Compression is Prediction: Creating a Language Model Based on Lossless Compression

Niels Glodny [1], Samuel Leßmann [1], and Niclas Dern [2]

Abstract: This work investigates the relationship between lossless compression and language modeling. While applying large language models to compression has led to significant advances, this work explores the reverse. We develop a language model that does not rely on neural networks. Instead, it uses Zstandard, a tool designed for file compression, to create an autoregressive next-token predictor. The model is trained on a corpus of English text, achieving a perplexity significantly lower than that of a uniform model. However, the generated text lacks coherence and meaning, making it clear that this is an interesting academic illustration of the connection between information theory and intelligence, rather than a practical approach for improving language models.

Keywords: File Compression, Machine Learning, Language Modeling, Zstandard, Information Theory

1 Introduction

Recent advances in large language models have renewed interest in the old intuition that compression and intelligence are deeply connected. Hutter's Prize, which rewards data compression improvements on Wikipedia data, is motivated by the fact that being able to compress well is closely related to acting intelligently [Hu06b]. This principle suggests that compression leads to intelligence, with researchers suggesting that they are fundamentally equivalent [Hu05]. Recent advances in large language models have renewed interest in this connection, with Delétang et al. [De23] demonstrating that large language models are powerful general-purpose predictors and that the compression viewpoint provides novel insights into their scaling laws, tokenization, and in-context learning.

While significant effort has been devoted to using machine learning techniques for compression—with transformer-based models improving the state-of-the-art compression quality across domains [De23]—little work has explored the inverse direction: using compression algorithms for machine learning tasks. Notable exceptions include Jiang et al. [Ji23], who demonstrated that a combination of a simple compressor like gzip with a k-nearest-neighbor classifier achieves results that are competitive with non-pretrained deep learning methods on text classification tasks. Furthermore, Delétang et al. [De23] show an example of using an image compression algorithm for inpainting and mention the possibility

[1] Ludwig Maximilian University of Munich,
n.glodny@campus.lmu.de, https://orcid.org/0009-0007-7845-3279;
samuel.lessmann@campus.lmu.de, https://orcid.org/0009-0004-6977-9064
[2] Technical University of Munich, niclas.dern@tum.de, https://orcid.org/0009-0004-6915-6713

of doing the same for language modeling. However, due to challenges arising in a practical implementation, they do not explore this direction further.

In this work, we answer the question: *Can file compressors be used for language modeling?* We show that this is possible by building a next-token predictor that performs better than random using the Zstandard compression tool [Co16]. To be clear, the goal of this work is not creating a better language model, but rather to create an academically interesting demonstration of a well-known theoretical connection between information theory and intelligence.

2 Background

2.1 Language Modeling and Perplexity

Most language models are next-token predictors, or *autoregressive* language models. Given a sequence of tokens $x_1, x_2, ..., x_{i-1}$, they estimate the probability distribution $P(x_i|x_1, ..., x_{i-1})$ over a vocabulary $\mathcal{V}$ of tokens. To generate a text, a token is sampled iteratively from this distribution.

The standard metric for evaluating language models is perplexity (PPL), defined as the exponentiated average negative log-likelihood of a sequence [JM23]. A lower perplexity indicates that the model assigns higher probabilities to the observed sequences, suggesting better modeling performance. Intuitively, a model with perplexity k is as confused as if it had to choose uniformly among k possibilities for each token.

2.2 The Compression-Prediction Equivalence

The connection between compression and prediction is well-established and can be expressed through the lens of information theory. For a sequence with probability $P(X)$, its *information* $I(X)$ can be computed as $I(X) = -\log_2 P(X)$ bits. Thus, given the theoretical information $I(X)$ of a sequence, the probability can be reconstructed as $\hat{P}(X) = 2^{-I(X)}$ [Sh48].

Using this foundation, we interpret the length of the output of a practical compression tool $L(X)$ as an estimate of the information content $I(X)$. This is a fundamental assumption of trying to implement this connection in practice and depends highly on the compression rate achieved by the compressor. We then use the following equation for the next token, with a prefix $X = x_1, ..., x_{i-1}$ and a possible next token x_i:

$$\hat{P}(x_i|X) = \frac{\hat{P}(X, x_i)}{\hat{P}(X)} = 2^{L(X)-L(X,x_i)}. \tag{1}$$

While in theory the resulting distribution is normalized, when using the real-world length $L(X)$ instead of the theoretical information $I(X)$, it has to be normalized.

3 Methods

The general idea of our approach is to use a real-world compression tool to estimate the information content of a sequence X. Then, we apply Eq. (1) to compute the probability distribution $\hat{P}(x_i|X)$ over the next tokens x_i, which will be sampled repeatedly to generate text. The following section outlines the techniques necessary to implement this approach in practice.

Dictionary-based training using ZStandard. We use the Zstandard compression library in all our experiments. A major reason for this was that Zstandard allows the creation of so-called dictionaries that were originally designed for the efficient compression of many small files that, despite their size, share common patterns. We exploit this feature to allow the compression algorithm to learn recurring patterns within a large text corpus. Without dictionaries, we would have to compress the entire training data set many times, which would be computationally infeasible. Instead of training a single dictionary on the entire training data, we split the training set and train multiple smaller dictionaries. When computing the message length $L(X)$, the lengths using each of the dictionaries are averaged, leading to a more stable estimate.

Token representation and encoding. Like most modern language models, we operate on a tokenized input, which we generate using a byte-pair encoding (BPE) tokenizer [SHB16]. A key part of our approach is how we get from the list of tokens to the data that gets compressed. While it would be possible to concatenate the token IDs as bytes, this did not lead to a working model in our experiments. Instead, we hash the token IDs to generate a multibyte representation for each token. Since most compression algorithms only output in entire bytes, this makes it possible to measure the compression rate more accurately.

Deriving the probability distribution. While we generally follow Eq. (1) to compute the probability distribution from the message lengths, we use a significantly smaller basis than 2. During hyperparameter tuning, we noticed that this leads to a significantly lower perplexity, likely because it reduces the impact of outliers in the message lengths. To generate text from the next-token distribution, we use a top-k sampling strategy.

Implementation. Since the approach of repeatedly compressing sequences with dictionaries can become computationally expensive, we focused on an efficient implementation. To achieve this, we implemented the model in Rust and directly used the low-level C API of the Zstandard library. This way, the initialization of the dictionaries does not have to be repeated for every sequence we compress, which speeds the process up significantly allowing for larger models to be used. The implementation is available at https://github.com/SamuelLess/chat-clm.

4 Results

We evaluate our model primarily using perplexity as a measure of the model's quality. As a baseline, we include the perplexity of uniform model, which chooses each token with equal

```
the␣quick␣brown␣fox␣jumps␣to␣create␣a␣new␣middle␣of␣the␣th␣century,␣the␣th␣centur

cats␣are␣headed␣by␣the␣prime␣minister␣in␣aded␣in␣the␣began␣in␣␣as␣a␣subset␣of␣the␣

it␣has␣been␣suggested,␣for␣example␣in␣the␣catholic␣encyclopedia␣articles␣in␣the␣
        united␣states,␣the␣party␣of␣the␣united␣states.␣␣the␣traditional␣market␣
        capitalization␣of␣the␣solidarity␣with␣the␣countrys␣largest␣city␣
```

Fig. 1: Text generated by the model. The initial prefix is printed in gray, the underlines mark the token boundaries. While the text lacks coherence and meaning, it mostly uses existing words and contains some meaningful sequences.

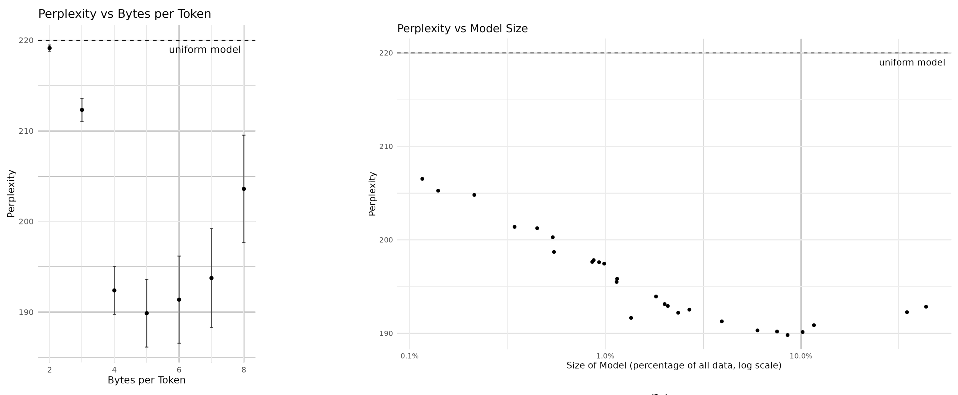
probability. This model would achieve a perplexity of exactly the size of the vocabulary $|\mathcal{V}|$, which was 220 in our experiments. For comparison, OpenAI reports that their GPT-3 model achieves a perplexity of 20.5 on a dataset of English sentences (Penn Treebank) [Br20].

Dataset. To train our model, we use the `enwik9` dataset, which is a subset of the English Wikipedia, stripped of XML metadata [Hu06a]. The dataset contains approximately 1GB of text, which is about 470 million tokens using our tokenizer. This is significantly less than the data used for state-of-the-art large language models (LLMs). The ablation studies were run on `enwik8`, which consists of the first 10% of `enwik9`. We normalized the dataset by reducing it to lowercase ASCII, using only periods and commas for punctuation.

Main result. Overall, we achieved a perplexity of 183.9 (95%-CI: [180.7, 187.2]), which is significantly lower than the 220 a uniform model would achieve. However, this is an order of magnitude higher than state-of-the-art language models achieve. The training takes about ten minutes and the resulting model can predict 5–10 tokens per second on a M1 MacBook Pro. The perplexity shows that from a statistical point of view, the model has learned some structure of the English language. However, it is susceptible to token count and sequence length, among other factors [JM23].

To get an intuitive understanding of how well the model performs, Fig. 1 shows some example text generated by the model. Overall, the text lacks coherence and the sentences are not meaningful. However, the model has clearly learned some structure of the English language. Most of the time, the tokens are arranged to form existing words, and occasionally even entire meaningful phrases of more than 10 tokens. An interesting phenomenon can be observed in the first example: Like early LLMs, the model tends to get into a loop, repeating the same phrase over and over again. This makes sense from a compression perspective, since almost no additional information is needed to increase the repetition count of a sequence.

Ablation studies. Fig. 2a shows the effect of the token encoding. The model works best with about 5 bytes per token, with a higher perplexity for shorter token encodings. This is in line with the intuition that motivated this approach and shows that almost no information can be extracted when the token representation is too short. The effect of model size on the perplexity is shown in Fig. 2b. A pattern that is often also observed in more traditional



(a) (b)

Fig. 2: Perplexity of the model on the test data. Subfigure (a) shows the effect of the bytes per token on the perplexity, demonstrating that using multiple bytes to encode tokens is crucial for the model's performance. Subfigure (b) shows the effect of the model size on the perplexity. Similar to traditional machine learning, a kind of overfitting effect can be observed.

machine learning contexts can be observed. Initially, a larger model leads to better prediction performance, but after a certain point (in our case about 10% of the training data), a type of overfitting occurs, leading to a higher perplexity on the test set. Finally, we analyzed the question of how the amount of training data impacts the results, which is shown in Fig. 3. The model size was held constant in relation to the training data size, to remove confounding by the effect seen in Fig. 2b. The experiment shows that the model benefits strongly from more training data, with the perplexity decreasing proportional to the logarithm of the training data size. This is analogous to LLMs, where similar effects have been described as scaling laws [Ka20].

5 Conclusion

In this paper, we have created a language model from just a file compression tool using the connection between information theory and statistical modeling. We used the Zstandard compression tool with its dictionary compression to create a practical implementation of this idea and trained it on a corpus of English language. While the statistics show that the model works, its quality is not comparable to more traditional approaches in language modeling. The text it generates has meaningful sequences of tokens, but can overall still be described as gibberish, as it lacks coherence. However, the goal of this paper was not to create a new state-of-the-art language model, but rather to create an interesting demonstration of the connection between compression and statistical modeling. In the analysis of the models' performance, we have observed many effects that are analogous to effects observable when using more traditional machine learning techniques. More practical research on this topic should likely focus on the other direction, i. e. using transformers to create better compression algorithms.

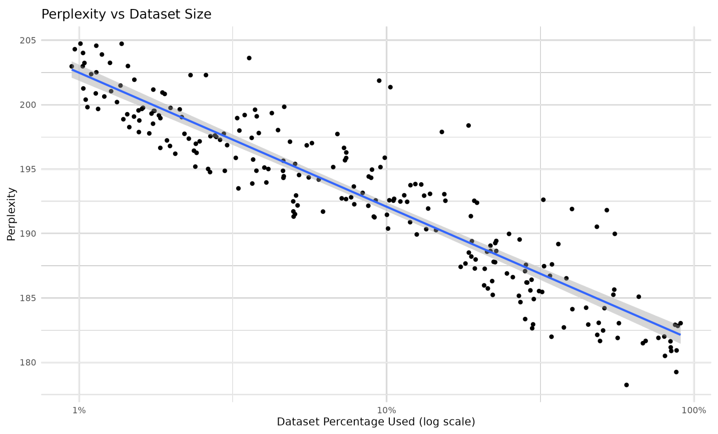


Fig. 3: Perplexity of the model as the percentage of the training dataset used. It can be seen that the model performance increases logarithmically with the amount of tokens in the training data.

References

[Br20] Brown, T. B. et al.: Language Models are Few-Shot Learners. In: Advances in Neural Information Processing Systems. Vol. 33. NeurIPS, Vancouver, Canada, 2020, https://arxiv.org/abs/2005.14165.

[Co16] Collet, Y.: Zstandard - Real-time data compression algorithm, http://facebook.github.io/zstd/, Facebook Research, 2016.

[De23] Delétang, G. et al.: Language modeling is compression. arXiv preprint arXiv:2309.10668, 2023.

[Hu05] Hutter, M.: Universal artificial intelligence: Sequential decisions based on algorithmic probability. Springer Science & Business Media, 2005.

[Hu06a] Hutter, M.: enwik9, http://mattmahoney.net/dc/textdata.html, English Wikipedia dump, first 10^9 bytes, 2006.

[Hu06b] Hutter, M.: Hutter Prize for Lossless Compression of Human Knowledge, http://prize.hutter1.net/, Accessed: 2024, 2006.

[Ji23] Jiang, Z. et al.: "Low-resource" text classification: A parameter-free classification method with compressors. In: Findings of the Association for Computational Linguistics: ACL 2023. Pp. 6810–6828, 2023.

[JM23] Jurafsky, D.; Martin, J. H.: Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition. Pearson, 2023.

[Ka20] Kaplan, J. et al.: Scaling Laws for Neural Language Models. arXiv preprint arXiv:2001.08361, 2020, arXiv: 2001.08361 [cs.LG], https://arxiv.org/abs/2001.08361.

[Sh48] Shannon, C. E.: A Mathematical Theory of Communication. Bell System Technical Journal 27 (3), pp. 379–423, 623–656, 1948.

[SHB16] Sennrich, R.; Haddow, B.; Birch, A.: Neural Machine Translation of Rare Words with Subword Units. Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (ACL), pp. 1715–1725, 2016.