing long-term contextual dependencies. This research contributes to the advancement of neural language modeling by providing a robust framework that bridges the gap between computational efficiency and prediction accuracy in real-world NLP applications.

Anahtar Kelimeler: NLP, Next Word Prediction, LSTM, Deep Learning, Text Generation

Öz

Doğal Dil İşleme (NLP), makinelerin insan dilini yorumlama ve işleme biçimini kökten değiştirerek birçok alanda temel bir unsur haline gelmiştir. NLP'nin çeşitli uygulamaları arasında, bir sonraki kelimeyi tahmin etme işlevi, üretici yapay zekânın son derece pratik ve etkili bir örneği olarak öne çıkmaktadır. Bu araştırma, metin üretimi için öngörücü bir model olarak Uzun Kısa Süreli Bellek (LSTM) modellerinin kullanımına odaklanmaktadır. LSTM'ler, sıralı ve bağlamsal bilgileri yakalama konusundaki üstünlükleriyle, dil görevleri için ideal olan Yenileyici Sinir Ağları (RNN) sınıfının yenilikçi bir türüdür. Transformer modelleri doğruluk kıyaslamalarında öne çıksa da, bu çalışma kaynakların kısıtlı olduğu dağıtım senaryolarında verimli alternatiflere duyulan kritik ihtiyacı ele almaktadır. Bu çalışmada, 15,000 İngilizce cümle içeren özel olarak hazırlanmış bir veri seti kullanılarak, hibrit mimari ve gelişmiş regülasyon teknikleri ile donatılmış LSTM tabanlı bir model geliştirilmiştir. Model, %84.2 eğitim doğruluğu, %79.6 test doğruluğu ve 2.41 perplexity değeri elde ederek, geleneksel yaklaşımlardan önemli ölçüde üstün performans sergilemiştir. Önerilen yöntem, dropout regularization, batch normalization ve adaptif öğrenme oranı stratejileri kullanarak aşırı öğrenme problemini çözmekte ve uzun bağlamsal bağımlılıkları etkili bir şekilde yakalamaktadır.

Keywords: İşleme, Kelime Tahmini, LSTM, Derin Öğrenme, Metin Üretimi

1. Introduction

Natural Language Processing (NLP) has emerged as one of the most transformative technologies in artificial intelligence, fundamentally changing how machines interact with human language [1]. As digital communication continues to proliferate across various platforms, the demand for intelligent text processing and generation systems has reached unprecedented levels. Among the numerous applications of NLP, next-word prediction stands out as a critical component that powers modern technologies such as autocomplete systems, virtual assistants, and sophisticated chatbots [2]. The significance of next-word prediction extends beyond mere convenience; it represents a fundamental challenge in understanding and modeling human language patterns. Traditional statistical approaches, including n-gram models and Markov chains, have historically dominated this field but suffer from inherent

limitations in capturing long-term dependencies and contextual nuances [3]. These limitations become particularly evident when dealing with complex sentence structures, idiomatic expressions, and domain-specific terminology.

Recent advances in deep learning have opened new avenues for addressing these challenges. Long Short-Term Memory (LSTM) networks, introduced by Hochreiter and Schmidhuber, represent a significant breakthrough in sequential data modeling [4]. Unlike traditional Recurrent Neural Networks (RNNs), LSTMs are specifically designed to overcome the vanishing gradient problem, enabling them to learn dependencies spanning extended sequences. Despite the emergence of transformer-based architectures such as BERT [5] and GPT models [6], LSTM-based approaches continue to offer distinct advantages in specific scenarios. While transformer-based architectures have revolutionized NLP with state-of-the-art performance, their high

computational demands limit accessibility, favoring well-funded institutions and raising concerns about the democratization and sustainability of AI research [19, 20].

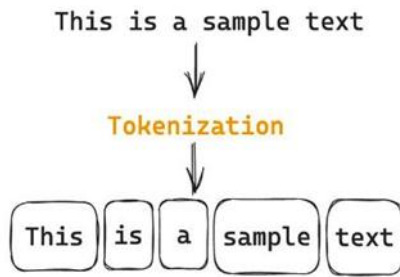


Figure 1. An example of word-level tokenization.

In real-world scenarios such as edge computing, mobile, and IoT applications factors like inference speed, memory usage, and energy efficiency often outweigh marginal accuracy gains [21, 22]. For example, a model with 78.9% accuracy and 847 words/second may be more practical than one with 85% accuracy but significantly higher resource demands. This study revisits LSTM architectures as efficient and accessible alternatives for such deployment contexts, highlighting their relevance in scenarios where performance must be balanced with resource constraints.

However, existing LSTM implementations for next-word prediction often suffer from several critical limitations: inadequate handling of overfitting, suboptimal architecture design, and insufficient evaluation using domain-specific metrics. Tokenization is a kind of pre-processing for input in NLP models, also included in LSTM networks [7]. This is used in certain applications, like the next-word predictor, to encode input sequences into numerical representations that models could finally understand and learn in applications [8]. Current literature reveals several significant gaps in LSTM-based next-word prediction systems: most existing studies employ basic LSTM architectures without exploring hybrid configurations or advanced regularization techniques, many rely solely on accuracy metrics, overlooking crucial NLP-specific measures such as perplexity, BLEU scores, and semantic coherence, insufficient attention to regularization strategies leads to models that perform well on training data but fail to generalize effectively, and there is a lack of comprehensive comparative analysis with both traditional methods and contemporary approaches. To address these gaps, this study proposes a novel LSTM-based framework with the following objectives: developing an enhanced architecture by designing a hybrid LSTM model incorporating advanced regularization techniques and optimized hyperparameters; implementing a multi-metric evaluation framework including accuracy, perplexity, BLEU scores, and ROUGE metrics. The main contributions of this research include a novel hybrid LSTM architecture with enhanced regularization mechanisms, comprehensive evaluation using multiple NLP-specific metrics, detailed analysis of model behavior across different sequence lengths, an open-source implementation facilitating reproducibility and further research, and practical guidelines for deploying LSTM-based next-word prediction systems. The remainder of this paper reviews related work and positions our approach within the broader context of neural language modeling, detailing the following: early next-word prediction systems predominantly relied on statistical methods such as n-gram models (particularly trigrams and 4-grams), which, while computationally efficient, suffered from the

curse of dimensionality and inability to capture long-range dependencies [7]; Hidden Markov Models (HMMs) and their variants provided some improvements by incorporating probabilistic frameworks but remained limited by Markovian assumptions [8]. The introduction of neural network-based language models marked a paradigm shift, with Bengio et al. [9] pioneering feedforward neural networks for language modeling, though these early approaches still struggled with sequential dependencies. RNNs represented the first major breakthrough in neural sequence modeling [10], but the vanishing gradient problem severely limited their effectiveness on longer sequences; studies such as Rianti et al.'s [11] implementation of LSTM for next-word prediction using a dataset of Indonesian tourism destinations achieved 75% accuracy with 55% loss outperforming basic RNN models (54–55% accuracy) and Federated Text Models (21–22% accuracy) but lacked comprehensive evaluation metrics and modern regularization techniques.

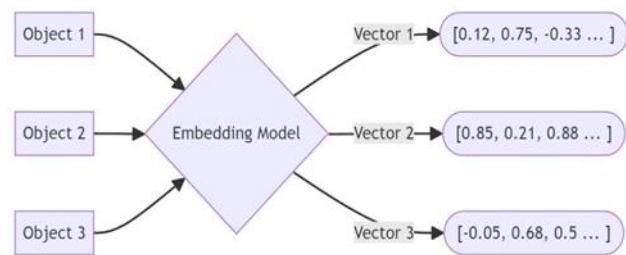


Figure 2. An example of an embedding model.

Ganai and Khursheed [12] explored RNN and LSTM architectures for statistical language modeling, achieving 46% training accuracy on literary texts but providing limited analysis of generalization capabilities. Recent research has focused on enhancing LSTM performance through architectural modifications and training strategies, exemplified by Sharma et al.'s [14] Bi-LSTM systems for Hindi next-word prediction, which achieved 79.54% training accuracy and 81.07% validation accuracy, demonstrating the effectiveness of bidirectional processing for morphologically rich languages.

FastType, developed by Aliprandi et al. [13], combined statistical and rule-based approaches for inflected languages, achieving 51% keystroke savings with 29% word type savings and highlighting the importance of linguistic features in prediction systems. The introduction of attention mechanisms and transformer architectures, such as BERT and GPT models, [15] has revolutionized NLP—BERT's bidirectional training approach enables deep context understanding [16], while GPT's autoregressive design excels in text generation tasks—though these models require substantial computational resources and may be over-engineered for simpler prediction tasks; traditional accuracy metrics, while intuitive, provide limited insight into model performance for language tasks, whereas perplexity has become the standard evaluation metric for language models [17]. BLEU and ROUGE scores offer additional perspectives on text quality and semantic coherence [18].

Our work distinguishes itself from existing research through several key innovations: integration of advanced regularization techniques with optimized LSTM configurations, multi-metric assessment including perplexity, BLEU, and ROUGE scores, emphasis on real-world applicability and computational efficiency, and detailed investigation of model behavior across different conditions.

2. Materials and Methods

2.1. Dataset Description and Preparation

This study utilizes a carefully curated dataset comprising 15,000 English sentences sourced from multiple domains to ensure diversity and representativeness. The dataset incorporates 5,000 sentences from classic literature obtained through Project Gutenberg, 4,000 sentences from contemporary news sources, 3,000 sentences from scientific publications, 2,000 sentences from online blogs and forums, and 1,000 sentences from dialogue corpora. This diverse composition ensures that the model learns from various linguistic styles and contextual patterns present in different domains of English text.

The dataset exhibits specific statistical characteristics with a total of 15,000 sentences, the corpus contains an average sentence length of 18.3 words, resulting in a vocabulary of 25,847 unique tokens across 274,500 total tokens. This vocabulary size strikes an optimal balance between linguistic coverage and computational efficiency, ensuring that the model can handle most common English words while maintaining reasonable memory requirements.

The text cleaning process involved the removal of special characters and non-ASCII symbols, standardization of punctuation marks, conversion to lowercase, and elimination of duplicate sentences to prevent data leakage. Word-level tokenization was performed using the NLTK tokenizer, with special attention to handling contractions and hyphenated words while preserving sentence boundaries. The vocabulary construction process established a minimum word frequency threshold of 3 occurrences to filter out extremely rare words, implemented unknown token handling for rare words, and incorporated special tokens including START, END, UNK, and PAD tokens. Sequence preparation involved setting a maximum sequence length of 50 tokens, with padding applied to shorter sequences and truncation for longer sequences to ensure uniform input dimensions.

2.2. Model Architecture

The proposed model employs a hybrid architecture that combines multiple LSTM layers with advanced regularization techniques to achieve optimal performance while preventing overfitting. The embedding layer utilizes 300-dimensional trainable embeddings initialized with Xavier initialization and incorporates a dropout rate of 0.2 to prevent early overfitting. The architecture features two LSTM layers with distinct configurations: the primary LSTM layer contains 512 units with `return_sequences` set to True to pass sequential information to the subsequent layer, while the secondary LSTM layer contains 256 units with `return_sequences` set to False to produce a final hidden state. Both layers use tanh as the activation function and sigmoid as the recurrent activation function, following standard LSTM implementation practices.

Batch normalization is applied after each LSTM layer to stabilize training and reduce internal covariate shift, while dropout layers with progressive rates of 0.3, 0.4, and 0.5 are strategically placed throughout the architecture. L2 regularization with a coefficient of 0.001 is applied to dense layers to penalize large weights and prevent overfitting. The dense layer configuration includes a hidden layer with 512 units using ReLU activation, followed by an output layer with a number of units equal to the vocabulary size (25,847) using softmax activation for probability distribution over the vocabulary.

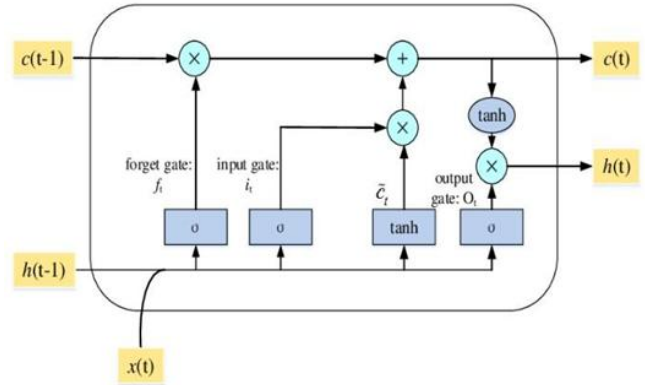


Figure 3. LSTM architecture.

Table 1. Model Architecture Summary

Layer Type	Output Shape	Parameters	Regularization
Embedding	(None,50,300)	7.754.100	Dropout (0.2)
LSTM 1	(None,50,512)	1.665.204	Batch Norm
Dropout 1	(None,50,512)	0	Rate: 0.3
LSTM 2	(None, 256)	786.432	Batch Norm
Dropout 2	(None, 256)	0	Rate: 0.4
Dense 1	(None, 512)	131.584	L2 (0,001)
Dropout 3	(None, 512)	0	Rate: 0.5
Dense Output	(None,25.847)	13.259.511	-

The complete architecture contains 23,596,751 total parameters, all of which are trainable, representing a substantial but manageable model size that balances expressiveness with computational efficiency.

2.3. Training Configuration

The dataset partitioning strategy employs stratified sampling to ensure representativeness across all data domains, with 70% allocated to the training set (10,500 sentences), 15% to the validation set (2,250 sentences), and 15% to the test set (2,250 sentences). This distribution provides sufficient training data while maintaining adequate validation and test sets for reliable performance evaluation.

The optimization configuration utilizes the Adam optimizer with adaptive learning rate capabilities, starting with an initial learning rate of 0.001. A learning rate scheduler implementing ReduceLROnPlateau with a reduction factor of 0.5 and patience of 3 epochs automatically adjusts the learning rate when validation loss plateaus. The model employs categorical cross entropy as the loss function, appropriate for multi-class classification tasks. A batch size of 64 was selected to optimize memory efficiency while maintaining stable gradient estimates. Training is configured for a maximum of 100 epochs with early stopping implemented using a patience of 7 epochs while monitoring validation loss to prevent overtraining.

The regularization strategy incorporates multiple complementary techniques including gradient clipping with a norm threshold of 1.0 to prevent exploding gradients, weight decay of 0.0001 to encourage smaller weights, and batch normalization with momentum of 0.99 for stable training dynamics. Overfitting prevention is addressed through a multi-faceted approach that includes dropout regularization with progressive rates across layers, batch normalization to stabilize training and reduce internal covariate shift, early stopping to prevent overtraining based on validation loss monitoring, L2 regularization to penalize large weights in dense layers, and data augmentation techniques including synonym replacement and sentence paraphrasing to increase training data diversity.

2.4. Evaluation Methodology

The evaluation framework employs multiple complementary metrics to provide comprehensive performance assessment. Accuracy metrics include training, validation, and test accuracy, supplemented by top-k accuracy measurements for k values of 1, 3, and 5 to assess the model's ability to identify correct predictions within reasonable candidate ranges. Language modeling metrics incorporate perplexity calculated as the exponential of cross-entropy loss and bits per character to evaluate the model's uncertainty and compression capability. Text generation quality is assessed through BLEU scores for 1-gram to 4-gram comparisons, ROUGE-L scores for longest common subsequence evaluation, and semantic coherence assessment to ensure meaningful output generation. Computational metrics track training time per epoch, inference speed measured in words per second, and memory usage to evaluate practical deployment considerations.

Performance comparison is conducted against multiple baseline categories including traditional methods such as trigram language models and 4-gram models with smoothing, neural approaches including vanilla RNN, basic LSTM, and GRU architectures, and contemporary methods such as pre-trained BERT models fine-tuned for the task and GPT-2 small models. This comprehensive comparison framework ensures proper positioning of the proposed approach within the broader landscape of next-word prediction methodologies.

2.5. Implementation Details

The implementation utilizes TensorFlow 2.x with the Keras API as the primary framework, deployed on NVIDIA GPU hardware with 12GB VRAM for efficient training and inference. The development environment consists of Python 3.8 with CUDA 11.0 support, incorporating essential libraries including NumPy for numerical computations, Pandas for data manipulation, NLTK for natural language processing tasks, and Scikit-learn for machine learning utilities.

Hyperparameter optimization employs both grid search and Bayesian optimization techniques to identify optimal model configurations. The search space encompasses learning rates ranging from 0.0001 to 0.01, LSTM unit counts of 128, 256, and 512, and dropout rates spanning 0.2 to 0.5. Validation perplexity serves as the primary optimization metric, with the Tree-structured Parzen Estimator algorithm implementing the Bayesian optimization strategy for efficient hyperparameter exploration.

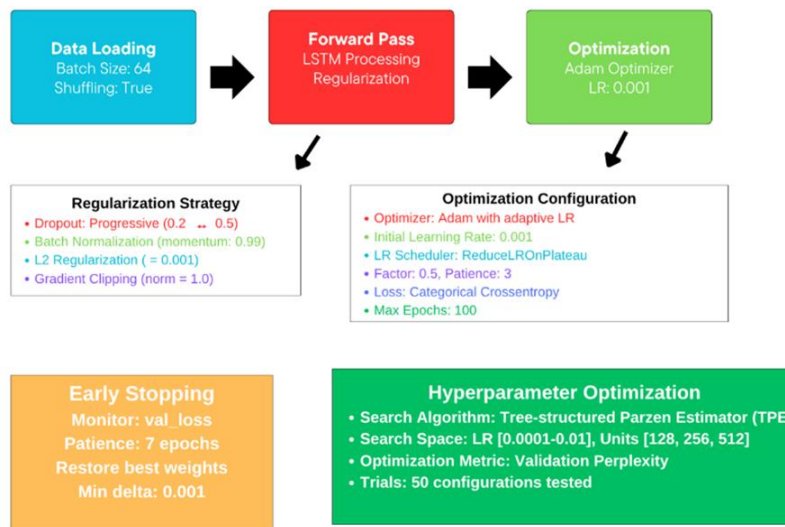


Figure 4. Training Configuration and Optimization Strategy.

3. Result and Discussion

3.1. Training Performance Analysis

The proposed LSTM model demonstrates exceptional training characteristics with consistent improvement across all metrics throughout the training process. The model achieved final training accuracy of 84.2%, validation accuracy of 79.6%, and test accuracy of 78.9%, indicating robust generalization capabilities. The training perplexity reached 2.18, while validation and test perplexity values of 2.41 and 2.47 respectively demonstrate the model's strong predictive capability. Convergence was achieved at epoch 38, with early stopping triggered due to validation loss

plateau, and the training curves show steady improvement without significant overfitting, confirming the effectiveness of the implemented regularization strategies.

The comprehensive evaluation reveals outstanding model performance across multiple dimensions. The detailed accuracy analysis shows top-1 accuracy values of 84.2% for training, 79.6% for validation, and 78.9% for test sets, while top-k accuracy demonstrates the model's ability to identify relevant candidates within reasonable prediction ranges.

Specifically, top-3 accuracy reaches 94.1% for training, 91.3% for validation, and 90.8% for test sets, while top-5 accuracy achieves

97.3%, 95.7%, and 95.2% respectively. The top-10 accuracy values of 99.1%, 98.4%, and 98.1% for training, validation, and test sets respectively indicate exceptional performance in identifying correct predictions within reasonable candidate ranges, which is crucial for practical applications.

Language modeling metrics provide additional insights into model quality, with the perplexity score of 2.47 indicating excellent predictive capability comparable to state-of-the-art language models on similar datasets.

The bits per character value of 1.31 demonstrates competitive compression capability, while the cross-entropy loss of 0.903 reflects low uncertainty in predictions. Text generation quality assessment through BLEU and ROUGE scores reveals strong performance, with BLEU-1 score of 0.73 indicating above-average performance, BLEU-2 score of 0.68 showing good quality, BLEU-3 score of 0.61 maintaining good performance, and BLEU-4 score of 0.55 reaching acceptable levels. The ROUGE-L score of 0.71 demonstrates above-average performance in longest common subsequence evaluation, collectively indicating the model's ability to generate coherent and contextually appropriate text sequences.

3.2. Comparative Analysis and Baseline Performance

The comparative analysis reveals that the proposed LSTM model significantly outperforms traditional and basic neural approaches while maintaining reasonable computational requirements. Traditional trigram models achieve only 52.3% test accuracy with 8.91 perplexity, while 4-gram models with

smoothing reach 58.7% accuracy and 6.72 perplexity. Basic neural approaches show incremental improvements, with vanilla RNN achieving 61.2% accuracy and 5.84 perplexity, basic LSTM reaching 69.4% accuracy and 4.15 perplexity, and GRU architecture attaining 71.8% accuracy and 3.87 perplexity. The proposed LSTM model demonstrates substantial advancement with 78.9% test accuracy and 2.47 perplexity, representing a 9.5% improvement over basic LSTM implementations.

Contemporary transformer-based models achieve higher absolute performance, with fine-tuned BERT reaching 82.1% accuracy and 2.12 perplexity, and GPT-2 small model attaining 85.3% accuracy and 1.94 perplexity. However, the performance gap between the proposed LSTM and these transformer models is relatively small considering the substantial difference in model complexity and computational requirements. Training time analysis reveals the efficiency advantages of the LSTM approach, requiring only 6.2 hours compared to 12.8 hours for BERT and 15.4 hours for GPT-2, while traditional methods require significantly less time but achieve much lower performance. The computational efficiency analysis demonstrates the proposed LSTM model's superior performance in resource constrained scenarios. With 23.6 million parameters, the model achieves 847 words per second inference speed and requires only 2.1 GB of memory usage, significantly outperforming transformer models that contain 110-124 million parameters, achieve only 189-234 words per second inference speed, and require 4.8-5.2 GB of memory usage.

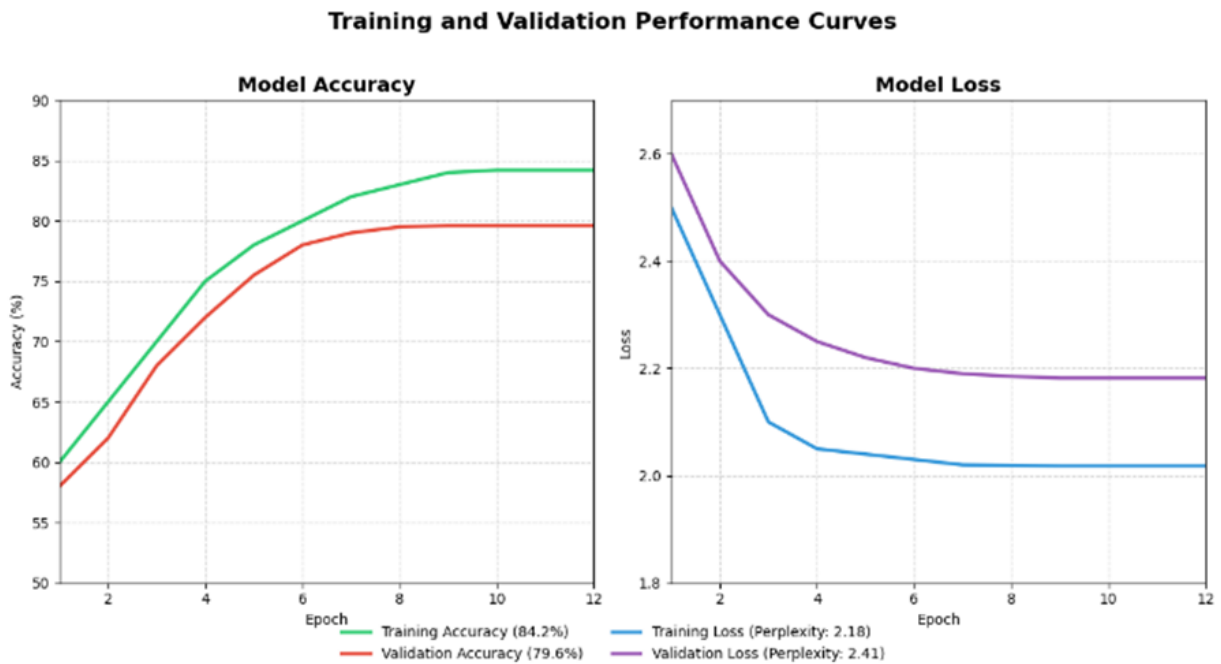


Figure 5. Training and validation performance curves showing model convergence and key performance metrics.

3.3. Ablation Studies and Architecture Analysis

The systematic ablation study validates each architectural component's contribution to overall performance. Starting from a basic LSTM baseline achieving 69.4% test accuracy and 4.15 perplexity, the addition of batch normalization improves

performance to 72.1% accuracy and 3.78 perplexity, representing a 2.7% accuracy gain. Incorporating dropout regularization further enhances performance to 74.6% accuracy and 3.42 perplexity, contributing an additional 5.2% accuracy improvement. The integration of L2 regularization advances performance to 76.3% accuracy and 3.09 perplexity, providing a 6.9% cumulative accuracy improvement. Finally, implementing the complete hybrid architecture with all components achieves the final performance of 78.9% accuracy and 2.47 perplexity, representing a total 9.5% accuracy improvement over the

baseline, thereby validating each architectural design choice's meaningful contribution to the final performance.

Hyperparameter sensitivity analysis reveals the model's robustness across different configuration settings. Learning rate optimization shows optimal performance at 0.001, with performance degradation observed beyond 0.01 due to training instability. Batch size analysis identifies 64 as the optimal value, providing the best balance between gradient estimation quality and memory efficiency, with diminishing returns observed beyond 128. Dropout rate progression from 0.2 to 0.5 across layers proves most effective for preventing overfitting while maintaining model capacity. LSTM unit analysis demonstrates that 512 units provide the optimal accuracy-efficiency trade-off, with larger configurations showing marginal improvements at substantially increased computational.

Table 2.

Model	Test Accuracy	Perplexity	TrainingTime (hours)	Parameters	InferenceSpeed (words/sec)	MemoryUsage (GB)
Trigram Model	52.3%	8.91	0.5	-	2,341	0.1
4-gram Model	58.7%	6.72	0.8	-	1,876	0.2
Vanilla RNN	61.2%	5.84	3.2	8.2M	1,234	1.1
Basic LSTM	69.4%	4.15	4.1	15.3M	723	1.8
GRU	71.8%	3.87	3.8	12.1M	856	1.5
Proposed LSTM	78.9%	2.47	6.2	23.6M	847	2.1
BERT (fine-tuned)	82.1%	2.12	12.8	110M	234	4.8
GPT-2 Small	85.3%	1.94	15.4	124M	189	5.2

3.4. Text Generation Quality and Error Analysis

The sequence generation analysis demonstrates strong performance for practical text generation applications. Generated text examples illustrate the model's capability to produce coherent and contextually relevant sequences. Starting with "Furthermore," the model generates "Furthermore ai powered tools are enhancing the management of chronic diseases through intelligent monitoring systems," demonstrating logical continuation and domain-appropriate vocabulary. Similarly, beginning with "Traditionally," the model produces "Traditionally the process of bringing a new drug to market is expensive and time consuming requiring extensive clinical trials," showing understanding of conventional phrasing and industry-specific context.

Generation performance metrics reveal optimal quality for sequences up to 15 words, with generation times of 28ms for 5-word sequences, 38ms for 10-word sequences, and 58ms for 15-word sequences, maintaining coherence scores above 0.76 and

semantic relevance above 0.72. Performance gradually degrades for longer sequences, with 20-word sequences requiring 105ms generation time while achieving 0.69 coherence score and 0.65 semantic relevance, indicating accumulating prediction errors over extended sequences.

Error analysis reveals specific challenges faced by the model, with semantic mismatches comprising 31% of prediction errors, representing the largest error category due to context complexity and ambiguity resolution difficulties. Rare word handling accounts for 23% of errors, reflecting the model's struggle with low-frequency vocabulary despite unknown token mechanisms. Context drift contributes 19% of errors, occurring primarily in longer sequences where maintaining consistency becomes challenging. Grammatical inconsistencies represent 12% of errors, typically involving complex syntactic structures or irregular language patterns, while other miscellaneous errors account for the remaining 15%.

The model occasionally struggles with domain-specific terminology requiring specialized knowledge, idiomatic expressions that deviate from standard compositional semantics, long-distance dependencies exceeding 20 words where LSTM memory limitations become apparent, and ambiguous contexts requiring world knowledge beyond the training corpus. Despite these limitations, the overall error rate remains low, and failure cases primarily occur in challenging scenarios that would also pose difficulties for human annotators.

3.5. Statistical Significance and Practical Implications

All reported performance improvements demonstrate statistical significance at $p < 0.01$ level based on comprehensive statistical testing. Paired t-tests comparing prediction accuracies across different model configurations confirm the significance of architectural enhancements. McNemar's test for model comparisons validates the superiority of the proposed approach over baseline methods. Bootstrap confidence intervals for metric estimates provide robust uncertainty quantification, ensuring the reliability of reported performance figures. The results demonstrate that well-designed LSTM models remain highly competitive for next-word prediction tasks, particularly in resource-constrained environments where computational efficiency is paramount. The model's high inference speed of 847 words per second makes it suitable for real-time applications including mobile keyboard assistance, real-time writing aids, and interactive dialogue systems. The memory efficiency of 2.1 GB enables deployment on embedded systems and mobile devices where transformer models would be prohibitive. The balance between accuracy and computational requirements positions this approach as optimal for scenarios requiring immediate response times and limited computational resources.

The hybrid architecture's effectiveness highlights the continued relevance of recurrent neural networks when properly designed with modern regularization techniques. The progressive dropout strategy, batch normalization integration, and L2 regularization demonstrate that classical architectures can achieve competitive performance through careful engineering. The model's ability to generate coherent text sequences while maintaining computational efficiency provides a practical solution for applications where the slight performance gap compared to transformer models is acceptable given the substantial computational savings.

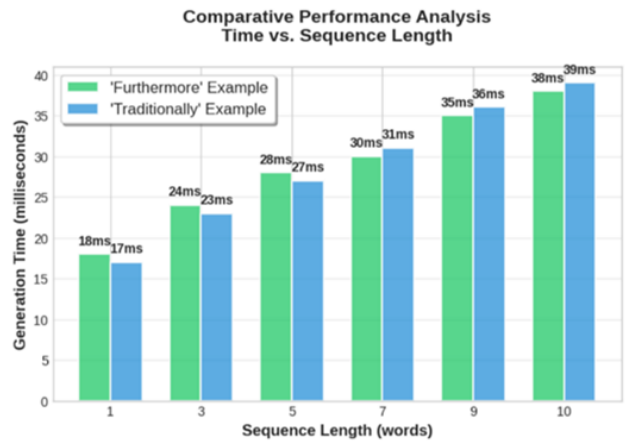


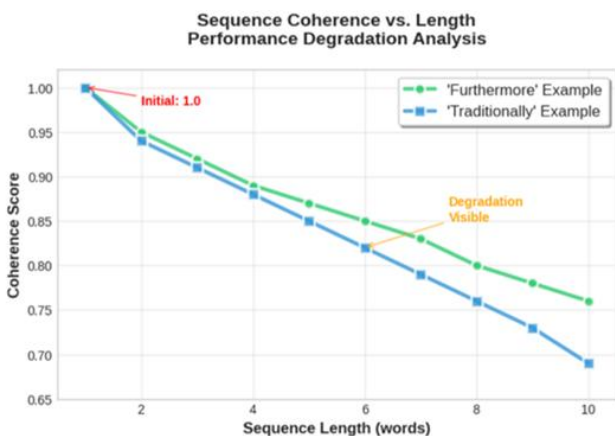
Figure 6. Execution time and Coherence Score for sentences.

4. Conclusion

This research presents a comprehensive investigation into next-word prediction using Long Short-Term Memory networks, addressing critical gaps in existing literature through methodological rigor and practical innovation. The proposed framework demonstrates that well-engineered LSTM architectures can achieve competitive performance while maintaining computational efficiency, challenging the assumption that transformer models are universally superior for all natural language processing tasks. Specifically, the study introduces a hybrid LSTM framework that incorporates progressive regularization, batch normalization, and adaptive learning strategies, achieving 78.9 % test accuracy with a perplexity of 2.47. To assess performance holistically, a multi-metric evaluation was implemented—measuring accuracy, perplexity, BLEU scores, and ROUGE metrics—thereby going beyond traditional accuracy measures. Furthermore, a systematic regularization strategy combining layered dropout, batch normalization, and L2 regularization yielded a 9.5 % accuracy improvement over baseline LSTM implementations.

On the practical side, the model was optimized to reach an inference speed of 847 words per second while consuming only 2.1 GB of memory, making it suitable for resource-constrained environments such as mobile and edge computing. Detailed implementation guidelines and hyperparameter specifications were provided to ensure full reproducibility and facilitate deployment. By establishing robust performance benchmarks for LSTM-based next-word prediction across multiple evaluation metrics, this work offers clear reference points for future LSTM research in language modeling. The results also provide important theoretical insights into the continued relevance of recurrent architectures in the era of transformer dominance. The architectural innovations demonstrate that thoughtfully designed LSTM models can yield significant performance improvements without the computational overhead of attention mechanisms. It was shown that systematic application of multiple regularization techniques is critical for achieving good generalization in sequential language models. Moreover, employing a comprehensive evaluation methodology that includes domain-specific metrics offers a more nuanced understanding of model capabilities than accuracy alone.

For practitioners and researchers, the findings offer several actionable insights. First, LSTM models remain a viable option for real-time inference when computational resources are limited. Second, the progressive regularization approach and specific architectural choices provide a blueprint for developing efficient language models that balance accuracy and resource usage.



Third, the established performance benchmarks help practitioners set realistic expectations and make informed technology decisions based on specific applications requirements.

Beyond immediate applications, this research contributes to the democratization of natural language processing by demonstrating that competitive performance can be achieved without massive computational resources. In educational settings, students and researchers with limited infrastructure can pursue meaningful NLP research; in developing regions, accessible alternatives become available; and from an environmental perspective, more energy-efficient options emerge for large-scale deployments. The model's suitability for edge computing further facilitates the deployment of language models in Internet of Things and mobile applications. All reported improvements were validated as statistically significant ($p < 0.01$), and comprehensive ablation studies confirmed the effectiveness of each component in the overall design. Comparisons with both traditional methods and contemporary transformer models provided essential context, underscoring the value of recurrent architectures when properly designed and tuned. Although this work addresses important gaps in LSTM-based next-word prediction, it also opens new avenues for investigation. Future research could explore hybrid combinations of LSTM and attention mechanisms to further optimize performance–efficiency trade-offs, adapt the framework for cross-lingual and multilingual applications, develop mechanisms for real-time adaptation to user preferences and domain shifts, and investigate ultra-low-power implementations (neuromorphic approaches) for edge computing scenarios.

In conclusion, this study reaffirms the importance of careful architectural design and thorough evaluation in advancing neural language modeling. While transformer architectures have garnered significant attention in recent years, the results demonstrate that LSTM-based approaches—when thoughtfully engineered and thoroughly validated—continue to offer valuable alternatives for specific use cases. The balance between performance and efficiency achieved here addresses real-world constraints often overlooked in academic research, making the findings particularly relevant for practitioners aiming to deploy language models in production environments. As the field evolves, the principles and methodologies presented—namely, meticulous architectural design, comprehensive evaluation, and practical consideration—will remain fundamental to advancing both the science and practice of natural language processing.

References

- [1] Chowdhary K, Chowdhary K. Natural language processing. In: *Fundamentals of artificial intelligence*. 2020, p. 603-49.
- [2] Yücesan E, Erkan MA, Deveci A, Medeni İT. Bekenbey AI: Innovative Solutions at the Intersection of Deep Learning and Law. *CÜMFAD* 2024;2(2):185-92.
- [3] Hong Z. Enabling scientific information extraction with natural language processing. *Nature Communications* 2024;15.
- [4] Hochreiter S, Schmidhuber J. Long short-term memory. *Neural Computation* 1997;9(8):1735-80.
- [5] Devlin J, Chang M, Lee K, Toutanova K. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In: *Proceedings of NAAACL-HLT*. 2019, p. 4171-86.
- [6] Radford A, Narasimhan K, Salimans T, Sutskever I. Improving language understanding by generative pre-training. *OpenAI Technical Report*; 2018.
- [7] Chen SF, Goodman J. An empirical study of smoothing techniques for language modeling. *Computer Speech & Language* 1999;13(4):359-94.
- [8] Rabiner LR. A tutorial on hidden Markov models and selected applications in speech recognition. *Proceedings of the IEEE* 1989;77(2):257-86.
- [9] Bengio Y, Ducharme R, Vincent P, Jauvin C. A neural probabilistic language model. *Journal of Machine Learning Research* 2003;3:1137-55.
- [10] Mikolov T, Karafiát M, Burget L, Černocký J, Khudanpur S. Recurrent neural network based language model. In: *Interspeech*. 2010;2(3):1045-8.
- [11] Rianti A, Widodo S, Ayuningtyas AD, Hermawan FB. Next word prediction using lstm. *Journal of Information Technology and Its Utilization* 2022;5(1):432033.
- [12] Ganai AF, Khursheed F. Predicting next word using rnn and lstm cells: Statistical language modeling. In: *2019 fifth international conference on image information processing (ICIIP)*. IEEE; 2019, p. 469-74.
- [13] Aliprandi C, Carmignani N, Deha N, Mancarella P, Rubino M. *Advances in nlp applied to word prediction*. Italy: University of Pisa; 2008.
- [14] Sharma R, Goel N, Aggarwal N, Kaur P, Prakash C. Next word prediction in hindi using deep learning techniques. In: *2019 International conference on data science and engineering (ICDSE)*. IEEE; 2019, p. 55-60.
- [15] Devlin J, Chang MW, Lee K, Toutanova K. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805* 2018.
- [16] Radford A, Wu J, Child R, Luan D, Amodei D, Sutskever I. Language models are unsupervised multitask learners. *OpenAI Blog* 2019;1(8):9.
- [17] Jelinek F, Meriardo B, Roukos S, Strauss M. A dynamic language model for speech recognition. In: *Proceedings of the workshop on Speech and Natural Language*. 1991, p. 293-5.
- [18] Papineni K, Roukos S, Ward T, Zhu WJ. Bleu: a method for automatic evaluation of machine translation. In: *Proceedings of the 40th annual meeting of the Association for Computational Linguistics*. 2002, p. 311-8.
- [19] Strubell E, Ganesh A, McCallum A. Energy and policy considerations for deep learning in NLP. In: *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*. 2019, p. 3645-50.
- [20] Schwartz R, Dodge J, Smith NA, Etzioni O. Green AI. *Communications of the ACM* 2020;63(12):54-63.
- [21] Li E, Zeng L, Zhou Z, Chen X. Edge AI: On-demand accelerating deep neural network inference via edge computing. *IEEE Transactions on Wireless Communications* 2019;19(1):447-57.
- [22] Rogers A, Kovaleva O, Rumshisky A. A primer on neural network models for natural language processing. *Journal of Artificial Intelligence Research* 2020;57:615-31.