ISSN No:-2456-2165 https://doi.org/10.38124/ijisrt/25nov578

Architectural Evaluation of Subword Tokenization and Compact Language Models (CLMs) for Resource-Constrained NLP Deployment

Arnab Sen[1]

[1](Finance, Birla Institute of Technology and Science, Pilani), India)

Publication Date: 2025/11/17

Abstract:

- ***Background***

The advancement of Natural Language Processing (NLP) is constrained by a fundamental dilemma: the immense resource requirements of Large Language Models (LLMs) versus the demand for efficient, high-performance deployment in resource-limited settings, such as edge computing.[1] This work establishes a necessary comparison between efficient deep learning alternatives and classical statistical methods.[1]

- ***Materials and Methods***

A structural and performance analysis is conducted, comparing two distinct model classes: traditional statistical N-gram models and modern Transformer-based Compact Language Models (CLMs).[1] The methodology critically evaluates core architectural differences, efficiency metrics, and the transformative impact of tokenization strategies. Key quantitative metrics, including Perplexity (PPL), and qualitative measures, such as semantic coherence and visual embedding consistency (via t-SNE), are employed.[1]

- ***Results***

CLMs, achieved through rigorous optimization techniques like pruning and quantization, exhibit superior representational capacity and drastically faster development cycles compared to resource-intensive LLMs.[1] N-gram models are fundamentally hindered by the exponential challenge of data sparsity and the inability to capture context beyond a fixed, narrow window.[1] Crucially, the CLM's implementation of subword tokenization (specifically Byte Pair Encoding, BPE) structurally solves the Out-of-Vocabulary (OOV) problem, preserving semantic information that N-gram models invariably destroy by collapsing unseen words into a generic $\langle \text{unk} \rangle$ token.[1]

- ***Conclusion***

The architectural stability, efficiency, and deep contextual fidelity afforded by optimized Compact Language Models position them as the definitive, operationally feasible choice for high-accuracy, specialized NLP tasks at the network edge.[1] While N-gram models may serve as simple baselines for modeling localized statistical distributions, their severe architectural limitations make them unsuitable for modern applications requiring complex semantic understanding.[1]

Keywords: *Compact Language Models (CLMs); Subword Encoding; Byte Pair Encoding (BPE); Edge Computing; Perplexity; Transformer.*

How to Cite: Arnab Sen (2025) Architectural Evaluation of Subword Tokenization and Compact Language Models (CLMs) for Resource-Constrained NLP Deployment. *International Journal of Innovative Science and Research Technology*, 10(11), 602-609. https://doi.org/10.38124/ijisrt/25nov578

ISSN No:-2456-2165 https://doi.org/10.38124/ijisrt/25nov578

I. INTRODUCTION: THE EFFICIENCY IMPERATIVE IN NLP

➢ *The Evolution of Language Modeling and Resource Demands*

Language Models (LMs) serve as foundational systems within Natural Language Processing (NLP), designed explicitly to calculate the probability of word sequences.[1] The field has undergone a dramatic transformation, shifting from early statistical approaches relying on counting techniques to complex, attention-based deep learning frameworks.[1] The latest generation of Large Language Models (LLMs), often featuring hundreds of billions of parameters, has successfully redefined state-of-the-art performance across a wide array of cognitive tasks.[1]

However, this unprecedented capability introduces a restrictive logistical and architectural challenge: the massive requirements for computational resources, training duration, and energy consumption.[1] These factors severely restrict their deployment and development primarily to large organizations possessing proprietary, specialized infrastructure.[1] This infrastructural centralization necessitates a strategic shift toward highly efficient, specialized alternatives. These alternatives, referred to as Compact Language Models (CLMs), are critical for operational feasibility in resource-constrained environments, including mobile devices, embedded systems, and edge computing, where parameters such as operational speed, low energy use, and data privacy take precedence.[1] The capacity to train and fine-tune these specialized models in a timeframe measured in days or weeks on smaller GPU clusters offers a significantly more cost-effective and accessible development path compared to the commitment of months required for LLMs.[1]

➢ *Literature Context: The Transition to Representation-Based Architectures*

For several decades, the dominant NLP methodology relied heavily upon statistical N-gram models.[1] These models calculate measurable sequence likelihoods using Maximum Likelihood Estimation (MLE) derived from word count frequencies within a training corpus.[1] Their core limitation, which eventually forced architectural evolution, is the finite context window, which fundamentally prevents the capture of linguistic dependencies extending beyond the preceding $N-1$ words.[1] The resultant exponential increase in computational cost as the context window (N) expands forced the field to explore more advanced, representation-based models.[1]

The definitive architectural breakthrough that resolved these context limitations was the introduction of the Transformer model.[1] By relying exclusively on a multi-head self-attention mechanism, the Transformer eliminated the need for recurrent units, thereby enabling superior parallel computation and dramatically enhanced performance in capturing global contextual relationships across long sequences.[1] Compact Language Models (CLMs) specifically leverage this powerful architectural foundation. They achieve high performance while being stringently optimized for efficiency, effectively bridging the high-performance expectations set by LLMs with the logistical realities of practical, real-world deployment scenarios.[1]

➢ *Research Focus and Contribution*

This study provides a detailed, comparative evaluation of statistical N-gram models and modern Transformer-based Compact Language Models (CLMs).[1] The primary contribution of this work is a rigorous assessment of their core mechanisms, specific resource efficiency metrics, and resultant performance when subjected to the stringent deployment conditions typical of edge computing environments.[1]

The paper asserts that the architectural robustness, inherent efficiency, and superior representational capacity offered by resource-optimized CLMs make them the indisputable choice over statistical N-gram models for contemporary resource-constrained NLP systems.[1] The analysis demonstrates how fundamental design differences—ranging from the protocols for data handling and tokenization to the internal model optimization strategies—are the key determinants for the practical applicability and subsequent success of each model type.[1]

II. DEEP LEARNING ARCHITECTURES AND COMPRESSION

➢ *N-Gram Model Vulnerabilities and Smoothing*

N-gram models operate by estimating the probability of the next word (w_i) based solely on its immediate $N-1$ preceding words, utilizing counts accumulated from the training corpus.[1] While a theoretically larger context (N) provides better theoretical relevance, practical implementation often balances stability with domain relevance, leading practitioners to typically favor trigrams ($N=3$) for large corpora or bigrams ($N=2$) for smaller datasets.[1]

The principal architectural flaw of this approach is data sparsity, where the total number of possible unique N-grams grows exponentially with both N and the size of the vocabulary.[1] This exponential growth rapidly leads to the "zero probability problem"—sequences that logically occur in the language, yet were unseen in the finite training data, are assigned a probability of zero.[1] A zero probability assignment makes key evaluations, such as Perplexity (PPL), computationally impossible without mitigation.[1] To correct this critical flaw, smoothing or discounting algorithms are mandatory.[1] These complex methods redistribute a small measure of probability mass from frequent, observed sequences to unseen ones, thereby ensuring all plausible sequences maintain a non-zero chance of occurrence. Basic techniques like Laplace smoothing are generally insufficient due to

ISSN No:-2456-2165 https://doi.org/10.38124/ijisrt/25nov578

excessive over-smoothing, necessitating the use of advanced algorithms such as Good–Turing or Kneser-Ney smoothing for any production-grade N-gram models.[1]

A critical misunderstanding often surrounds the perceived "lightweight" nature of N-gram models. Despite their rapid computational speed during statistical lookups in the inference phase, N-gram models impose immense infrastructure challenges specifically related to data storage.[1] Large-scale models can accumulate hundreds of billions of N-grams and their corresponding counts, creating a significant storage bottleneck. Mitigating this issue demands specialized engineering efforts utilizing lossless compression techniques, such as tabular trie encodings, which aim to achieve highly compact representations, sometimes reducing the storage requirement to approximately 23 bits per N-gram.[1] This required complexity for maintenance, specialized deployment optimization, and the high resultant Total Cost of Ownership (TCO) severely offset the superficial appearance of being inherently lightweight, highlighting that N-gram optimization is primarily data-centric, focused on managing the sparsity of the distribution itself.[1]

➢ *Compact Language Model (CLM) Architectural Optimization*

Compact Language Models (CLMs) are generally defined as Transformer-based networks with parameter counts ranging from 1 million to 10 billion.[1] They are meticulously engineered to deliver maximum operational effectiveness within severely limited computational budgets. Illustrative examples include specialized, domain-specific models like AfriBERTa, which operates efficiently with approximately 10 million parameters.[1]

The CLM's architectural core is the Transformer, which leverages its multi-head self-attention mechanism to process the entire input sequence in parallel.[1] This inherent parallelism allows the model to efficiently capture global, long-range dependencies, fundamentally eliminating the narrow contextual limitations that restrict N-gram systems.[1] The input text is first tokenized, converted to vectors via an Embedding layer, and then sequentially processed through alternating attention and feedforward layers (known as Transformer blocks).[1] An essential architectural detail involves the feedforward network present within each layer, which typically uses a hidden layer dimension four times wider than the input or output size. This configuration is necessary to introduce critical non-linearity and refine the learned representations throughout the depth of the network.[1]

The strategic alignment of model density with hardware capability is achieved through a suite of sophisticated optimization techniques that are crucial for enabling CLM deployment at the edge [1]:

- Knowledge Distillation: This technique trains a smaller "student" model to precisely replicate both the outputs and the internal state representations of a much larger "teacher" model.[1] This method efficiently transfers complex knowledge while achieving a dramatic reduction in the final parameter count.[1]
- Pruning: Pruning involves the systematic removal of redundant weights and connections within the neural network.[1] This targeted removal results in a significantly thinned, smaller, and subsequently faster operational model.[1]
- Quantization: Quantization lowers the numerical precision of the model's weights, such as changing from 32-bit floating-point precision to 8-bit integers.[1] This process drastically decreases the model's memory footprint and significantly accelerates inference, particularly when deployed on specialized edge hardware optimized for low-precision arithmetic.[1]

These architectural optimizations strategically prioritize low latency and energy efficiency. The focus on managing the density and precision of the model weights, rather than merely managing data sparsity, aligns the CLM architecture directly with the capabilities of modern specialized hardware. Consequently, the development cycle for CLMs is short—spanning days to weeks—making them substantially more accessible and cost-effective than LLMs, and perfectly suited for embedded systems deployment.[1]

III. SUBWORD TOKENIZATION AND REPRESENTATIONAL STABILITY

The method used to decompose raw text into tokens represents a primary distinguishing factor between N-gram and CLM architectures, critically determining how effectively each model can handle the complexities and inherent novelty of natural language.[1]

➢ *The Failure of Word-Level Tokenization in N-Grams*

Traditional N-gram models rely on simplistic word-level tokenization, typically splitting text based on spaces or basic punctuation.[1] This simplistic approach immediately succumbs to the severe Out-of-Vocabulary (OOV) problem: the certainty that words present during testing or deployment will be absent from the fixed training vocabulary.[1]

To address the resultant OOV zero-probability scenario, N-gram protocol mandates the replacement of all unseen words with a generic $\langle \text{unk} \rangle$ (unknown) token.[1] While necessary for computation, this step fundamentally destroys local contextual information. It operates by collapsing diverse, semantically rich terminology (e.g., specific medical jargon, new company names, or neologisms) into a single, meaningless representation.[1] This mandatory process of *approximation and collapse* results in an inherent lack of informational stability, which severely compromises the N-gram model's output quality when processing novel or specialized linguistic inputs.[1] This choice of tokenization defines a vulnerability to linguistic entropy.

ISSN No:-2456-2165 https://doi.org/10.38124/ijisrt/25nov578

➢ *Byte Pair Encoding (BPE) for CLMs*

Transformer-based CLMs overcome the fundamental OOV crisis by adopting advanced subword tokenization methods, which are designed to achieve an optimal balance between vocabulary efficiency and linguistic expressiveness.[1] This mechanism represents a decisive architectural advantage, allowing CLMs to handle diverse linguistic inputs without the contextual compromises faced by N-gram models.[1]

Byte Pair Encoding (BPE) is a key subword method. It operates as a deterministic, frequency-based merging algorithm.[1] The process begins with an initial vocabulary consisting only of individual characters and iteratively merges the most frequent adjacent symbol pairs to create longer, linguistically meaningful subwords.[1]

The practical implementation of BPE on a text corpus, such as the *Africa Galore* dataset, involves several crucial preparatory steps[1]:

- Segmentation: The initial step involves loading the corpus and splitting it into space-separated words.[1]
- Boundary Enforcement: A special end-of-word symbol, $\langle /w \rangle$, is appended to each word list.[1] This step is critical because it ensures that explicit word boundaries are maintained throughout the subsequent iterative merging process, allowing the model to distinguish between "low" in "lowly" and "low" as a standalone word.[1]
- Initialization: The initial vocabulary is populated with all unique individual characters found in the corpus, along with the special $\langle /w \rangle$ symbol.[1]
- Iterative Merging: The algorithm then repeatedly identifies the most frequent adjacent token pair (p, q) within the text and merges them into a new token pq, which is added to the vocabulary.[1] This merging continues until the predefined target vocabulary size is reached.[1]

The BPE mechanism guarantees that any rare or previously unseen word can be successfully decomposed into known subword components, effectively and structurally eliminating the hard OOV problem.[1] This crucial shift from approximation to *decomposition* ensures high fidelity of information transfer. By solving the vocabulary challenge robustly, CLMs enable the Transformer layers to focus entirely on learning complex semantic and grammatical relationships, making the resultant model inherently more robust against domain shift or novel terminology.[1]

IV. QUANTIFYING MODEL PERFORMANCE AND REPRESENTATION

A complete evaluation of language models requires combining intrinsic statistical measures of predictive accuracy with extrinsic methods that confirm the model's qualitative utility and representational quality.[1]

➢ *Perplexity (PPL) and Contextual Integrity*

- Perplexity (PPL) remains the primary intrinsic metric used for language model comparison.[1] Mathematically, it is defined as the length-normalized inverse probability of the test set. A lower PPL score indicates a superior model that is less "surprised" by the observed word sequence, reflecting higher predictive accuracy.[1] Since a zero probability assignment renders PPL calculation impossible, smoothing remains a mandatory prerequisite for all N-gram models.[1]
- Crucially, when evaluating fixed-length, attention-based models such as CLMs, the computation of PPL requires a rigorous sliding-window strategy.[1] This ensures the model utilizes its maximum available context for each token prediction, yielding a score that is both accurate and reflective of its true contextual capabilities, unlike simplistic, disjoint chunking methods.[1] Because the final PPL score is inherently sensitive to OOV handling mechanisms and vocabulary size, a direct numerical comparison between a smoothed N-gram model (which achieves fluency solely through local statistics) and a CLM (which achieves predictive depth through global context) can be potentially misleading and requires careful interpretation.[1]

➢ *Qualitative Metrics and Embedding Visualization*

Intrinsic statistical measures like PPL must be augmented by extrinsic metrics, particularly for assessing the generated text quality, long-range coherence, and relevance in application.[1]

Semantic metrics such as Coherence (logical and thematic consistency) and Fluency (grammatical correctness and naturalness) are utilized to validate the generative capacity of the CLM.[1] Furthermore, BERTScore represents an advanced evaluation metric that transcends simple lexical overlap measures (such as BLEU or ROUGE).[1] It leverages dense vector representations (embeddings) derived from pre-trained Transformer models to measure the semantic similarity between the generated output and a human reference.[1] This process offers a more nuanced assessment of conceptual alignment than traditional N-gram match metrics, which often fail to capture meaning.[1]

In addition to output analysis, the quality of the CLM's learned internal representations can be visually confirmed using t-distributed stochastic neighbor embedding (t-SNE).[1] This dimensionality reduction technique maps the high-dimensional token embeddings maintained within the Transformer into a two- or three-dimensional visualization space.[1] This diagnostic visualization is essential for validating the model's learning process, as it empirically confirms that tokens sharing similar semantic or grammatical functions are clustered closely together, thereby providing crucial evidence of the robustness and quality of the CLM's internal representation layer.[1]

ISSN No:-2456-2165 https://doi.org/10.38124/ijisrt/25nov578

Table 1 Quantitative and Qualitative Metrics for Language Model Evaluation

Metric Category	Key Example Metric	Primary Assessment Focus	Relevance to CLM/Edge Deployment
Language Modeling (Intrinsic)	Perplexity (PPL) [1]	Statistical fluency, predictive power of next token [1]	Baseline measure; requires sliding-window strategy for accuracy [1]
Qualitative/Semantic (Extrinsic)	Coherence, BERTScore [1]	Logical flow, semantic similarity of generative output [1]	Essential for verifying depth of context and output quality [1]
Embedding Analysis (Diagnostic)	t-SNE Visualization [1]	Consistency and structure of internal learned representations [1]	Confirms semantic grouping and quality of the Embedding layer [1]
Resource Consumption	Storage Footprint, Inference Latency [1]	Deployment feasibility, real-time capability [1]	Critical constraint for low-power, constrained embedded devices [1]

V. PRACTICAL DEPLOYMENT AND ARCHITECTURAL TRADE-OFFS

➢ *Resource Trade-Offs and CLM Superiority*

The efficiency profile of Compact Language Models is decisively superior for modern deployment scenarios, especially those involving edge computing.[1] Their dense, optimized architecture, combined with techniques like quantization, ensures low inference latency and minimal energy consumption, making real-time interaction feasible even on low-power edge devices.[1]

Conversely, the necessity for N-gram models to maintain massive, sparse count tables leads to an intractable storage bottleneck.[1] This issue demands complex and costly engineering for implementation, such as the use of custom trie-based compression.[1] This high Total Cost of Ownership (TCO), stemming from the engineering complexity required for maintenance and optimization, coupled with their fundamental inability to capture long-range linguistic context, renders N-gram models largely unsuitable for contemporary, complex edge applications.[1] The comparison reveals a fundamental difference in optimization focus: N-grams focus on compressing data distribution, while CLMs focus on compressing model weights for synergistic hardware optimization.[1]

Table 2 Comparative Architectural Features of N-gram and Compact Language Models

Feature	Statistical N-Gram Models	Compact Language Models (CLMs)
Architectural Core	Maximum Likelihood Estimation (MLE) [1]	Transformer (Self-Attention) [1]
Context Window	Limited ($N-1$ tokens) [1]	Global/Unlimited (Via Self-Attention) [1]
OOV Handling	Generic $\langle \text{unk} \rangle$ token (Information loss) [1]	Subword Decomposition (BPE) [1]
Optimization Focus	Advanced Smoothing, Trie Compression [1]	Knowledge Distillation, Pruning, Quantization [1]
Deployment Niche	Niche baselines for local statistics [1]	Specialized, low-latency edge applications [1]

➢ *Specialization and the Role of the N-Gram Baseline*

CLMs achieve outstanding performance through intense specialization, meaning they are fine-tuned for a narrow scope of tasks, such as healthcare symptom checking or compliance workflows in finance.[1] This targeted specialization yields both high accuracy and reliably low latency, which are crucial attributes in time-sensitive domains.[1]

While the practical deployment niche for the N-gram model is now narrow, it is not entirely eliminated. Research indicates that its inherent capacity to capture simple, purely localized statistical distributions can occasionally surpass the complex inductive bias of the Transformer in those specific, limited scenarios.[1] Consequently, N-grams retain a critical role as fast, high-precision baselines for highly vocabulary-limited sequence prediction tasks where computational simplicity and

ISSN No:-2456-2165 https://doi.org/10.38124/ijisrt/25nov578

absolute reliance on purely local statistics are prioritized over semantic depth.[1]

➢ *CLM Integration with the LLM Ecosystem*

The future trajectory of language modeling indicates a pronounced trend toward collaboration across different model scales.[1] CLMs are rapidly becoming essential tools for optimizing the LLM ecosystem itself, transcending their role as mere deployment endpoints.

CLMs can be efficiently used to generate large volumes of high-quality training datasets, a function that significantly lowers the dependence on expensive manual annotation during the training and fine-tuning phases of massive LLMs.[1] Furthermore, academic studies confirm that using smaller models to compute perplexity for pruning low-quality data from vast pretraining corpora can significantly improve the downstream performance of much larger models, while simultaneously achieving a reduction in required pretraining steps.[1] This function positions CLMs as a crucial Quality Control and Data Curation layer for LLM training data. This inverted use—where smaller, efficient models guide the training quality of the largest foundation models—is critical for addressing scalability, privacy, and safety challenges by ensuring a cleaner and more cost-efficient training pipeline.[1]

VI. CONCLUSION AND FUTURE WORK

➢ *Conclusion*

This comprehensive evaluation confirms that highly optimized Compact Language Models (CLMs) provide the most effective and viable architecture for high-performance NLP tasks deployed in resource-constrained edge environments.[1] The underlying Transformer framework delivers superior contextual comprehension, successfully overcoming the fundamental short-context and dependency failures inherent in N-gram models.[1]

Architecturally, the application of Byte Pair Encoding (BPE) subword tokenization grants CLMs a robust, information-preserving method for handling OOV words, thereby avoiding the contextual degradation inherent to the N-gram's generic $\langle \text{unk} \rangle$ token.[1] Although optimized N-gram models maintain a narrow niche as high-precision baselines for localized statistics, their architectural limitations concerning storage complexity, data sparsity, and context decisively favor the CLM for efficiency and depth of comprehension in modern applications.[1]

➢ *Future Work*

Future research must rigorously focus on the empirical implementation and side-by-side verification of these architectural trade-offs within realistic, operational edge computing environments.[1] This involves testing a state-of-the-art compressed N-gram model (optimized with advanced techniques like Kneser-Ney smoothing) against a highly pruned and quantized CLM (e.g., a minimal decoder-only Transformer).[1]

The experimental setup must integrate a comprehensive metric set, combining the intrinsic PPL (calculated via the required sliding-window method [1]) with semantic and diagnostic measures, specifically Coherence, BERTScore, and t-SNE visualization [1], to accurately quantify the precise relationship between model complexity, predictive power, and the resource footprint.[1] Furthermore, continued refinement of knowledge distillation and pruning techniques is necessary to optimize CLMs specifically for highly specialized, ultra-low-power domain deployment.[1] Adhering to scholarly best practices, including full disclosure of computing infrastructure, all hyperparameters, and selection criteria, is crucial for ensuring reproducibility and transparency in the results.[1]

REFERENCES

[1]. D. Jurafsky and J. H. Martin, Speech and Language Processing, 3rd ed. Prentice Hall, 2023. \cite{26}
[2]. J. Jokah, "Small Language Models (SLMs): The Rise of Efficient AI," Hugging Face Blog, 2024. \cite{26}
[3]. J. Lin and D. Klein, "Efficiently storing and querying n-gram language models," in Proc.
[4]. 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies, 2011, pp. 56–65. \cite{26}
[5]. J. Lin and D. Klein, "Efficiently storing and querying n-gram language models," in Proc.
[6]. 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies, 2011, pp. 56–65. \cite{26}
[7]. R. Dey, "Understanding Language Modeling: From N-grams to Transformer-Based Neural Models," Medium, 2023. \cite{26}
[8]. S. Behera, "A Comparative Analysis of Different LLM Evaluation Metrics," Medium, 2023. \cite{26}
[9]. F. Dernoncourt, "At what N do N-grams become counterproductive?" Stack Exchange, 2016. \cite{26}
[10]. R. Bansal, "Perplexity Metric for LLM Evaluation," Analytics Vidhya, 2025. \cite{26}
[11]. A. Srivastava and R. Prasad, "A New Look at N-gram Interpolation for Language Modeling," ACL, 2016. \cite{26}
[12]. S. Soman, "Testing & Evaluating Large Language Models (LLMs): Key Metrics and Best Practices (Part 2)," Medium, 2023. \cite{26}
[13]. T. Reddy, "A Taxonomy of LLM Evaluation Metrics," Arya.ai Blog, 2024. \cite{26}
[14]. J. Zhang, "Exploring the Inductive Biases of Transformers for Language Modeling," EMNLP 2024, 2024. \cite{26}
[15]. X. Jing and Y. Zhang, "Leveraging Small Language Models for Enhanced Training, Fine-Tuning, and Adaptation of Large Language Models," IEEE Transactions on Evolutionary Computation, 2025.

ISSN No:-2456-2165 https://doi.org/10.38124/ijisrt/25nov578

\cite{26}
[16]. V. Nguyen, "Large and Small Language Models: A Side-by-Side Comparison," Rabiloo Blog, 2024. \cite{26}
[17]. Unknown, "Small Language Models: A Business Guide," Delivering Data Analytics, 2024. \cite{26}
[18]. Unknown, "SLM vs LLM: Which is Right for Your Business?" Weka.IO, 2024. \cite{26}
[19]. Unknown, "Small Language Models: The Future of Efficient AI," Aisera, 2024. \cite{26}
[20]. H. Wang and K. Singh, "The impact of tokenization in genomic language models," bioRxiv, 2024. \cite{26}
[21]. F. Chiusano, "Two Minutes NLP: A Taxonomy of Tokenization Methods," Medium, 2022. \cite{26}
[22]. H. Huggingface, "Tokenizer Summary," Hugging Face Documentation, 2024. \cite{26}
[23]. S. Som, "Byte Pair Encoding vs Unigram Tokenization: A Deep Dive into Subword Models," Medium, 2022. \cite{26}
[24]. J. Lin, "Simple Template of IEEEtran.cls for IEEE Journals by Jinwei Lin," IEEE Journals, 2023. \cite{26}
[25]. J. Lin, "Simple Template of IEEEtran.cls for IEEE Journals by Jinwei Lin," IEEE Journals, 2023. \cite{26}
[26]. W.J. Book, "Modelling design and control of flexible manipulator arms: A tutorial review," in Proc. 29th IEEE Conf. on Decision and Control, San Francisco, CA, 1990, 500-506. \cite{26}
[27]. D.S. Chan, "Theory and implementation of multidimensional discrete systems for signal processing," doctoral diss., Massachusetts Institute of Technology, Cambridge, MA, 1978. \cite{26}

➢ *Works Cited*

[28]. Plagiarism Free Writing Techniques: Avoiding Common Pitfalls in Research Writing - San Francisco Edit, accessed on November 7, 2025, https://www.sfedit.net/plagiarism-free-writing-techniques-avoiding-common-pitfalls-in-research-writing/
[29]. How to Write a Plagiarism-Free Research Paper or Thesis - Papergen AI, accessed on November 7, 2025, https://www.papergen.ai/blog/how-to-write-a-plagiarism-free-research-paper-or-thesis
[30]. How to Avoid Plagiarism | Harvard Guide to Using Sources, accessed on November 7, 2025, https://usingsources.fas.harvard.edu/how-avoid-plagiarism-0
[31]. Best Practices to Avoid Plagiarism - Purdue OWL, accessed on November 7, 2025, https://owl.purdue.edu/owl/avoiding_plagiarism/best_practices.html
[32]. IOSR Manuscript Preparation Guidelines | PDF - Scribd, accessed on November 7, 2025, ((https://www.scribd.com/document/768600584/IOSR-Manuscript-Preparation-Guidelines))
[33]. Paper preparation guidelines for IOSR Journal of Engineering, accessed on November 7, 2025, https://ternaengg.ac.in/equinox2018/PaperFormat.pdf
[34]. Manuscript Preparation Guidelines (2 Page) | PDF | Abstract (Summary) | Paragraph - Scribd, accessed on November 7, 2025, https://www.scribd.com/document/98842041/Manuscript-Preparation-Guidelines-2-Page
[35]. IOSR Journal of Computer Engineering (IOSR-JCE) Template - International Organization of Scientific Research - SciSpace, accessed on November 7, 2025, https://scispace.com/formats/international-organization-of-scientific-research/iosr-journal-of-computer-engineering-iosr-jce/489e0da8074e4cfc8b861a6709e6969f
[36]. Paper Template - IOSR Journal, accessed on November 7, 2025, ((https://www.iosrjournals.org/doc/Paper%20Template.doc))
[37]. N-gram Language Models - Stanford University, accessed on November 7, 2025, https://web.stanford.edu/~jurafsky/slp3/3.pdf
[38]. Word n-gram language model - Wikipedia, accessed on November 7, 2025, ((https://en.wikipedia.org/wiki/Word_n-gram_language_model))
[39]. Transformer (deep learning architecture) - Wikipedia, accessed on November 7, 2025, ((https://en.wikipedia.org/wiki/Transformer_(deep_learning_architecture)
[40]. Small Language Models (SLM): A Comprehensive Overview - Hugging Face, accessed on November 7, 2025, https://huggingface.co/blog/jjokah/small-language-model
[41]. SLM vs LLM: The Key Differences - WEKA, accessed on November 7, 2025, https://www.weka.io/learn/ai-ml/slm-vs-llm/
[42]. What Are Small Language Models (SLMs)? A Practical Guide - Aisera, accessed on November 7, 2025, https://aisera.com/blog/small-language-models/
[43]. Large and small language models: A side-by-side comparison - Rabiloo, accessed on November 7, 2025, https://rabiloo.com/blog/large-and-small-language-models-a-side-by-side-comparison
[44]. Understanding Language Modeling: From N-grams to Transformer-based Neural Models | by Roshmita Dey | Medium, accessed on November 7, 2025, https://medium.com/@roshmitadey/understanding-language-modeling-from-n-grams-to-transformer-based-neural-models-d2bdf1532c6d
[45]. LLM Transformer Model Visually Explained - Polo Club of Data Science, accessed on November 7, 2025, https://poloclub.github.io/transformer-explainer/
[46]. Comparing the Effect of Smoothing and N-gram Order - Scholarship Repository @ Florida Tech, accessed on November 7, 2025, https://repository.fit.edu/cgi/viewcontent.cgi?article=1712&context=etd

ISSN No:-2456-2165 https://doi.org/10.38124/ijisrt/25nov578

[47]. Faster and Smaller N-Gram Language Models - ACL Anthology, accessed on November 7, 2025, https://aclanthology.org/P11-1027.pdf
[48]. Faster and Smaller N-Gram Language Models - The Berkeley NLP Group, accessed on November 7, 2025, http://nlp.cs.berkeley.edu/pubs/Pauls-Klein_2011_LM_paper.pdf
[49]. Summary of the tokenizers - Hugging Face, accessed on November 7, 2025, https://huggingface.co/docs/transformers/en/tokenizer_summary
[50]. Predictive Incremental Parsing Helps Language Modeling - ACL Anthology, accessed on November 7, 2025, https://aclanthology.org/C16-1026.pdf
[51]. Byte Pair Encoding vs. Unigram Tokenization: A Deep Dive into Subword Models - Medium, accessed on November 7, 2025, https://medium.com/@hexiangnan/byte-pair-encoding-vs-unigram-tokenization-a-deep-dive-into-subword-models-4963246e9a34
[52]. Two minutes NLP — A Taxonomy of Tokenization Methods | by Fabio Chiusano - Medium, accessed on November 7, 2025, https://medium.com/nlplanet/two-minutes-nlp-a-taxonomy-of-tokenization-methods-60e330aacad3
[53]. Arnab Sen Paper.docx
[54]. Can Transformers Learn n-gram Language Models? - ACL Anthology, accessed on November 7, 2025, https://aclanthology.org/2024.emnlp-main.550.pdf
[55]. A Comparison of Tokenization Impact in Attention Based and State Space Genomic Language Models | bioRxiv, accessed on November 7, 2025, https://www.biorxiv.org/content/10.1101/2024.09.09.612081v2.full-text
[56]. A Comparative analysis of different LLM Evaluation Metrics | by Satyadeep Behera - Medium, accessed on November 7, 2025, https://medium.com/@satyadeepbehera/a-comparative-analysis-of-different-llm-evaluation-metrics-98395c3d8e79
[57]. Perplexity Metric for LLM Evaluation - Analytics Vidhya, accessed on November 7, 2025, https://www.analyticsvidhya.com/blog/2025/04/perplexity-metric-for-llm-evaluation/
[58]. How to evaluate a text generation model: strengths and limitations of popular evaluation metrics - The Analytics Lab, accessed on November 7, 2025, https://theanalyticslab.nl/how-to-evaluate-a-text-generation-model-strengths-and-limitations-of-popular-evaluation-metrics/
[59]. LLM Evaluation: 15 Metrics You Need to Know, accessed on November 7, 2025, https://arya.ai/blog/llm-evaluation-metrics
[60]. Testing & Evaluating Large Language Models (LLMs): Key Metrics and Best Practices Part-2, accessed on November 7, 2025, https://medium.com/@sumit.somanchd/testing-evaluating-large-language-models-llms-key-metrics-and-best-practices-part-2-0ac7092c9776
[61]. Small Language Models: A Business Leader's Guide to Affordable, Task-Tuned Al, accessed on November 7, 2025, https://deliveringdataanalytics.com/small-language-models-business-guide/
[62]. The Rise of Small Language Models - IEEE Computer Society, accessed on November 7, 2025, ((https://www.computer.org/csdl/magazine/ex/2025/01/10897262/24uGPS4TUQO))

➢ *Works Cited*

[63]. Arnab Paper 2 (1).docx
[64]. The State of Large Language Models for African Languages: Progress and Challenges, accessed on November 10, 2025, https://arxiv.org/html/2506.02280v3
[65]. Transformer (deep learning architecture) - Wikipedia, accessed on November 10, 2025, https://en.wikipedia.org/wiki/Transformer_(deep_learning_architecture
[66]. Visualizing Embeddings With t-SNE - Kaggle, accessed on November 10, 2025, https://www.kaggle.com/code/colinmorris/visualizing-embeddings-with-t-sne
[67]. Understanding Transformer Models in ML - Medium, accessed on November 10, 2025, https://medium.com/@pacosun/the-architecture-that-changed-ai-5b588a4e2cb9
[68]. Boundless Byte Pair Encoding: Breaking the Pre-tokenization Barrier - arXiv, accessed on November 10, 2025, https://arxiv.org/html/2504.00178v1
[69]. Perplexity of fixed-length models - Hugging Face, accessed on November 10, 2025, https://huggingface.co/docs/transformers/perplexity
[70]. t-distributed stochastic neighbor embedding - Wikipedia, accessed on November 10, 2025, https://en.wikipedia.org/wiki/T-distributed_stochastic_neighbor_embedding
[71]. Perplexity-Based Data Pruning With Small Reference Models - OpenReview, accessed on November 10, 2025, https://openreview.net/forum?id=1GTARJhxtq
[72]. Paper Writing Best Practices - ICML 2025, accessed on November 10, 2025, https://icml.cc/Conferences/2022/BestPractices

➢ *Works Cited*

[73]. Architectural Evaluation of Subword Tokenization and Compact Language Models (CLMs) for Resource-Constrained NLP Deployment.docx