Text Distance from Nested and Hierarchical Repetitions: A Compression-Based Perspective

Authors: Xiaojun Hu, Yu Liu, Yu Liu

Date: 2025-06-11T12:06:52+00:00

Abstract

We present a new method for structural sequence analysis grounded in Algorithmic Information Theory (AIT). At its core is the Ladderpath approach, which extracts nested and hierarchical relationships among repeated substructures in linguistic sequences—an instantiation of AIT's principle of describing data through minimal generative programs. These structures are then used to define three distance measures: a normalized compression distance (NCD), and two alternative distances derived directly from the Ladderpath representation. Integrated with a k-nearest neighbor classifier, these distances achieve strong and consistent performance across in-distribution, out-of-distribution (OOD), and few-shot text classification tasks. In particular, all three methods outperform both gzip-based NCD and BERT under OOD and low-resource settings. These results demonstrate that the structured representations captured by Ladderpath preserve intrinsic properties of sequences and provide a lightweight, interpretable, and training-free alternative for text modeling. This work highlights the potential of AIT-based approaches for structural and domain-agnostic sequence understanding.

Full Text

Preamble

TEXT DISTANCE FROM NESTED AND HIERARCHICAL REPETITIONS: A COMPRESSION-BASED PERSPECTIVE

Xiaojun Hu[1,2,3,**], Jing Wang[1,2,3,*], Jingwen Zhang[1,2], Fengyao Zhai[1,2,3], Xiao Xie[1,4], Zengru Di[1,2], Yu Liu[1,2,††]

[1]Department of Systems Science, Faculty of Arts and Sciences, Beijing Normal University, Zhuhai, China
[2]International Academic Center of Complex Systems, Beijing Normal University, Zhuhai, China

[3]School of Systems Science, Beijing Normal University, Beijing, China
[4]School of Physics and Astronomy, Sun Yat-sen University, Zhuhai, China

ABSTRACT

We present a new method for structural sequence analysis grounded in Algorithmic Information Theory (AIT). At its core is the Ladderpath approach, which extracts nested and hierarchical relationships among repeated substructures in linguistic sequences—an instantiation of AIT's principle of describing data through minimal generative programs. These structures are then used to define three distance measures: a normalized compression distance (NCD), and two alternative distances derived directly from the Ladderpath representation. Integrated with a k-nearest neighbor classifier, these distances achieve strong and consistent performance across in-distribution, out-of-distribution (OOD), and few-shot text classification tasks. In particular, all three methods outperform both gzip-based NCD and BERT under OOD and low-resource settings. These results demonstrate that the structured representations captured by Ladderpath preserve intrinsic properties of sequences and provide a lightweight, interpretable, and training-free alternative for text modeling. This work highlights the potential of AIT-based approaches for structural and domain-agnostic sequence understanding.

Keywords: Algorithmic Information Theory (AIT) · Normalized Compression Distance (NCD) · Compression · Ladderpath · Text Classification · Hierarchical Structure

Introduction

The rapid advancements in natural language processing (NLP) and machine learning have significantly improved the performance of text classification and regression tasks [**?**, **?**, **?**, **?**]. A wide range of approaches, from traditional bag-of-words models to deep neural architectures and pre-trained language models such as BERT [**?**], have been developed to tackle these tasks. However, deploying these models in low-resource or distributionally inconsistent scenarios remains a major challenge due to their heavy reliance on extensive annotated data and high computational demands [**?**, **?**]. Although these models demonstrate good performance under ideal conditions, their generalization ability often diminishes in settings with scarce data or in the presence of domain shifts.

Traditional models and word embedding-based classifiers require feature engineering or fine-tuning [**?**, **?**], while large-scale models encode a highly compressive mapping of data in an ultra-high-dimensional space. From an information-theoretic perspective, the process of extracting patterns from data can be viewed as a form of compression—identifying and retaining only the most informative structures [**?**, **?**]. Inspired by this insight, Jiang et al. proposed a parameter-free classification method that combines a standard compressor (e.g., gzip) with a k-nearest neighbor (k-NN) classifier to approximate deep learning-like per-

formance without training [**?**]. This method uses the normalized compression distance (NCD) to measure text similarity, offering a lightweight and generalized solution that is especially suited for low-resource and heterogeneous data.

The theoretical foundation of this approach originates from the concept of information distance and Kolmogorov complexity [**?**]. Bennett et al. introduced the notion of normalized information distance (NID), a universal similarity metric derived from Kolmogorov complexity [**?**]. However, due to the uncomputability of Kolmogorov complexity, NID cannot be directly applied in practice. To overcome this limitation, Li et al. proposed using compression algorithms to approximate complexity and introduced NCD, which serves as a computable alternative [**?**]. Cilibrasi and Vitanyi later extended this idea to clustering tasks [**?**]. By estimating the complexity of data objects through compression, NCD provides a model-free method for distance measurement, supporting training-free classification frameworks [**?**].

In recent years, compression-based techniques have shown promising results in text classification. For instance, gzip-based NCD combined with k-NN has been shown to perform effectively on low-resource datasets and, in some settings, can even outperform large-scale models like BERT [**?**]. The appeal of this method lies in its independence from extensive training or prior domain knowledge—it captures intrinsic regularities in data through general-purpose compression. However, general-purpose compressors like gzip are not optimized for semantic or hierarchical textual structures—features commonly found in natural human language—which can constrain their classification accuracy.

To address these limitations, we propose an alternative compression-based classification framework based on the Ladderpath approach [**?**, **?**], which falls under the broader framework of Algorithmic Information Theory (AIT). Ladderpath performs efficient compression by computing the minimal number of hierarchical reconstruction steps required to reproduce a given string or other data object. This allows it to capture nested structural features more effectively than traditional compressors. Unlike pre-trained models or parameter-tuned systems, Ladderpath remains both model-free and parameter-free, which significantly enhances its adaptability in dynamic or data-sparse environments [**?**, **?**]. This makes it particularly suitable for real-world scenarios characterized by low data availability or inconsistent distributions.

The main contributions of this paper are summarized as follows: (1) We propose a new approach grounded in AIT, utilizing the Ladderpath approach to extract the nested and hierarchical relationships among repeated substructures in linguistic sequences and leverage them for compression. (2) We demonstrate that these structured relationships can be used both for compression-based distance computation—yielding a new normalized compression distance, NCDlp—and for defining distances derived from the Ladderpath representation using ideas analogous to the Dice coefficient and Jaccard index, resulting in LDice and LJaccard. (3) Experiments show that all three distance measures are effective for text classification tasks. NCDlp achieves performance comparable to the

previously strongest compression-based method (NCDgzip), while LDice and LJaccard consistently outperform it. Notably, in out-of-distribution (OOD) and few-shot settings, all three methods outperform BERT. This not only provides a practical solution for scenarios with limited labeled data, but more importantly, highlights that the nested and hierarchical relationships extracted by Ladderpath capture intrinsic structural properties of sequences—enabling classification without any training.

2.1 Recap Ladderpath Approach: Capturing Nested and Hierarchical Relationships

The Ladderpath approach, which falls under the umbrella of AIT, seeks to find the shortest path for reconstructing an object (in this context, a string), with a key assumption that previously reconstructed substructures can be directly reused in subsequent steps—an idea that echoes François Jacob's notion of evolutionary tinkering [**?**, **?**, **?**, **?**]. It achieves this goal by identifying repeated substructures and the hierarchical relationships among these substructures.

We define the length of this shortest path as the ladderpath-index λ, while the hierarchical and nested relationships can be represented as a partially ordered multiset, or equivalently, a directed acyclic graph, referred to as the laddergraph (see Fig. 1a [Figure 1: see original paper] and 1b for two examples). For more detailed information about the Ladderpath approach, refer to [**?**, **?**, **?**, **?**]; here, we provide only a brief recap.

Taking the string 'ABCDBCDBCDCDEFEF' as an example, we first compute its ladderpath (the open-source code is available at [**?**]), which can be represented as a partially ordered multiset: {A, B, C, D, E, F // CD, EF // BCD(2)}. The corresponding laddergraph is shown in Fig. 1a. The ladderpath-index λ for this string can also be computed, yielding a value of 10, indicating the minimum number of steps required to reconstruct the target string.

We can naturally apply the Ladderpath approach to compression because it computes the shortest reconstruction path by algorithmically identifying repeated substructures (referred to as ladderons) and capturing their nested and hierarchical relationships. Each ladderon can be encoded in a dictionary with a unique ID, allowing us to simply reference its ID whenever it reappears (see Section 2.2 for a detailed description of the compression procedure).

[Figure 1: see original paper]

2.2 Ladderpath-Based Compressor

Fig. 1a illustrates the process of compressing a single string using the Ladderpath approach, with the previously discussed string 'ABCDBCDBCDCDEFEF' as an example. After computing its ladderpath, each ladderon is assigned a unique ID: 'BCD' receives an ID of 0, 'CD' an ID of 1, and 'EF' an ID of 2 (the higher the ID number, the lower the hierarchical level and typically the shorter

the ladderon). Referring to Fig. 1b, starting from the highest ID and moving downward, 'EF' comprises the basic building blocks 'E' and 'F', thus represented directly as (E,F). Similarly, 'CD' consists of the basic building blocks 'C' and 'D', denoted as (C,D). The ladderon 'BCD' is composed of 'B' and ladderon 1 (namely, 'CD'), and is therefore represented as (B,1). This construction reduces the number of unique symbols: we no longer need to write the full sequence 'B', 'C', 'D', effectively compressing one character.

Next, for the target string, we assign it a negative ID, here noted as -1, and represent it as (A,0,0,0,1,2,2). Every time ID 0 appears, we avoid rewriting 'BCD', thereby saving two characters per occurrence. Since ID 0 occurs three times, a total of six characters is saved. Similarly, each occurrence of ID 1 saves one character, thus two occurrences save two characters, and so forth. This strategy of eliminating redundant rewrites is the basis of compression in the Ladderpath approach.

Now, we can write all this information into a single sequence. The final compression using the Ladderpath approach can be represented as z = (1; A,0,0,0,1,2,2; B,1; C,D; E,F) where the first number indicates the total number of target strings—in this case, 1. The first semicolon-separated section encodes the target string using ladderon IDs. The following sections, also separated by semicolons, define all the ladderons: 'B,1' corresponds to ladderon ID 0, 'C,D' to ID 1, and 'E,F' to ID 2.

Excluding the initial number 1, the total length of this compressed string z is 13, which equals λ plus the total number of ladderons. In this example, λ equals 10, and there are a total of 3 ladderons. This can be clearly demonstrated since λ itself represents the shortest number of steps required to reconstruct the target string from basic units. The compressed string directly reflects this minimal reconstruction path. Note that the number of ladderons (3 in this case) is added because combining n ladderons involves only (n - 1) steps; for example, forming 'EF' from 'E' and 'F' is counted as one step, but in the compressed sequence we must explicitly write two characters, 'E' and 'F'. Similarly, constructing 'BCD' from 'B' and ladderon 1 is one step, but we must write both 'B' and 1 in the compressed output. Finally, Fig. 1b demonstrates the compression of two target strings using the same approach.

In principle, we can further compress z into a binary sequence, for example, by using Huffman coding or converting the final compressed sequence into another format (see Appendix A.1 for more details). However, we have not pursued these additional steps here, as our primary objective is to define a distance measure and subsequently perform text classification tasks based on Ladderpath compression. In summary, as the ladderpath-index λ is defined as the length of the shortest reconstruction path derived from the hierarchical and nested relationships among repeated substrings, we can simply use λ as an effective proxy for the optimally compressed length.

Finally, according to [**?**], to define NCD using a compressor, the compressor

should satisfy the following four key properties: Idempotency, ensuring that duplicate data does not affect compression efficiency; Monotonicity, requiring that compressing multiple strings does not yield a smaller result than compressing a single string; Symmetry, indicating that compression results should be independent of the order of input data; and Distributivity, ensuring consistent processing of different string combinations regardless of data structure, order, or chunking method.

A compressor that meets these properties within an acceptable error margin is considered a normal compressor [**?**]. Not all compressors strictly satisfy these four properties. For instance, widely used compressors such as Zstandard exhibit deviations in distributivity. We have conducted systematic experiments and found that the Ladderpath-based compressor satisfies these properties within an acceptable error margin (see Appendix A.2 for details).

Normalized Compression Distance (NCD): Ladderpath-based

After verifying the effectiveness of the Ladderpath-based compressor as a normal compressor, we can use it to define the Ladderpath-based NCD. When using a compressor, the NCD between two strings, X and Y, is defined as follows [**?**, **?**]:

$$NCD_c(X,Y) = \frac{c(X,Y) - \min\{c(X), c(Y)\}}{\max\{c(X), c(Y)\}}$$

where c(X) is the length of X after it is compressed using the compressor c. By substituting in the Ladderpath-based compressor, we obtain

$$NCD_{lp}(X,Y) = \frac{\lambda'(X,Y) - \min\{\lambda'(X), \lambda'(Y)\}}{\max\{\lambda'(X), \lambda'(Y)\}}$$

where $\lambda'(X) \equiv \lambda(X) - 1$ and $\lambda'(X,Y) \equiv \lambda(X,Y) - 2$, and λ is the ladderpath-index. This small adjustment—subtracting 1 and 2—is applied to balance the operation of taking out the target string(s) during reconstruction. If n targets are taking out, then n should be subtracted. We now apply NCDlp to a text classification task, with results presented in Section 3.

2.3 Define Ladderpath-distance L

Before conducting the text classification task, we define two alternative distance measures based directly on the hierarchical and nested relationships among repeated substrings (i.e., ladderons). One of them is based on the idea behind the Dice coefficient. The similarity between two sets, P and Q, is defined as the ratio of the size of their intersection to the average size of the two sets, namely $|P \cap Q|/((|P| + |Q|)/2)$, and consequently, the distance is defined as one minus this similarity. By applying the inclusion-exclusion principle, this distance can be derived as:

$$\frac{|P \cap Q|}{\frac{|P|+|Q|}{2}} = \frac{2|P \cap Q|}{|P| + |Q|} = \frac{|P| + |Q| - |P \cup Q|}{\frac{|P|+|Q|}{2}} = \frac{|P \cup Q| - \frac{|P|+|Q|}{2}}{\frac{|P|+|Q|}{2}}$$

Thus, we can define a Ladderpath-distance based on the Dice coefficient as:

$$L_{Dice}(X, Y) = \frac{\lambda'(X, Y) - \frac{\lambda'(X)+\lambda'(Y)}{2}}{\frac{\lambda'(X)+\lambda'(Y)}{2}}$$

where $\lambda'(X)$ represents the shortest path length required to reconstruct string X individually, corresponding to the set size $|P|$ in Eq. (1), and similarly for $\lambda'(Y)$. Furthermore, $\lambda'(X, Y)$ represents the shortest path length when reconstructing strings X and Y jointly, corresponding to $|P \cup Q|$ in Eq. (1).

Alternatively, the other distance measure is based on the idea behind the Jaccard index, in which the similarity between two sets P and Q is defined as $|P \cap Q|/|P \cup Q|$. The only difference from the Dice coefficient lies in the denominator: the union size rather than the average size. Following a derivation similar to the one above, we can define the Ladderpath-distance based on the Jaccard index as:

$$L_{Jaccard}(X, Y) = \frac{\lambda'(X, Y) - \frac{\lambda'(X)+\lambda'(Y)-\lambda'(X,Y)}{2}}{\lambda'(X, Y)}$$

Note that both distance measures ensure that when X = Y, the distance is 0, and when X and Y share no common substructure, the distance is 1.

It is worth noting that the similarity or distance measures based on the Jaccard index and the Dice coefficient do exhibit some differences (although they can be transformed into one another). It is well known that the Jaccard-based distance satisfies the triangle inequality, making it a true metric. In contrast, the Dice-based distance does not satisfy the triangle inequality, and is therefore considered a semimetric version of the Jaccard distance. Nevertheless, both measures are commonly used in practice. In some cases, the Dice-based distance is even preferred because empirical evidence suggests that, compared to the Jaccard index, the Dice coefficient tends to yield higher similarity scores, especially in high-dimensional, sparse data—such as in bag-of-words models or image segmentation masks. For example, in medical image segmentation (CT, MRI), the Dice coefficient is widely used as an evaluation metric. Similarly, certain n-gram–based tasks in natural language processing employ the Dice coefficient. In deep learning applications involving medical imaging—such as Mask R-CNN and other segmentation tasks—Dice loss is extensively used as a loss function (while Jaccard loss is less common), due to its smoother gradient properties, which help with network convergence [**?**].

For the sake of completeness, we employ both distance measures in this work to perform text classification tasks.

3 Results

In the work by Jiang et al. [**?**], the authors employed NCDgzip (a gzip-based normalized compression distance), combined with a k-NN classifier, to perform text classification across three distinct scenarios: in-distribution datasets, OOD datasets, and few-shot learning settings. They demonstrated that this simple, training-free approach can achieve performance comparable to—or even surpass—that of large language models such as BERT, despite BERT being a substantially more complex model.

In this study, we employ the newly defined distances—NCDlp (Ladderpath-based normalized compression distance), LDice and LJaccard—to carry out classification tasks and compare their performance against the original NCDgzip. The results are presented in the following subsections.

3.1 For In-Distribution Datasets

We begin by cleaning the datasets, removing duplicates and contaminated entries (see Appendix A.3 for details), as duplicate data can cause the k-NN classifier to yield artificially high performance. Next, we conduct classification experiments using the newly defined distances, and the results are presented in Table 1 . For comparison, we adopt the accuracy values for TextCNN, LSTM, W2V, and BERT from the work by Jiang et al. [**?**], and use TextLength (i.e., classifying based on text length) as the baseline. It is important to note that the accuracy of NCDgzip reported in [**?**] was slightly inflated [**?**], due to the use of an uncleansed dataset and an optimistic tie-breaking strategy in the k-NN classifier (see Appendix A.4 for details). In fact, the authors later acknowledged this issue. Consequently, we recomputed the classification accuracy for NCDgzip using the same methodology but on the cleaned dataset.

As shown in Table 1, we report the results for the three newly defined distance measures: NCDlp, LDice and LJaccard. Among methods that do not involve pre-training or training, the performance of LDice and LJaccard is generally stronger; NCDlp, in most cases, performs better than NCDgzip. LDice and LJaccard yield identical values in this case—not because they are always equivalent, but because their differences are negligible under the current conditions (see Appendix A.5 for details). These results indicate that the proposed measures exhibit competitive classification performance, suggesting that the hierarchical and nested structures captured by Ladderpath approach can provide meaningful similarity representations for text classification tasks. Note that in our experiments, we set the parameter $k = 7$ for all k-NN-based methods, rather than using $k = 2$ as in the study by Jiang et al., since $k = 7$ consistently yields near-optimal performance across these methods. For comparison, we also include results for $k = 2$, as detailed in Appendix A.5.

In addition, we report the classification accuracies obtained for different values of k in Fig. 2 [Figure 2: see original paper]. The results reveal that (1) the Ladderpath-compression-based distance NCDlp surpasses NCDgzip on DBpedia and R52, performs comparably on AGNews, and is outperformed by NCDgzip only on R8; (2) the Ladderpath-distance LDice and LJaccard achieve the best performance across all four datasets—except that on R8, NCDgzip remains similarly strong.

Finally, we also present the results of the bag-of-words approach, as described in [**?**]. This method represents a conventional text classification technique that relies on word occurrence patterns rather than deep contextual understanding or direct distance-based comparison. Specifically, it follows a three-step preprocessing pipeline: (1) removing punctuation and replacing it with spaces; (2) filtering out words below a certain frequency threshold; and (3) converting all uppercase letters to lowercase. After this transformation, texts are represented as independent word vectors, which are then used to compute similarities and perform classification. Notably, we observe that this approach also yields strong performance in classification tasks. However, it is important to emphasize that its success primarily stems from an inherent form of semantic segmentation, in which words naturally serve as meaningful semantic units. In contrast, compression-based and Ladderpath-based methods operate without incorporating any prior semantic knowledge.

[Figure 2: see original paper]

3.2 For Out-of-Distribution (OOD) Datasets

OOD robustness is a major challenge for modern machine learning systems, focusing on maintaining reliable performance despite significant differences between test data and training data. Compression-based and Ladderpath-based methods offer advantages in this context, as they do not require pre-training.

Experiments were conducted on five OOD datasets, as shown in Table 2 . For comparison, we use BERT as the baseline, with values reported in the study by Jiang et al. [**?**]. The results show that LDice and LJaccard consistently outperform NCDgzip, while the performance of NCDlp and NCDgzip is comparable, with each outperforming the other in approximately half of the cases.

Note that, for the same reasons discussed earlier, the accuracy values reported in the study by Jiang et al. [**?**] were slightly inflated [**?**]. The authors later acknowledged this issue and provided revised results in follow-up comments on GitHub [**?**], from which we have taken the revised values for our comparison.

3.3 Few-Shot Setting

Few-shot classification tasks pose a significant challenge for conventional machine learning models, particularly when the number of labeled examples per class is extremely limited or when the label distribution is highly imbalanced.

While many recent few-shot methods rely on large pre-trained models or meta-learning frameworks, we explore a fundamentally different direction based on compression-based and training-free approaches. These methods, including NCDlp, LDice, and LJaccard, are naturally suited to low-resource and OOD scenarios.

To evaluate the effectiveness of different compression-based distance metrics in such low-resource conditions, we conduct experiments on five OOD datasets using the 5-shot setting. The results, presented in Table 3 , show that: (1) the traditional compression-based method NCDgzip outperforms BERT on 3 out of the 5 datasets and performs comparably to the Ladderpath-based compression method NCDlp; (2) the distance measures computed directly from Ladderpath, namely LDice and LJaccard, significantly outperform both BERT and NCDgzip. These findings suggest that pre-training-free methods—especially those based on our Ladderpath approach, which captures the nested and hierarchical relationships among repeated substructures—are better equipped to extract discriminative features from limited samples.

Beyond OOD tasks, few-shot learning is also applicable to in-distribution scenarios, particularly when aiming for lightweight models or when annotation costs are high. To further evaluate the scalability and robustness of the proposed distance measures, we carried out additional experiments under different n-shot settings (n {5, 10, 50, 100}) on three datasets: AGNews, DBpedia, and SogouNews. The results, shown in Fig. 3 [Figure 3: see original paper], indicate that the Ladderpath-based distances LDice and LJaccard markedly outperform all other methods across all tested cases. These results complement the OOD evaluation in Table 3 by illustrating performance trends as the number of labeled examples increases. We selected these three datasets because their data scales are sufficiently large to support 100-shot experiments, and they differ in average text length and language. This diversity enables a more comprehensive evaluation across heterogeneous input conditions while ensuring consistency across varying few-shot levels.

[Figure 3: see original paper]

4 Discussions

The effectiveness of text classification fundamentally depends on both the quality of feature extraction and the suitability of the classification model. Deep learning architectures—such as convolutional neural networks and pre-trained language models like BERT—have achieved remarkable success by capturing deep semantic patterns in text. However, these methods typically require large-scale annotated datasets, substantial computational resources, and often struggle with generalization under domain shift or low-resource conditions.

In contrast, the Ladderpath approach offers a lightweight, training-free alternative grounded in AIT. By directly identifying repeated substructures and capturing their nested and hierarchical relationships, Ladderpath constructs a struc-

tural representation that is both compact and informative. These relationships are then leveraged to define distance measures—either through compression-based comparison (yielding NCDlp) or via structure-inspired metrics (LDice, LJaccard)—and combined with a simple k-NN classifier. Our empirical results demonstrate that these Ladderpath-based distances consistently achieve strong classification performance, often comparable to deep learning baselines. In particular, LDice and LJaccard outperform traditional gzip-based NCD in both OOD and few-shot settings, highlighting the expressiveness and robustness of the hierarchical structure captured by Ladderpath.

We also compared our method to bag-of-words-based approaches, which typically involve aggressive preprocessing steps such as lowercasing, punctuation removal, and filtering of short or infrequent words. These procedures, while not explicitly semantic, introduce strong linguistic priors that help segment the input into semantically meaningful units. In contrast, Ladderpath retains the raw sequential order of the text while structurally compressing it based on internal repetition and recursive reuse. As such, it captures meaningful regularities in a domain-agnostic and language-independent manner—providing a structural and interpretable perspective on text compression and similarity.

We believe that Ladderpath offers a broader and more general perspective on sequence modeling. As a structural representation technique, it holds promise as a tokenization strategy that preserves nested relationships, potentially serving as a preprocessing module for downstream neural models. The core principles of Ladderpath—recursive structure reuse and information-theoretic minimality—align with emerging interests in information bottlenecks and compression within neural systems, further underscoring the relevance of compression-based learning paradigms [**?**, **?**, **?**]. Its interpretability and independence from labeled data make it particularly suited for transparent or low-resource NLP applications.

Looking forward, Ladderpath could be extended to a variety of natural language processing tasks, including structure-preserving tokenization, semantic similarity estimation, document retrieval, and anomaly detection. Such directions would not only broaden its utility but also deepen our understanding of how structural compression can inform and enhance sequence-level language processing.

Acknowledgments

This study was funded by the National Natural Science Foundation of China (Grant No. 12205012 to Y.L.) and GuangDong Basic and Applied Basic Research Foundation (Grant No. 2025A1515012923 to Y.L.).

References

[1] Xipeng Qiu, Tianxiang Sun, Yige Xu, Yunfan Shao, Ning Dai, and Xuanjing Huang. Pre-trained models for natural language processing: A survey. *Science*

China Technological Sciences, 63(10):1872–1897, 2020.

[2] Hesham Allam, Lisa Makubvure, Benjamin Gyamfi, Kwadwo Nyarko Graham, and Kehinde Akinwolere. Text classification: How machine learning is revolutionizing text categorization. *Information*, 16, 2025.

[3] Qian Li, Hao Peng, Jianxin Li, Congying Xia, Renyu Yang, Lichao Sun, Philip S Yu, and Lifang He. A survey on text classification: From traditional to deep learning. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 13(2):1–41, 2022.

[4] Fabrizio Sebastiani. Machine learning in automated text categorization. *ACM Computing Surveys (CSUR)*, 34(1):1–47, 2002.

[5] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, volume 1 (Long and Short Papers), pages 4171–4186, 2019.

[6] Mengde Yang. A survey on few-shot learning in natural language processing. In *2021 International Conference on Artificial Intelligence and Electromechanical Automation (AIEA)*, pages 294–297. IEEE, 2021.

[7] Yanxu Mao, Peipei Liu, Tiehan Cui, Congying Liu, and Datao You. Low-resource fast text classification based on intra-class and inter-class distance calculation. *arXiv preprint arXiv:2412.09922*, 2024.

[8] Dinghan Shen, Guoyin Wang, Wenlin Wang, Martin Renqiang Min, Qinliang Su, Yizhe Zhang, Chunyuan Li, Ricardo Henao, and Lawrence Carin. Baseline needs more love: On simple word-embedding-based models and associated pooling mechanisms. *arXiv preprint arXiv:1805.09843*, 2018.

[9] Jeffrey Pennington, Richard Socher, and Christopher D Manning. Glove: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543, 2014.

[10] Grégoire Delétang, Anian Ruoss, Paul-Ambroise Duquenne, Elliot Catt, Tim Genewein, Christopher Mattern, Jordi Grau-Moya, Li Kevin Wenliang, Matthew Aitchison, Laurent Orseau, et al. Language modeling is compression. *arXiv preprint arXiv:2309.10668*, 2023.

[11] William J Teahan and David J Harper. Using compression-based language models for text categorization. *Language Modeling for Information Retrieval*, pages 141–165, 2003.

[12] Zhiying Jiang, Matthew Yang, Mikhail Tsirlin, Raphael Tang, Yiqin Dai, and Jimmy Lin. "Low-resource" text classification: A parameter-free classification method with compressors. In *Findings of the Association for Computational Linguistics: ACL 2023*, pages 6810–6828, 2023.

[13] Andrei N Kolmogorov. On tables of random numbers. *Sankhyā: The Indian Journal of Statistics, Series A*, pages 369–376, 1963.

[14] C.H. Bennett, P. Gacs, Ming Li, P.M.B. Vitanyi, and W.H. Zurek. Information distance. *IEEE Transactions on Information Theory*, 44(4):1407–1423, 1998.

[15] Ming Li, Xin Chen, Xin Li, Bin Ma, and P.M.B. Vitanyi. The similarity metric. *IEEE Transactions on Information Theory*, 50(12):3250–3264, 2004.

[16] R. Cilibrasi and P.M.B. Vitanyi. Clustering by compression. *IEEE Transactions on Information Theory*, 51(4):1523–1545, 2005.

[17] Jiapeng Wang and Yihong Dong. Measurement of text similarity: a survey. *Information*, 11(9):421, 2020.

[18] Juri Opitz. Gzip versus bag-of-words for text classification. *arXiv preprint arXiv:2307.15002*, 2023.

[19] Yu Liu, Zengru Di, and Philip Gerlee. Ladderpath approach: how tinkering and reuse increase complexity and information. *Entropy*, 24(8):1082, 2022.

[20] Zecheng Zhang, Chunxiuzi Liu, Yingjun Zhu, Lu Peng, Weiyi Qiu, Qianyuan Tang, He Liu, Ke Zhang, Zengru Di, and Yu Liu. Evolutionary tinkering enriches the hierarchical and nested structures in amino acid sequences. *Physical Review Research*, 6(2):023215, 2024.

[21] Eamonn Keogh, Stefano Lonardi, and Chotirat Ann Ratanamahatana. Towards parameter-free data mining. In *Proceedings of the Tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 206–215, 2004.

[22] François Jacob. Evolution and tinkering. *Science*, 196(4295):1161–1166, 1977.

[23] Yu Liu, Cole Mathis, Michał Dariusz Bajczyk, Stuart M Marshall, Liam Wilbraham, and Leroy Cronin. Exploring and mapping chemical space with molecular assembly trees. *Science Advances*, 7(39):eabj2465, 2021.

[24] Iain G Johnston, Kamaludin Dingle, Sam F Greenbury, Chico Q Camargo, Jonathan PK Doye, Sebastian E Ahnert, and Ard A Louis. Symmetry and simplicity spontaneously emerge from the algorithmic nature of evolution. *Proceedings of the National Academy of Sciences*, 119(11):e2113883119, 2022.

[25] Hector Zenil, Narsis A Kiani, Francesco Marabita, Yue Deng, Szabolcs Elias, Angelika Schmidt, Gordon Ball, and Jesper Tegner. An algorithmic information calculus for causal discovery and reprogramming systems. *Iscience*, 19:1160–1172, 2019.

[26] Zhuoying Xu, Yingjun Zhu, Binbin Hong, Xinlin Wu, Jingwen Zhang, Mufeng Cai, Da Zhou, and Yu Liu. Correlating measures of hierarchical struc-

tures in artificial neural networks with their performance. *npj Complexity*, 1(1):15, 2024.

[27] Shu Li, Lu Peng, Liuqing Chen, Linjie Que, Wenqingqing Kang, Xiaojun Hu, Jun Ma, Zengru Di, and Yu Liu. Discovery of highly bioactive peptides through hierarchical structural information and molecular dynamics simulations. *Journal of Chemical Information and Modeling*, 64(21):8164–8175, 2024.

[28] Codes (v2.0) to calculate ladderpath of sequences. Available at https://github.com/yuernestliu/lppack.

[29] Xiaoya Li, Xiaofei Sun, Yuxian Meng, Junjun Liang, Fei Wu, and Jiwei Li. Dice loss for data-imbalanced nlp tasks. *arXiv preprint arXiv:1911.02855*, 2019.

[30] Code for paper: "low-resource" text classification: A parameter-free classification method with compressors. https://github.com/bazingagin/npc_{gzip}.

[31] Grégoire Delétang, Anian Ruoss, Paul-Ambroise Duquenne, Elliot Catt, Tim Genewein, Christopher Mattern, Jordi Grau-Moya, Li Kevin Wenliang, Matthew Aitchison, Laurent Orseau, et al. Language modeling is compression. *arXiv preprint arXiv:2309.10668*, 2023.

[32] Cole Wyeth, Dongbo Bu, Quan Yu, Wen Gao, Xingwu Liu, and Ming Li. Lossless data compression by large models. *Nature Machine Intelligence*, 2025.

[33] Hector Zenil, Narsis A Kiani, Allan A Zea, and Jesper Tegnér. Causal deconvolution by algorithmic generative models. *Nature Machine Intelligence*, 1(1):58–66, 2019.

Appendix

A.1 Implementation of Ladderpath-based Compressor

[FIGURE:S1]

Still taking the string "ABCDBCDBCDCDEFEF" as an example (Fig. 1a in the main text), after processing with the Ladderpath approach, we obtain three ladderons as Fig. S1 shows. The original string is essentially composed of basic building blocks and these ladderons.

At the operational level, we can compute it using the code we provide as follows:

```
import ladderpath as lp
import ladderpath_{tools}.compress as lp_c

X = ['ABCDBCDBCDCDEFEF']
lpjson = lp.get_{ladderpath}(X)
compressed_{list} = lp_c.compress(lpjson, SEP='@')

# The result is:
# compressed_{list} = [1, '@', 'A', 2, 2, 2, 1, 0, 0, '@', 'EF', '@', 'CD', '@', 'B', 1]
```

The result is not exactly the same as the one shown in the original text, z = (1; A,0,0,0,1,2,2; B,1; C,D; E,F), but they correspond one-to-one. Let's now explain the structure of the above 'compressed_{list}': '@' serves as a delimiter to separate different sequences and ladderons. The first element, '1', indicates that the target sequence contains only one string. The subsequent elements describe the reconstruction process of the original string, where " 'A', 2, 2, 2, 1, 0, 0" means that the basic building block 'A' is first combined with element 2 (explained below), then again with element 2, followed in turn by element 2, element 1, element 0, and element 0, until the full string is recovered. Next, three '@' symbols separate three elements. The first pair '@', 'EF', designates element 0, meaning element 0 is created by combining the basic building blocks 'E' and 'F'. The second pair, '@', 'CD', designates element 1, formed by combining the basic building blocks 'C' and 'D'. The third group, '@', 'B', 1, designates element 2, which is constructed from the basic building block 'B' and element 1. Note that the IDs shown in the 'compressed_{list}' can differ from those assigned by the Ladderpath algorithm itself, owing to implementation details.

This design is primarily motivated by operational considerations, making subsequent processing more convenient. For instance, the inclusion of explicit separators (such as '@') in the compressed list facilitates structural parsing and reconstruction. Moreover, this list can be directly used as input for entropy encoding methods—such as Huffman coding—to achieve further compression and efficient storage.

A.2 Normal Compressor

Firstly, we recap the concept of , which we need later. In the Ladderpath approach, there are three primary indices for measuring a system:

- **Ladderpath-index (λ):** The length of the shortest ladderpath for a target.
- **Size-index (S):** The length of the shortest trivial ladderpath for an object.
- **Order-index (ω):** Defined as ω(x) := S(x) - λ(x), this means that the work has been saved from combining blocks when constructing the target.

Based on these definitions, is calculated by the following formula:

$$\eta = \frac{\omega(x) - \omega_0(S)}{\omega_{\max}(S) - \omega_0(S)}$$

where ω_0(S) represents the average Order-index of all possible sequences of length S, and ω (S) represents the maximum Order-index among all sequences of length S. Through this formula, quantifies the degree of orderliness of a sequence, reflecting the structural characteristics of the system at a specific length S (details can be found in [**?**]).

A compressor that can satisfy these four properties with an error range of log(n) is called a "Normal Compressor" [**?**]:

- **Idempotency:** C(xx) = C(x), and C(A) = 0 when A = .
- **Monotonicity:** C(xy) ≥ C(x).
- **Symmetry:** C(xy) = C(yx).
- **Distributivity:** C(xy) + C(z) ≤ C(xz) + C(yz).

For Idempotency, with x = ABCDEFBCD, C(x) = λ - 1 = 6, and C(xx) = λ - 2 = 6. Combined with the reuse principle of Ladderpath, it is clear that after constructing the subsequence x, it can be reused infinitely many times without consuming additional resources. In the process of combining xx, we need to take out the sequence x twice, and this operation has nothing to do with the compression process, so we have C(xx) = λ - 2.

For Monotonicity, it is clear that Idempotency is satisfied when y = 0 or y = x in the presence of C(xy) = C(x). We focus on the cases y ≠ 0 as well as y ≠ x. We can start by going through the construction steps and reuse principles of the Ladderpath. The construction of the string xy must contain the steps necessary to construct x; thus, its cost is at least equal to the cost of constructing x. Specifically, the minimum number of steps to construct xy: λ ≥ λ , and the savings from reuse do not affect this inequality, so we can know that C(xy) ≥ C(x). Even if y introduces a new structure, its steps do not reduce the overhead of the x part, thus ensuring that Monotonicity holds.

Symmetry is the same as Idempotency. The Ladderpath compressor tends to generate x and y first when compressing objects xy and yx, so the difference between generating xy and yx is whether to take out x or y first, which is only the last step, and the resources for compressing x and y in the previous step are unchanged.

For Distributivity, there is a stronger distributive law C(xyz) + C(z) < C(xz) + C(yz), and here we continue the idea of NCD and use a slightly weaker distributive law for verification (subsequently referred to as "weak"). Firstly, it should be noted that in the experimental design, in order to encompass as much as possible the various possibilities from purely random sequences to fully homogeneous sequences and to eliminate the influence of the number of repeated substructures within the sequence on the compression efficiency, we chose = [0, 0.1, 0.2, ..., 0.95], each containing 100 strings. In Table S1, we compare the different compressors that do not satisfy the distributivity for each data group under the strong and weak constraints, respectively.

Table S1: Counts of violations under different values for various compression methods.

Across different values, nearly all compression methods satisfy the weakly relaxed distributivity of Kolmogorov complexity within the acceptable margin of error. Ladderpath demonstrates relatively greater stability, with only two string sequences failing to meet the weak distributivity. In contrast, gzip ex-

hibits three violations, while Zstandard performs the worst, with eight sequences not conforming to the property. Under the strong constraints of Kolmogorov complexity, Ladderpath's anomalous values remain in an intermediate state. Gzip shows the poorest performance, particularly when compressing disordered sequences with smaller values, where a substantial number of instances fail to satisfy the strong distributivity. Regardless of the chosen -value, LZ77 consistently achieves the best performance.

[FIGURE:S3]

To be more comprehensive, we conducted additional tests to examine the satisfaction of "strong" distributivity—the most challenging property to satisfy—namely, $C(xy) + C(z) \leq C(xz) + C(yz)$, as shown in Fig. S3. The results show that none of the tested compression algorithms can strictly satisfy the strong distributivity, with gzip and bzip2 showing significant outliers and large deviations. Ladderpath-based compressor almost completely satisfies the strong distributivity for small (e.g., less than 0.4). It almost always satisfies the strong distributivity, while typically, natural language texts have an ordering rate in this range. For strong distributivity, the better the satisfaction, the more effective their discriminative distances are. In summary, the Ladderpath-based compressor shows good properties compared to some other compressors, and outperforms traditional compressors such as gzip and Zstandard.

A.3 Data Preprocessing

In this study, we utilized the dataset provided in work by Jiang et al. [**?**]. To ensure high data quality and reliable experimental results, we implemented a thorough cleaning and screening process. First, we focused on eliminating duplicates to retain only unique records, which is crucial for maintaining the integrity of our dataset. Following that, we removed any data entries deemed contaminated to ensure that the remaining dataset was pure and unaffected by noise. While this cleaning process reduced the overall amount of data, it significantly enhanced the accuracy and robustness of our experimental outcomes.

Moreover, we adjusted the data volume of each dataset based on both quality and our computational resources. In cases where the datasets were extensive (as illustrated in Table S2), we opted to select approximately 10% of the subset for our experiments. This decision was driven by the need to manage the computational overhead associated with the lengthy original data sequences and our limited processing power. By adopting this strategy, we were able to effectively control computational costs while maintaining the representativeness of the data, facilitating the successful execution of our experiments under constrained computing conditions.

Table S2: Original data volume and data volume used in our experiments.

Since the dataset is characterized by a large amount of data and long single text, we truncate the text to split the long text into shorter segments, for the sake of

speed. The experimental results show that there is no significant difference in the compression effect between the truncated sequence and the original sequence, but the Ladderpath compression efficiency is greatly improved. Therefore, this method of truncation followed by aggregate compression is feasible and will not have much impact on the final experimental results.

In the Ladderpath method, the entire sequence is traversed to find the longest repeating subsequence. As the length of the sequence increases, the probability of multiple longer duplicates decreases. The time and space complexity of traversing the sequence to find duplicates increases exponentially. Based on this, we try to truncate the text, truncate the original long text into shorter segments 1 to 50 times, and repeat the Ladderpath compression for each truncated segment. The vertical axis of the graph shows the average value of the difference in length between each compression compared to the original text; takes the values of [0, 0.1, 0.2, ..., 0.95], each value of corresponds to 10 strings, and the horizontal axis indicates the number of text cuts.

Combined with the results in Fig. S4, the following conclusions can be drawn:

- **Low :** When is small (0-0.2), the overall mean difference fluctuates significantly with the number of truncations. It is hypothesized that when is small, there are only a limited number of repeated fragments available. The truncation operation may further disrupt the redundant repetitions that could be utilized by Ladderpath, leading to a significant response in the truncated compression difference.
- **Medium :** When takes a moderate value (0.4-0.7), the results indicate that most of the difference magnitude is at a medium level. This suggests that within this range of , the repetitiveness and randomness of text sequences are relatively balanced. An increase in the number of truncations does not result in extreme variations in compression performance; in some cases, a well-designed segmentation strategy may even moderately improve compression efficiency.
- **High (close to 1.0):** When the value is high, the text is more structured with a greater number of repeated subsequences. In this case, segmentation does not significantly enhance or degrade the compression performance of Ladderpath.

The above results show that for a text sequence of about 1000 lengths in this experiment, the split has a small effect on the compressed length. The average difference between splits and original sequences is mostly in the range of -2 to 0.5, which is negligible. Sequences with different values have different sensitivities to truncation; the larger the value and the more organized the sequence, the weaker the effect of truncation on the compression effect. In any case, we can see that the truncation of text does not significantly change the compression result, thus verifying the feasibility of reducing the complexity of the algorithm by cutting the sequence.

[FIGURE:S4]

A.4 Tie-Breaking Details in k-NN

k-NN is a widely used algorithm for classification tasks due to its intuitive approach. The fundamental idea is to classify a sample by identifying the nearest k neighbors in the training set. Based on the distribution of categories among these neighbors, the sample's category is determined through voting or a weighted decision. However, challenges arise when there is a tie in the votes among the k neighbors.

There are various strategies for dealing with this problem: (1) **Random:** The label of a randomly selected neighbor in the flat ticket category is assigned to the sample to be classified; (2) **Nearest:** Among the neighbors of the tie-ballot, rank them according to their distance from the sample to be classified and assign the label of the nearest neighbor to that sample.

After resolving the tie vote, researchers usually evaluate classification accuracy by comparing predicted labels with true labels. In the context of the study conducted by Jiang et al. [**?**] (k = 2), it did not use Random or Nearest methods when facing a tie. Instead, it directly analyzes the labels of the two neighbors to provide results on classification accuracy. Their strategies for interpreting tie votes are:

- **Optimistic:** Under this strategy, the prediction is considered correct if at least one of the two neighbors' categories matches the true category during a tie scenario.
- **Pessimistic:** This approach demands that both neighbors must agree with the true category for the prediction to be deemed correct when a tie occurs.

Table S3 and **Table S4** present the outputs from the two decision-making strategies when k = 2, with W representing wrong classifications and C representing correct classifications. The results indicate that the Optimistic strategy achieves a correctness rate of 3/4, while the Pessimistic strategy yields a significantly lower correctness rate of only 1/4. Although the original article mentions a Random method, it was not practically used. This is likely due to the fact that with k = 2, the Random choice can result in a 50% error probability, which does not meaningfully enhance performance.

Ultimately, the authors in [**?**] opted for the optimistic strategy, which contributed to the relatively high accuracy of their results (albeit slightly unrealistically high). In this study, we have chosen to implement the Nearest method for resolving the tie vote problem, which provides a fairer comparison.

Table S5 gives the comparison results on multiple datasets using Nearest and Optimistic strategies. The difference in performance of Nearest over Optimistic with k = 2 is clearly visible, which is why the data in the original article is inflated.

Table S5: Performance comparison of different decision-making methods under

k = 2.

A.5 In-Distribution Case with k = 2, and Result Interpretation

Table S6: Comparison of text classification accuracy across various methods and datasets with k = 2.

It should be noted that some results reported by Jiang et al. appear to be on the higher side. This is attributed to the fact that, as described in Appendix A.4, the k-NN classification in their study consistently adopts the most optimistic scenario. Jiang et al. have subsequently acknowledged this issue. Although they have updated certain results on their GitHub repository, the data in Table 1 of their original paper remain unaltered. Consequently, the NCDgzip values presented here are derived from our own experiments, with k = 2.

In Fig. 1, we observe a high degree of similarity in classification accuracy between the Ldice and Ljaccard. This phenomenon can be explained through the following analysis: We further examined the distance matrix comparisons between the single text and several other texts within the AGNews dataset, as illustrated in Table S7. The results indicate that for the same text pairs, the distance values calculated by the Ldice and Ljaccard methods are extremely close.

Such minimal differences in distance values have a negligible impact on the final classification accuracy when using the k-NN classification method. Therefore, despite theoretical differences between the two distance measures, they exhibit a high degree of consistency in measuring text similarity in practical applications, which in turn leads to the observed similarity in classification accuracy.

Table S7: Distances between the same train text and other test texts in AGNews dataset under two different distance measures.

A.6 Out-of-Distribution Case with k = 2

Table S8: Comparison of text classification accuracy across various methods and various OOD datasets with k = 2.

A.7 Out-of-Distribution Case in Few-Shot Setting with k = 2

Table S9: Comparison of text classification accuracy across various methods on OOD datasets in the 5-shot setting using different methods with k = 2.

Note: Figure translations are in progress. See original paper for figures.