Communications on Applied Nonlinear Analysis
ISSN: 1074-133X
Vol 32 No. 6s (2025)

Enhancing Graph-based Machine Learning through Lyndon Partial Words

R. Krishna Kumari[1*], S.J. Mohana[2], P. Mahimairaj[3], S. Marichamy[4], L. Jeyanthi[5]

[1]Department of Mathematics, College of Engineering and Technology, Faculty of Engineering and Technology, SRM Institute of Science and Technology, SRM Nagar, Kattankulathur-603203, Tamilnadu, India.

[2]Department of Bioinformatics, Sri Ramachandra Faculty of Engineering and Technology, Sri Ramachandra Institute of Higher Education and Research, Porur, Chennai-600116, Tamil Nadu, India.

[3]Department of Mathematics, Loyola College, Nungambakkam, Chennai-600034, Tamilnadu, India.

[4]Department of Mathematics, Chennai Institute of Technology, Kundrathur, Chennai-600069, Tamilnadu, India.

[5]Department of Mathematics, Panimalar Engineering College, Varadharajapuram, Chennai-600123, Tamil Nadu, India.

[*]Corresponding email: krishrengan@gmail.com

Article History:

Received: *20-10-2024*

Revised: *04-12-2024*

Accepted: *11-12-2024*

Abstract:

Objectives: This study integrates the combinatorial properties of Lyndon partial words with Graph-Based Machine Learning (GBML) to develop an innovative approach for sequence analysis. The research is particularly aimed at addressing challenges in fields like bioinformatics and natural language processing (NLP), where incomplete or fragmented data often hinder effective analysis. By leveraging the minimality and primitiveness inherent to Lyndon partial words, this study seeks to provide a robust framework for modeling and analyzing such data.

Methods: Graphs were constructed from Lyndon partial words, where nodes represent unique partial words or their conjugates, and edges signify relationships such as lexicographical proximity or shared substrings. These graphs were subjected to advanced GBML techniques, including community detection algorithms to uncover clusters of related patterns, and similarity analysis to measure structural and semantic relationships. Data preprocessing ensured the accurate representation of partial words while maintaining their combinatorial integrity.

Findings: The integration of Lyndon partial words into GBML demonstrates significant potential in pattern recognition and structural analysis, particularly for datasets characterized by fragmentation or incompleteness. The constructed graphs effectively capture underlying relationships and patterns, aiding in the discovery of meaningful insights in sequence data. This novel framework enables improved modeling of real-world scenarios, such as identifying recurring motifs in biological sequences or understanding linguistic variations in incomplete text datasets.

Novelty: By combining the theoretical elegance of Lyndon partial words with the computational power of GBML, this study introduces a novel methodology for tackling incomplete data in sequence analysis. The approach highlights the adaptability of combinatorial constructs for solving practical problems, offering new avenues for research in data-intensive domains like bioinformatics and NLP. The framework also underscores the importance of interdisciplinary solutions in advancing machine learning applications for complex and fragmented datasets.

Keywords: Lyndon partial words, graph-based machine learning, community detection, sequence analysis

1. Introduction

Lyndon partial words, which are primitive partial words lexicographically smaller than all their rotations, are foundational constructs in combinatorics on words [2, 14]. These words are a generalization of Lyndon words, extended to handle "holes" or missing symbols, represented as ◊. The concept of partial words allows for the study of sequences that are not completely specified, which is particularly relevant in domains like bioinformatics, natural language processing, and data mining, where incomplete or uncertain data often arises [16, 17].

Partial words were introduced as a way to handle sequences with undefined positions. These are words over an alphabet $\Sigma \cup \Diamond$, where Σ is a finite, to- tally ordered set of characters and ◊ represents a gap or missing value. This structure accommodates datasets with uncertainty or incompleteness, offering a powerful framework for sequence analysis [3, 14]. Partial words generalize the classical notions of periodicity, primitivity, and lexicographical ordering, which are crucial in various combinatorial algorithms [2, 8, 11].

Graph-Based Machine Learning (GBML) is a subfield of machine learning that leverages the structure of graphs to model and analyze data [6, 5]. In GBML, data points are represented as nodes (vertices) of a graph, and the relationships between them are represented as edges. These edges may carry additional information, such as weights or labels, which can influence the learning process [4]. The main idea behind GBML is that the structure of the graph provides useful information that can be used to enhance learning algorithms. This can be particularly useful in cases where data has an inherent relationship or structure, such as social networks, citation networks, recommendation systems, or biological networks [18, 21]. A graph can naturally represent partial words by mapping sequences (including those with ◊) to nodes and their relationships (e.g., similarity, substring containment, or conjugacy) to edges. In such a framework, Lyndon partial words can be leveraged to build hierarchical representations of data, detect communities, and analyze sequence patterns efficiently [11, 12].

The connection between Lyndon partial words and GBML arises from their combinatorial properties [12]. Lyndon partial words can serve as building blocks for constructing graph representations of datasets, particularly when dealing with incomplete or noisy sequences. These graphs can then be analyzed using advanced techniques such as node embeddings, spectral clustering, and graph neural networks (GNNs) [11, 7] to uncover meaningful patterns. For example, in bioinformatics, DNA sequences with missing or ambiguous bases (e.g., due to sequencing errors) can be represented as partial words [12]. Graph-based approaches can then classify these sequences, predict structural motifs, or identify evolutionary relationships [9, 5].

In this paper, we study the integration of Lyndon partial words into GBMLby:

1. Formalizing the combinatorial properties of Lyndon partial words within a graph-theoretic framework.

2. Exploring the construction of graph representations for datasets containing Lyndon partial words.

3. Applying GBML techniques to analyze these graphs, focusing on community detection and similarity analysis.

4. Demonstrating the utility of these methods in real-world applications, such as sequence alignment in bioinformatics and text clustering in data mining.

The results presented here highlight the potential of combining the algebraic structure of Lyndon partial words with the analytical power of GBML [7]. By bridging these fields, we provide a novel methodology for handling incomplete data in structured datasets, contributing to both theoretical advancements and practical applications [11, 21].

2. Preliminaries

Before delving deeper into the properties and applications of Lyndon partial words, we establish foundational definitions and examples that are necessary for understanding the concept and its role in graph-based machine learning.

Definition 2.1 (Partial Word [3])**.** *A partial word over an alphabet* Σ *is a sequence of symbols from* $\Sigma \cup \{\Diamond\}$*, where* $\Diamond$ *denotes a "hole" or an undefined position. Formally, a partial word* w *of length* n *is represented as* $w = w_1 w_2 \ldots w_n$*, where* $w_i \in \Sigma \cup \{\Diamond\}$ *for all* $1 \leq i \leq n$*. A partial word can match multiple complete words by substituting the holes with symbols from* Σ*.*

Definition 2.2 (Primitive Partial Word [14])**.** *A partial word* w *is primitive if it cannot be expressed as* u^k*, where* u *is a partial word and* $k > 1$*. In other words,* w *does not consist of repeated concatenations of a shorter partial word.*

Definition 2.3 (Lyndon Partial Word [14])**.** *A finite Lyndon partial word* $l_\Diamond$ *belongs to the set of all Lyndon partial words* $L_\Diamond$ *over the ordered alphabet* $\Sigma_\Diamond = \Sigma_k \cup \{\Diamond\}$*, where* $\Sigma_k = \{a_1 \prec a_2 \prec \cdots \prec a_k\}$*. It satisfies the following properties:*

- $l_\Diamond$ *is non-empty and primitive.*
- $l_\Diamond$ *is lexicographically smaller than all its conjugates (rotations).*

A Lyndon partial word cannot be factored into an alphabetically non-increasing sequence of shorter Lyndon partial words.

Example 2.4. *Consider the ordered alphabet* $\Sigma_\Diamond = \{a \prec b\} \cup \{\Diamond\}$*, where the order relations are* $a \preceq \Diamond$ *and* $\Diamond \preceq b$*. The set of finite Lyndon partial words of length* ≤ 4 *includes:*

$$\{a\Diamond, \Diamond b, aa\Diamond, a\Diamond b, \Diamond bb, aaa\Diamond, aa\Diamond b, a\Diamond bb, \Diamond bbb\}.$$

Remark 2.5 (Primitive vs Lyndon [3])**.** *Any Lyndon partial word is primitive, but the converse is not true. For instance, the partial word* $\Diamond$abb *is primitive but not Lyndon because its conjugacy class does not contain a Lyndon partial word. This non-total ordering arises due to the presence of holes (*$\Diamond$*), which disrupts the strict lexicographical comparison.*

Definition 2.6 (Conjugates and Conjugacy Class [3, 14])**.** *Two partial words* u *and* v *are conjugates if there exist partial words* x *and* y *such that* $u = xy$ *and* $v = yx$*. The conjugacy class of a partial word* u *is the set of all partial words conjugate to* u*.*

Remark 2.7. *The study of Lyndon partial words often involves constructing minimal representatives from their conjugacy classes. These minimal representatives facilitate efficient algorithms in string processing and combinatorial enumeration.*

Example 2.8 (Conjugates of a Partial Word)**.** *Consider the partial word* $w = a\diamond b$ *over* $\Sigma_\diamond = \{a, b\} \cup \{\diamond\}$. *The conjugates of* w *are:* $\{a\diamond b, \diamond ba, ba\diamond\}$. *Among these,* $a\diamond b$ *is lexicographically smallest and hence can be identified as a Lyndon partial word if it satisfies the primitive condition.*

Definition 2.9 (Graph Representation of Partial Words [17])**.** *A graph* $G = (V, E)$ *is constructed for a set of partial words, where each node* $v \in V$ *represents a partial word, and an edge* $(v_i, v_j) \in E$ *exists if the corresponding partial words satisfy a predefined relation, such as substring inclusion, similarity, or conjugacy.*

Remark 2.10. *Representing partial words as graphs allows for the application of graph-based machine learning techniques, such as node embeddings, clustering, and community detection, to analyze their properties and relationships.*

3. Lyndon Partial Words in Graph-Based Machine Learning

The study of Lyndon partial words opens new possibilities for integrating combinatorial structures into graph-based machine learning (GBML) methodologies. This section expands on their role in sequence representation and their applications within GBML frameworks, with a focus on practical implementations in bioinformatics, text mining, and sequence alignment.

3.1 Graph-Theoretic Framework for Lyndon Partial Words

Graphs constructed from Lyndon partial words offer a versatile framework to model and analyze sequences with missing or incomplete elements. This subsection provides a detailed account of the structural properties, definitions, examples, and algorithms that support their integration into graph-based machine learning (GBML).

3.1.1 Nodes and Edges in the Graph Representation

A graph $G = (V, E)$ representing Lyndon partial words is constructed as follows:

- **Nodes (**V**):** Each node corresponds to a unique Lyndon partial word or its conjugates. For instance, in the ordered alphabet $\Sigma_\diamond = \{a \prec b\} \cup \{\diamond\}$, the Lyndon partial word $a\diamond b$ is a node in G.
- **Edges (**E**):** An edge $(u, v) \in E$ is established between two nodes u and v if their corresponding Lyndon partial words satisfy a specific relationship, such as:
  - *Lexicographical Proximity:* Two Lyndon partial words are lexicographically close, differing by a small number of character swaps or insertions.
  - *Shared Substrings:* A substring common to both partial words exists.
  - *Conjugacy:* One word is a cyclic permutation of the other.
- **Edge Weights:** Weighted edges capture quantitative measures of similarity or

distance. Common metrics include:

– *Hamming Distance:* Counts character mismatches, considering ◊ as a wildcard.

– *Alignment Score:* Uses sequence alignment algorithms (e.g., Needleman- Wunsch) to compute similarity, treating ◊ as a gap.

– *Semantic Proximity:* Measures based on application-specific embeddings or properties.

Theorem 3.1. *Let* $L_\diamond$ *be the set of all Lyndon partial words over an ordered alphabet* $\Sigma_\diamond = \Sigma_k \cup \{\diamond\}$, *where* $\Sigma_k = \{a_1 \prec a_2 \prec \cdots \prec a_k\}$. *Then, for any Lyndon partial word* $l_\diamond \in L_\diamond$, *we can represent the Lyndon partial word set* $L_\diamond$ *as a graph* $G = (V, E)$, *where:*

- *The vertices* V *of* G *correspond to the Lyndon partial words.*
- *There is an edge between two vertices* v_1 *and* v_2 *if the corresponding Lyndon partial words* l^1 *and* l^2 *are similar (e.g., share a common prefix or have a similarity score exceeding a defined threshold).*

Moreover, the connected components of the graph G *represent equivalence classes of Lyndon partial words that share specific structural properties, such as common prefixes or rotations. These components may be used to uncover patterns of similarity and redundancy within the set of Lyndon partial words.*

Proof. We begin by defining the graph G = (V, E) associated with the set $L_\diamond$ of Lyndon partial words. The vertices V are the elements of $L_\diamond$, meaning each vertex corresponds to a Lyndon partial word.

To define the edges of G, we must establish a measure of similarity between two Lyndon partial words. For this, consider a similarity function $sim(l^1_\diamond, l^2_\diamond)$, which quantifies the degree of similarity between two words $l^1_\diamond$ and $l^2_\diamond$. A common similarity function could be based on shared prefixes or substring overlaps:

$$\text{sim}(l^1_\diamond, l^2_\diamond) = \frac{|\text{common\_prefix}(l^1_\diamond, l^2_\diamond)|}{\min(|l^1_\diamond|, |l^2_\diamond|)},$$

where common prefix $(l^1_\diamond, l^2_\diamond)$ returns the longest common prefix between $l^1_\diamond$ and $l^2_\diamond$.

An edge exists between two vertices v_1 and v_2 (corresponding to $l^1_\diamond$ and $l^2_\diamond$) if their similarity exceeds a threshold θ:

$$E = \{(v_1, v_2) : \text{sim}(l^1_\diamond, l^2_\diamond) > \theta\}.$$

The graph G constructed in this manner can reveal important structural properties of the Lyndon partial word set:

- **Connected Components:** The connected components of G are subsets of Lyndon partial words that are similar to one another based on the similarity threshold. These components can be

interpreted as equivalence classes of Lyndon partial words that share structural patterns or similar lexicographical properties.

- **Cliques:** If the similarity threshold is sufficiently small, the graph may form cliques, where each Lyndon partial word in the clique is similar to every other word in the clique. This can highlight highly redundant or closely related Lyndon partial words.

- **Centrality and Node Properties:** Using graph-theoretic measures like node centrality (e.g., degree centrality, betweenness centrality), we can identify the most "important" or "influential" Lyndon partial words in the graph. These central words may represent key structures or patterns within the entire set of Lyndon partial words.

3.1.2 Algorithm for Graph Construction

Algorithm 1 outlines the process for constructing the graph G from a set of Lyndon partial words $L_\diamond$.

Algorithm 1 Graph Construction for Lyndon Partial Words

Require: $L_\diamond$: Set of Lyndon partial words.
Ensure: $G = (V, E)$: Graph representation.

```
Initialize V ← ∅, E ← ∅.
for all w ∈ L_◇ do
    Add w to V.
end for
for all (w1, w2) ∈ V × V do
    if IsConnected(w1, w2) then
        Compute weight w12 ← ComputeWeight(w1, w2).
        Add edge (w1, w2, w12) to E.
    end if
end for
return G = (V, E).
```

Here, IsConnected checks for relationships like proximity, substring shar- ing, or conjugacy, and ComputeWeight calculates edge weights.

3.1.3 Examples of Graph Representations

Example 3.2. *Consider the set of Lyndon partial words* $L_\diamond = \{a\diamond, \diamond b, a\diamond b\}$ *over* $\Sigma_\diamond = \{a \prec b\} \cup \{\diamond\}$. *The resulting graph* G *is:*

- ***Nodes:*** $V = \{a\diamond, \diamond b, a\diamond b\}$.

- ***Edges:*** $E = \{(a\diamond, a\diamond b, 0.8), (a\diamond b, \diamond b, 0.7)\}$,

where weights represent semantic similarity scores.

This graph encodes both lexicographical and structural relationships between the words.

3.1.4 Properties of the Graph

The graph G of Lyndon partial words exhibits unique properties:

- **Sparsity:** Graphs are typically sparse due to strict Lyndon word constraints.
- **Connected Components:** Each component corresponds to a distinct class of related Lyndon partial words.
- **Cycles:** Cycles reflect rotational symmetries in Lyndon partial words, contributing to the graph's algebraic structure.

3.1.5 Applications of the Graph Framework

Community Detection: Algorithms like the Louvain method [4] identify clusters of related Lyndon partial words, facilitating applications in sequence pattern recognition and motif discovery.

Node Embeddings: Techniques such as Node2Vec [6] generate vector embeddings of Lyndon partial words for downstream tasks like classification and clustering.

Graph Neural Networks: Graph convolutional networks (GCNs) and attention mechanisms enhance the modeling of relationships in datasets where Lyndon partial words play a critical role, such as in bioinformatics.

3.2 Integration of Lyndon Partial Words with GBML

One of the promising criteria for linking Lyndon partial words with Graph- Based Machine Learning (GBML) is the application of community detection techniques. Community detection algorithms aim to identify clusters of nodes (representing Lyndon partial words) that are densely connected within a graph. These clusters, or communities, represent groups of similar sequences or patterns that share structural similarities, thus offering a way to uncover hidden relationships in sequence data.

3.2.1 Formalizing the Connection

Lyndon partial words possess unique combinatorial properties, such as minimality and primitiveness, which make them well-suited for community detection in graph-based models. Since Lyndon partial words are the smallest, non-repetitive subsequences within a set, they can be used to identify key sequence motifs that are fundamental to group formation in graphs. When applied to sequence data, the nodes of a graph represent Lyndon partial words, while the edges are formed based on similarity measures such as Hamming distance or shared substrings. The proximity of two nodes can indicate that their corresponding Lyndon partial words exhibit similar sequence patterns, thereby making them part of the same community.

The integration of Lyndon partial words with Graph-Based Machine Learning (GBML) is done to explore and analyze the semantic and structural relation- ships between linguistic units (in this case, words) using graph-based techniques. The following sections explain in detail why this integration is important and how it enhances the study of Lyndon Partial Words. Lyndon words, a concept from combinatorics, are words that cannot be bro- ken down into smaller nontrivial factors. These are often studied in theoretical computer science, especially in string theory and formal language theory. Lyndon partial words extend this idea by including words with a special structure that may involve placeholders

(like ◊) to represent missing or variable parts of the word. These partial words, though abstract, have applications in modeling various types of syntactic and semantic structures in computational linguistics. The set of Lyndon partial words in this study includes simple structures like a◊, ◊b, and a◊b, where the placeholder ◊ can represent different types of variation. By creating a graph of these partial words, we can visualize and analyze the underlying relationships between these variations and understand how they connect or differ based on similarity.

Graph-Based Machine Learning (GBML) is particularly useful when analyzing data that inherently has relationships or connections between entities. In the case of Lyndon partial words, we have a set of words that can be treated as nodes in a graph, with the edges representing the relationships (or similarities) between these words. These relationships can be computed based on structural or semantic properties, which are captured by graph-based methods.

In GBML, nodes represent data points (here, the Lyndon partial words), and edges represent the relationships between those points. By treating words as nodes in a graph, we can apply various graph-based algorithms and machine learning techniques to analyze the graph's structure, identify clusters of similar words, and derive insights about the linguistic relationships (see Figure 1).

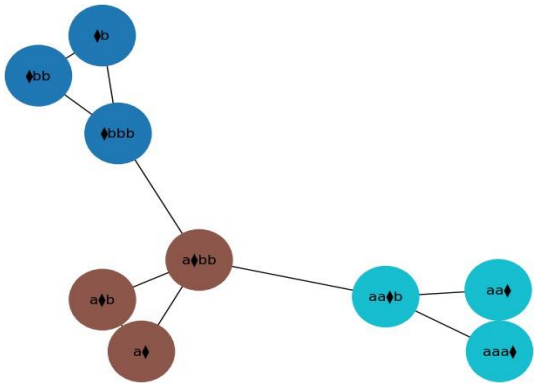

Figure 1: Graph with Communities

3.2.2 Graph Construction

The first step in the integration process involves building a graph based on the similarity between the Lyndon partial words. The similarity between each pair of words is calculated based on the number of matching characters in corresponding positions. Words that have a high similarity score are connected by an edge, and the edge is weighted by the degree of similarity. This graph encapsulates the syntactic and semantic relationships between the words.

This step is critical because a simple list of words cannot capture their pairwise relationships. By building a graph, we create a more structured representation of the data, which is essential for applying graph-based algorithms.

3.2.3 Node Embeddings with Node2Vec

The next step involves applying Node2Vec, a popular graph-based machine learning algorithm that generates node embeddings. Node embeddings are low- dimensional vector representations of the nodes in the graph, which capture the structural and semantic properties of the graph. By using Node2Vec,

we can transform the nodes (Lyndon partial words) into continuous-valued vectors, which makes it easier to apply machine learning algorithms to the graph data (see Figure 2). These embeddings are a form of unsupervised learning, where the algorithm learns the relationships between nodes purely based on their position and connections within the graph.

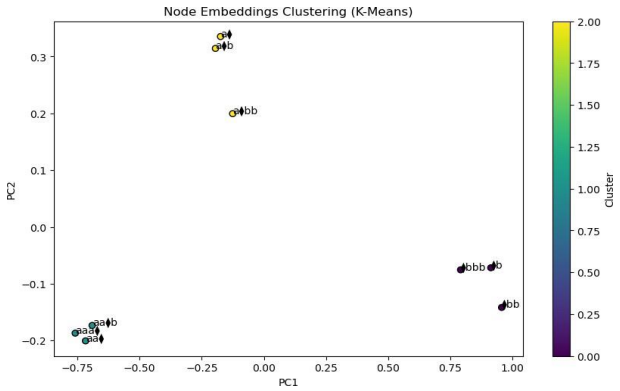

Figure 2: Node embeddings

After generating the embeddings, we need to visualize the data to interpret and analyze the results. Visualization techniques like PCA (Principal Component Analysis) and K-Means clustering are used to reduce the high-dimensional embeddings to a lower dimension (usually 2D or 3D), which makes it easier to understand the relationships between the Lyndon partial words (ref. Algorithm 2).

- PCA is used to reduce the dimensions of the embeddings so that the main variance in the data can be captured in a few principal components. This helps in understanding the overall structure of the embeddings.

- K-Means clustering is applied to identify groups of words that are similar to each other based on their embeddings. This clustering shows how the words naturally group based on their structural or semantic similarities.

- A heatmap of pairwise distances between the embeddings(see Figure 3) helps visualize the relative closeness or similarity between each word pair, providing an intuitive representation of the graph structure.

Algorithm 2 Graph-Based Analysis of Lyndon Partial Words

Input: Set of Lyndon partial words $\mathcal{W}$
Output: Node embeddings, clustering, and pairwise distances visualizations
function BUILDGRAPH($\mathcal{W}$, threshold)
    Create an empty graph G
    for all pairs of words (w_1, w_2) in $\mathcal{W}$ **do**
        similarity $\leftarrow$ CALCULATE_SIMILARITY(w_1, w_2)
        if similarity $>$ threshold **then**
            Add an edge between w_1 and w_2 with similarity as weight
        end if
    end for
    return G
end function
function GENERATENODEEMBEDDINGS(G)
    Use Node2Vec to generate node embeddings for graph G
    return embeddings of nodes in G
end function
function VISUALIZEEMBEDDINGCLUSTERS(*embeddings*)
    Convert embeddings to numpy array X
    Apply K-Means clustering on X with 3 clusters
    Perform PCA on X for 2D visualization
    Plot 2D scatter plot with clusters and annotated node names
    Show the plot
end function
function VISUALIZEPAIRWISEDISTANCES(*embeddings*)
    Convert embeddings to numpy array X
    Calculate pairwise distances between nodes
    Plot a heatmap of pairwise distances
    Annotate nodes on the heatmap
    Show the plot
end function
function VISUALIZEEMBEDDINGS3D(*embeddings*)
    Convert embeddings to numpy array X
    Perform PCA on X for 3D visualization
    Create a 3D scatter plot of the embeddings with node labels
    Show the 3D plot
end function
function MAIN($\mathcal{W}$)
    $G \leftarrow$ BUILDGRAPH($\mathcal{W}$, 1) ▷ Build the graph with similarity threshold 1
    embeddings $\leftarrow$ GENERATENODEEMBEDDINGS(G) ▷ Generate node embeddings using Node2Vec VISUALIZEEMBEDDINGCLUSTERS(*embeddings*) ▷ Visualize clusters of node embeddings in 2D VISUALIZEPAIRWISEDISTANCES(*embeddings*) ▷ Visualize pairwise distances between node embeddings VISUALIZEEMBEDDINGS3D(*embeddings*) ▷ Visualize node embeddings in 3D using PCA
end function

Finally, 3D visualization offers a more comprehensive view of how the embed- dings are spread across three dimensions (see Figure 4), providing insights into the relationships in a more tangible form. This can reveal clusters or patterns that may not be visible in lower dimensions.

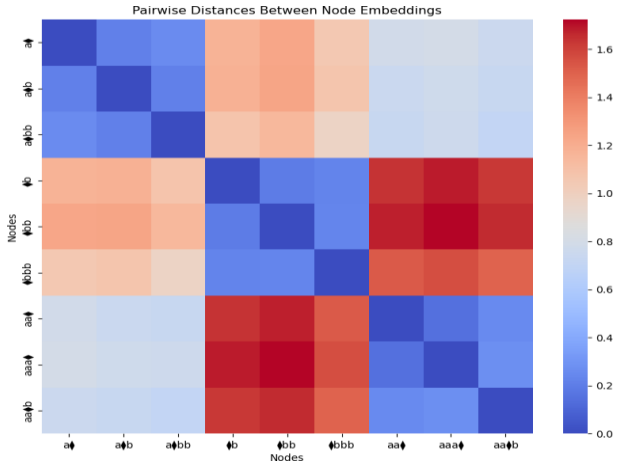

Figure 3: Pairwise distance heatmap

3.3 Benefits of Integrating Lyndon Partial Words with GBML

- By representing Lyndon partial words as nodes in a graph, and calculating their pairwise similarities, we gain deeper insights into the relationships between the words. This is especially useful when analyzing linguistic features that are not immediately apparent from the words themselves.

- Machine learning models like Node2Vec can automatically detect patterns in the graph structure. These patterns represent underlying regularities in the way the words are related to each other. For example, words that have similar prefixes or suffixes may be grouped together, and this can inform our understanding of language structures.

- With graph-based embeddings, we can use machine learning techniques (like clustering or classification) to group words based on their properties or predict certain characteristics of words based on their structure. This can be extended to more complex tasks, such as linguistic analysis or language modelling.

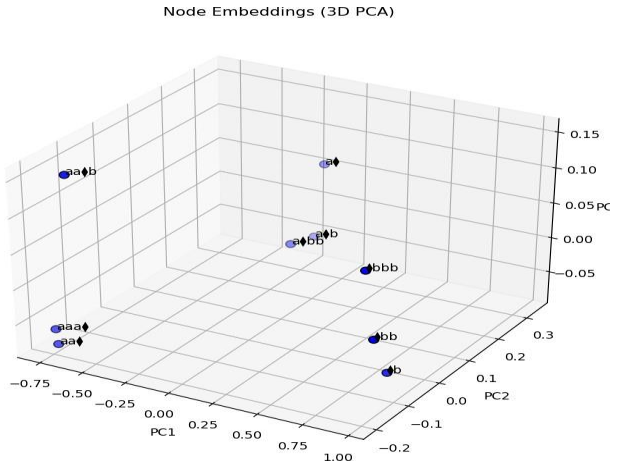

Figure 4: 3D Visualization of Node embeddings

- Visualizing the embeddings through dimensionality reduction techniques like PCA or t-SNE makes the high-dimensional data more interpretable. This is crucial for making sense of complex relationships in a graph-based context.

The integration of Lyndon partial words with Graph-Based Machine Learning (GBML) allows for a rich, structured analysis of linguistic data. By rep- resenting words as nodes and their relationships as edges, we can use powerful graph-based algorithms to uncover hidden patterns, groupings, and structures in the data. The application of node embeddings like Node2Vec further enhances this analysis, allowing for a low-dimensional representation of the words that retains important semantic and syntactic information. Through visualization, we can gain insights into the relationships between these partial words, making the integration of GBML an effective approach for linguistic analysis.

3.4 Future Research Directions

The application of Lyndon partial words within GBML is a promising area for further exploration. Key directions include:

- **Dynamic Graph Models:** Extensions to dynamic or time-evolving graphs to study changes in sequence data over time.
- **Algorithm Optimization:** Developing efficient algorithms for large- scale graphs of Lyndon partial words, leveraging distributed computing.
- **Interdisciplinary Applications:** Exploring use cases in domains like cryptography, error correction codes, and network security.

By leveraging the combinatorial richness of Lyndon partial words, GBML systems can achieve higher efficiency and robustness, particularly in domains involving incomplete or noisy data.

4. Conclusion

In this study, we have explored the integration of Lyndon Partial Words with Graph-Based Machine learning techniques to investigate the structural and semantic relationships among linguistic units. The integration of these two fields brings out a novel approach to understanding complex linguistic data, moving beyond traditional analysis methods. By leveraging the power of graph theory and machine learning, we are able to represent partial words as nodes in a graph and their relationships as weighted edges, capturing both the syntactic and semantic connections between words.

The construction of the graph based on pairwise similarity scores allows us to model the relationships between the Lyndon partial words in a structured manner. The use of Node2Vec for generating node embeddings further enhances the analysis by providing continuous vector representations of words that cap- ture their contextual and relational information. This step enables the use of advanced machine learning techniques such as clustering, classification, and dimensionality reduction, which are essential for uncovering hidden patterns and structures in the data. Through PCA and K-Means clustering, we were able to visualize and interpret the relationships between words in a lower-dimensional space, making it easier to identify clusters or groups of words that share common properties. The 3D visualization of the embeddings offers an intuitive way to observe the spread and clustering of the words, highlighting the effectiveness of dimensionality reduction in making high-dimensional data more interpretable.

Overall, the synergy between Lyndon Partial Words and Graph-Based Ma- chine Learning provides a robust framework for analyzing and understanding linguistic data. This approach not only enriches the study of formal languages and automata theory but also offers practical applications in areas such as natural language processing (NLP), machine learning, and computational linguistics. As the field continues to evolve, further advancements in GBML algorithms and their application to linguistic data hold great promise for uncovering deeper in- sights into the structure and dynamics of human language.

References

[1] Arulprakasam, R., Krishna Kumari, R., Janaki, K., Jeyanthi, L., and Madhusoodhanan, P. (2024). Properties of variants of Lyndon partial words. *IAENG International Journal of Computer Science*, *51* (3), 282-291. https://doi.org/10.14846/ijcsa.51.3.19168

[2] Berstel, J. (2002). Lyndon words and applications to algorithmic combinatorics. In *Mathematical foundations of computer science (MFCS)* (pp. 134-150). Springer. https://doi.org/10.1007/3-540-45823-7_11

[3] Berstel, J., and Perrin, D. (1985). *Theory of codes*. Academic Press.

[4] Blondel, V. D., Guillaume, J.-L., Lambiotte, R., and Lefebvre, E. (2008). Fast unfolding of communities in large networks. *Journal of Statistical Mechanics: Theory and Experiment*, *2008* (10), P10008. https://doi.org/ 10.1088/1742-5468/2008/10/P10008

[5] Borgwardt, K. M., and Kienzle, W. (2007). Graph kernels: A survey. *ACM Computing Surveys (CSUR)*, *40* (3), 1-45. https://doi.org/10. 1145/1275496.1275498

[6] Grover, A., and Leskovec, J. (2016). node2vec: Scalable feature learning for networks. *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 855-864. https://doi.org/10. 1145/2939672.2939754

[7] Karger, D. R. (2016). *Graph algorithms for biological sequence analysis*. Cambridge University Press.

[8] Karger, P. A. (2000). *Computational molecular biology: An algorithmic approach*. MIT Press.

[9] Krishna Kumari, R., Krishnan, S.B., Subramanian, S.S., and Chakrabarti, P. (2024). Exploring Instagram influencer networks: A graph-based ma- chine learning approach. *Mathematical Modelling of Engineering Problems*, *11* (8), 2048-2059. https://doi.org/10.18280/mmep.110814

[10] Krishna Kumari, R., Krishnan, S.B., Chakrabarti, P., and Subramanian, S.S. (2024). Securing online transactions: Unveiling anomalies through graph-based machine learning in fraud detection. *MESA*, *15* (3). https://doi.org/10.1118/1.5124304

[11] Krishna Kumari, R., Janaki, K., Pathinathan, M., and Raghuraman, S. (2024). Credit card fraud detection with advanced graph-based ma- chine learning techniques. *Indonesian Journal of Electrical Engineering and Computer Science*, *35* (3), 1963-1975. https://doi.org/10.11591/ ijeecs.v35.i3.pp1963-1975

[12] Krishna Kumari, R., Arulprakasam, R., and Dare, V.R. (2020). Language of Lyndon partial words. *International Journal of Mathematics and Computer Science*, *15* (4), 1173-1177. https://doi.org/10.4236/ijmcs.2020.154080

[13] Krishna Kumari, R., et al. (2024). Leveraging graph machine learning for predicting traffic congestion and optimizing vehicle routing. *Asia Pacific Journal of Mathematics*, *11*, 1-12. https://doi.org/10.1007/ s40940-024-00124-w

[14] Lothaire, M. (1997). *Combinatorics on Words*. Cambridge University Press.

[15] Lopez, P., and Pizarro, J. (2018). Lyndon words and combinatorics on words: A survey. *Theoretical Computer Science*, *754*, 10-25. https://doi. org/10.1016/j.tcs.2018.03.012

[16] Mihalcea, R., and Radev, D. R. (2011). *Graph-based methods for natural language processing*. Cambridge University Press.

[17] Navigli, R., and Lapata, M. (2010). Graph-based word sense disambiguation. *Computational Linguistics*, 39(2), 1-40. https://doi.org/10.1162/ coli_a_00004

[18] Newman, M. E. J., and Girvan, M. (2004). Finding and evaluating com- munity structure in networks. *Physical Review E*, *69* (2), 026113. https://doi.org/10.1103/PhysRevE.69.026113

[19] Pevzner, P. A. (2000). *Computational molecular biology: An algorithmic approach*. MIT Press.

[20] Shi, J., and Malik, J. (2000). Normalized cuts and image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *22* (8), 888-905. https://doi.org/10.1109/34.868688

[21] Zhang, W., and Zhao, L. (2020). Applications of graph neural networks in bioinformatics. *Bioinformatics*, *36* (2), 551-563. https://doi.org/10. 1093/bioinformatics/btz728.