EXPLORING THE INTERSECTION OF COMPRESSION TECHNIQUES AND TEXT CLASSIFICATION: A COMPARATIVE STUDY

Srihari Chowdadenahalli Krishnamurthy, Bukhtiar Murtaza, Piyusha Khode, Liron Dsilva, Ammar Ahmed

University Of East Anglia

ABSTRACT

Kolmogorov complexity (KC) serves as a fundamental measure of data complexity, representing the shortest program to reproduce a piece of data. However, its computation often poses challenges, leading to the use of compression algorithms like gzip and bzip2 for approximation. This paper investigates the synergy between compression techniques and text classification, utilizing compression algorithms to approximate KC for minimalist text classification. We focus on Naive Bayes as the classifier and conduct empirical evaluations to assess its efficacy. Our results demonstrate competitive performance, with Naive Bayes offering simplicity and computational efficiency. This study sheds light on the potential of compression-based text classification methodologies, contributing to the discourse on efficient solutions in machine learning and information theory.

***Index Terms*—** Kolmogorov Complexity, Machine Learning, Text Classification, Compression

1. INTRODUCTION

In the expansive landscape of information theory, Kolmogorov complexity (KC) serves as a cornerstone, offering a measure of the inherent complexity within data. It represents the shortest computer program required to reproduce a piece of data, yet its exact calculation often eludes us, prompting the need for approximation techniques that lean on compression algorithms.

Compression algorithms, such as gzip, bzip2, and LZMA, are pivotal in this pursuit, striving to encode data in its most succinct form possible. Within the Python ecosystem, libraries like zlib, gzip, and lzma provide robust support for compression and decompression tasks, equipping practitioners with versatile tools for data analysis.

However, our exploration doesn't halt at data compression. Text classification, a pivotal task in natural language processing, demands efficient categorization of textual data based on its content. Enter Naive Bayes, a venerable and versatile method renowned for its simplicity and effectiveness. Unlike complex deep learning architectures, Naive Bayes operates efficiently on datasets of varying sizes, making it an appealing choice for practitioners with limited computational resources.

Against this backdrop, our paper delves into the intersection of compression techniques and text classification. By harnessing the power of compression algorithms to approximate KC, we aim to develop a minimalist yet effective approach to text classification. Through empirical evaluation and critical analysis, we strive to unravel the intricacies of our proposed methodology, contributing to the ongoing discourse on efficient and accessible solutions in machine learning and information theory.

2. KOLMOGOROV COMPLEXITY

Kolmogorov complexity is a measure of the amount of information or complexity that resides within an object, such as a string of text or a sequence of characters or bits. It can be applied to any type of object that can be represented in a formal language or encoded in a binary format. Kolmogorov complexity is described as the length of the shortest computer program that can produce the specific object as an output, or in simpler terms, the length of the most concise form or description of the object. For a string s, its Kolomogorov complexity will be defined as $K(s)$, which is the length of the shortest program which can be produce the output s [1].

The value of Kolmogorov complexity depends on the Universal Turing machine used, thus the exact value of Kolmogorov complexity of a string can not be computed and only be approximated. To approximate this Kolmogorov complexity, Li et al. put forth a innovative version of information distance called as Normalized Compression Distance (NCD). This version uses the compressed length $C(s)$ to approximate the Kolmogorov Complexity $K(s)$ of a string s [2]. We have adopted this approach and utilized various compression algorithms and Python libraries in our own methodology to approximate Kolmogorov complexity.

3. COMPRESSION LIBRARIES

Python uses multiple compression libraries with regards to text compression. Below is the comparison of few of the compression libraries:

3.1. zlib

Zlib uses the algorithm DEFLATE to compress the data. The deflate algorithm uses Huffman coding and LZ77 compression techniques. Huffman coding was developed in the year 1950 and it is a text compression algorithm which is widely used. This technique decreases the overall length of the data by assigning code words to the frequently used characters [3]. LZ77 compression technique works by finding the sequence of the repeated data. This algorithm was developed in the year 1977 for the lossless data compression. The repetitive data is encoded as pointers to a previous reference of the data which is accompanied by the number of characters to be matched [3].

3.2. gzip

Gzip is a lossless text compression technique that uses the DEFLATE algorithm. Using the LZ77 algorithm, gzip identifies the repetition of data in the content before it is shared. The output is then given as an input to the Huffman coding for forming the tree and further compressing the data.

3.3. bz2

Bz2 also known as Bzip2 is also a compression library in Python which uses Burrows-Wheeler Transform (BWT), Run-Length Encoding (RLE), Move-to-front Transform (MTF) and Huffman coding. RLE was created for the data which contained repetitive symbols in the strings. The repeated symbols are combined as pairs, the length of the string and the symbols [3]. The BWT uses two transformations, a forward transformation which permutes the input data and a matching reverse transformation which recovers the original string from the permuted string [4]. MTF ranks the symbols based on their last usage and maintains a list of symbols. When an already existing symbol is occurred, it is moved to the front of the list. Huffman coding is them applied to further compress the text [4].

3.4. lzma

Lzma is also a lossless compression technique that is an advanced version of the LZ77 technique. LZMA is a bit-based compression technique that does not use unrelated bits together in the same context. Compared to the traditional LZ77 algorithm, LZMA has a larger dictionary which takes the advantage of the larger memory space available in the modern systems.

4. CLASSIFIERS FOR TEXT CATEGORIZATION

4.1. Support Vector Machine

The Support Vector Machine (SVM) was introduced by Vapnik in 1995 as a novel approach for solving two-class pattern recognition problems. SVM operates in a vector space where its objective is to find a decision surface that effectively separates the data into two classes. For cases where the data is linearly separable, the decision surface is represented by a hyperplane given by the equation $wx + b = 0$, where w is a vector and b is a constant learned from a training set. In scenarios where the data is not linearly separable, SVM transforms the original input data into a higher-dimensional space using a non-linear mapping. This transformation enables SVM to find a linearly separating hyperplane in the new space, thereby avoiding the increased computational complexity associated with quadratic programming. SVMs strive to maximize the boundary between positive and negative examples in the dataset by mapping the n-dimensional input space into a higher-dimensional feature space. To optimize this process, SVMs leverage various kernel methods to enhance the computation of inner numerical products [5].

4.2. Naive Bayes Classifier

The Naive Bayes classifier is a widely used text classification method known for its simplicity and reasonable performance. Despite its reliance on an unrealistic independence assumption, Naive Bayes remains a popular choice in text classification tasks. The classifier applies Bayes' Theorem under the assumption that all variables $A_1, \ldots, A_n$ within a given category C are conditionally independent of each other given C. In a probability inference task, Naive Bayes calculates the probability of a hypothesis C given observed data $A_1, \ldots, A_n$. The classifier selects the category C_j with the highest probability value $P(C_j \mid A_1, A_2, \ldots, A_n)$ as the "most probable target value". The Naive Bayes classification algorithm computes this probability using a formula that considers the joint probability of the observed data given the category C and the prior probability of C. Despite its simplicity, Naive Bayes demonstrates reasonable performance in various text classification tasks [6].

4.3. Comparing SVM and Naive Bayes Classifiers for Text Categorization

We choose the Naïve Bayes classifier over Support Vector Machines due to its computational efficiency, simplicity, and robustness to noise in text classification tasks. Naïve Bayes pull the assumption of feature independence, describing quick training and inference. It provides a balance between performance and efficiency, particularly valuable in scenarios with limited computational resources. By selecting Naïve Bayes, we aim to strike a balance between performance and efficiency, especially in text classification tasks prioritizing practicality and ease of implementation.

5. METHODOLOGY

To check whether compression techniques are better than the traditional machine learning approaches, we ran a simulation of our own. We used 'newsgroup20' dataset for our simulation. This dataset contains information of news articles classified into 20 groups. To keep our classification model simple we used only four of the 20 available groups namely : 'alt.atheism', 'talk.religion.misc', 'comp.graphics', 'sci.space'.

5.1. Naïve Bayes Classifier

Before trying to implement classification through compression, we used a popular traditional machine learning model 'Naïve Bayes Classifier' to set a benchmark for evaluation. To start with the classifier, we first imported some necessary libraries like numpy, pandas, sklearn and nltk.

5.1.1. Data Preprocessing

After importing the dataset, we split the data into training and testing using the 'train_test_split' function of the sklearn library with a test size of 25%. Next we used the nltk library to tokenise the string and CountVectorizer to vectorize the token. CountVectorizer takes the tokenized text and converts it into a matrix containing the counts of tokens.

5.1.2. Model Building

Once the preprocessing is done, we move onto building our machine learning model. To build our model, we initially train the Naïve Bayes classifier using its default parameters. To increase its accuracy we use GridSearchCv for hyperparameter tuning with cv=5. We obtain the Naïve Bayes classifier with alpha = 0.1 as the best model. Once the parameters are set, we generate a classification report and confusion matrix to evaluate our model.

5.2. Classification Through Compression

To classify text using compression, we use a similar methodology followed by Eamonn Keogh et al. in [7]. We start by importing the 'newsgroup20' dataset and use the same four groups that was used for our Naïve Bayes classifier. Next we use a part of the dataset to build a single document for each category by concatenating the texts belonging to that category. After concatenation, we compress each of the document and measure their size. To classify the document, we concatenate the remaining part of the dataset with the concatenated document of each category. We then compresses each of these documents and measure the size. We then return the label for which the increase in size of the document was the least. Here the idea is that a document belonging to the same category will have similar pattern and hence the increase in size would be the least. We used 'zlib', 'gzip', 'bz2' and 'lzma' for text compression. We obtained the best result using 'lzma' and hence we will use that to compare with our Naive Bayes classifier.

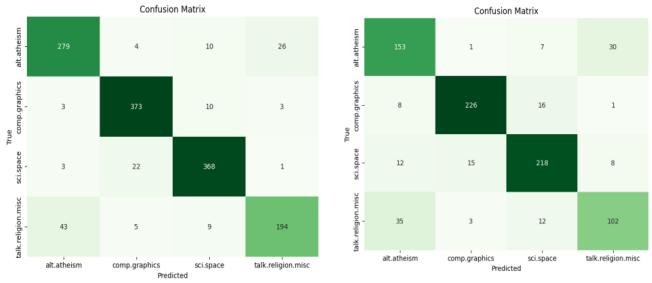

Fig. 1. Confusion Matrix of LZMA(left) and Naïve Bayes Classifier(right)

6. RESULTS

Now we discuss about the findings of our comparative study. Fig. 1. shows the confusion matrix of our classifiers. The accuracy of the Naïve Bayes classifier was 0.83 with a total computation time of 5 minutes and 47 seconds. On the other hand, the accuracy of classification from 'lzma' compression was 0.897 but the total computation time was around 50 minutes and 19 seconds which is about 10 times more than our Naïve Bayes classifier. Table 1 shows the accuracy and computation time for all the classifiers.

Table 1. Comparison of Classifier Performance

Classifier	Accuracy	Computation Time (s)
Naïve Bayes	0.83	347
LZMA	0.897	3019
GZIP	0.749	479
ZLIB	0.745	383
BZ2	0.455	634

7. CONCLUSION

In conclusion, our study delved into the intriguing intersection of compression techniques and text classification, offering valuable insights into their collaborative potential. By leveraging compression algorithms to approximate Kolmogorov complexity (KC) for minimalist text classification, we shed light on a novel approach to efficient categorization. Our exploration revealed that while compression-based methods, particularly utilizing LZMA, exhibited superior accuracy compared to the Naive Bayes (NB) classifier, NB stood out for its remarkable computational efficiency. Despite its

slightly lower accuracy, NB's swift processing time presents a compelling advantage, especially in resource-constrained environments.

This comparative study underscores the nuanced trade-offs between accuracy and computational time, highlighting the importance of considering both aspects when evaluating classification methods. While compression techniques offer promising accuracy, NB's efficiency makes it a viable alternative, particularly when speed is of the essence. Our findings contribute to the ongoing discourse on effective and accessible solutions in machine learning and information theory, emphasizing the need for a balanced approach that prioritizes both accuracy and computational efficiency.

8. BENEFITS AND DRAWBACKS

While performing our experiment on compression-based classification methods, we observed notable benefits compared to a traditional machine learning method such as Naïve Bayes. Classification through compression is relatively simper to understand and implement unless venturing further into the compression techniques. It is more of an intuitive approach and not based on complex mathematical models unlike machine learning algorithms whose understanding can be a demanding task. Additionally, compression-based classification offers more versatility than traditional classifiers as it can be applied to any form of data, irrespective of its format or language.

While compression-based classification methods offer several advantages, they also exhibit drawbacks that have to be taken in consideration. One significant limitation is compression-based classifiers operate primarily on the basis of the compressed data itself, rather than considering the semantic meaning or discriminative characteristics of the underlying features. Consequently, these methods may overlook subtle yet crucial patterns or distinctions present in the data, leading to suboptimal classification performance. Computation complexity is another factor to be considered in the implementation of this approach particularly with large scale datasets, as it is quite resource intensive and results in increased computational overhead and longer processing times, which makes it unfavourable as a chosen classifier [8]. Degradation in their performance may also be observed when compression-based techniques are applied using datasets with non-linear relationships [9].

9. REFERENCES

[1] Sihem Belabbes and Gilles Richard, "On using svm and kolmogorov complexity for spam filtering.," in *FLAIRS*, 2008, pp. 130–135.

[2] Zhiying Jiang, Matthew YR Yang, Mikhail Tsirlin, Raphael Tang, and Jimmy Lin, "With a little help from gzip: Text classification with no training," .

[3] Senthil Shanmugasundaram and L. Robert, "A comparative study of text compression algorithms," *ICTACT Journal on Communication Technology*, vol. 2, 12 2011.

[4] P. M. Fenwick, "The Burrows–Wheeler Transform for Block Sorting Text Compression: Principles and Improvements," *The Computer Journal*, vol. 39, no. 9, pp. 731–740, 01 1996.

[5] A Aizerman, "Theoretical foundations of the potential function method in pattern recognition learning," *Automation and remote control*, vol. 25, pp. 821–837, 1964.

[6] Haiyi Zhang and Di Li, "Naïve bayes text classifier," in *2007 IEEE International Conference on Granular Computing (GRC 2007)*, 2007, pp. 708–708.

[7] 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*, 2004, pp. 206–215.

[8] Edward Raff and Charles Nicholas, "An alternative to ncd for large sequences, lempel-ziv jaccard distance," in *Proceedings of the 23rd ACM SIGKDD international conference on knowledge discovery and data mining*, 2017, pp. 1007–1015.

[9] Tzu-Wei Tseng, Kai-Jiun Yang, C-C Jay Kuo, and Shang-Ho Tsai, "An interpretable compression and classification system: Theory and applications," *IEEE Access*, vol. 8, pp. 143962–143974, 2020.

[10] Beniamin Stecuła, Kinga Stecuła, and Adrian Kapczyński, "Compression of text in selected languages—efficiency, volume, and time comparison," *Sensors*, vol. 22, no. 17, 2022.

[11] C.D. Jones, A.B. Smith, and E.F. Roberts, "Article title," in *Proceedings Title*. IEEE, 2003, vol. II, pp. 803–806.

[12] Wen Zhang, Taketoshi Yoshida, and Xijin Tang, "Text classification based on multi-word with support vector machine," *Knowledge-Based Systems*, vol. 21, no. 8, pp. 879–886, 2008.

[13] Sundus Hassan, Muhammad Rafi, and Muhammad Shahid Shaikh, "Comparing svm and naïve bayes classifiers for text categorization with wikitology as knowledge enrichment," in *2011 IEEE 14th International Multitopic Conference*, 2011, pp. 31–34.

[14] Han-joon Kim, Jiyun Kim, Jinseog Kim, and Pureum Lim, "Towards perfect text classification with wikipedia-based semantic naïve bayes learning," *Neurocomputing*, vol. 315, pp. 128–134, 2018.

[15] J Randall Curtis, "Crippen d, kilcullen jk, kelly df: Three patients: International perspectives on intensive care at the end of life. boston, ma. kluwer academic publishers; 2002. 275 pp. isbn 0-7923-7671-4 (hbk)," *Critical Care*, vol. 7, pp. 1–2, 2002.

[16] Dario Benedetto, Emanuele Caglioti, and Vittorio Loreto, "Language trees and zipping," *Physical review letters*, vol. 88, no. 4, pp. 048702, 2002.

[17] Charles H Bennett, Péter Gács, Ming Li, Paul MB Vitányi, and Wojciech H Zurek, "Information distance," *IEEE Transactions on information theory*, vol. 44, no. 4, pp. 1407–1423, 1998.

[18] Yuxuan Lan and Richard Harvey, "Image classification using compression distance.," in *VVG*, 2005, pp. 173–180.