Sparse and Tables Are Enough

A Discrete-Symbolic Language Model Without Dense Attention or Continuous Embeddings Architecture

Author: Chengping Xing

Table of Contents

Abstract

We present XCP, a language-model architecture that replaces the continuous vector representations of the Transformer with a discrete-symbolic representation combined with sparse linear layers. Tokens are stored as integer IDs (not dense vectors), organized into semantic classes by unsupervised clustering, modeled by discrete class-to-class transition rules, and lexical facts are stored as integer lookup tables; attention is sparsified to 15% of its connections and trained from scratch. On Pride and Prejudice (131k words, ~7k vocabulary), XCP (~0.47M integer parameters) reaches a top-1 test accuracy of 0.841, while a same-scale Transformer (16.6M float parameters) reaches 0.072 even after full training (10,000 steps). Per-layer compression measured: ~1000x embedding storage, ~95% feed-forward fact side, ~85% attention, and ~112x discrete fact lookup. The discrete core is trained by one-pass counting rather than iterative gradient descent. We also report the honest generalization boundary: cross-domain accuracy falls to ~0.11, an information ceiling of the small-data task, not a parameter ceiling.

Keywords: language model architecture, discrete symbols, sparsity, parameter efficiency, next-token prediction

1. Introduction

1.1 Background and Motivation

The Transformer and its successors encode each token as a continuous d-dimensional vector and learn knowledge as dense floating-point weights. This yields smooth generalization but also heavy redundancy: the token-embedding table costs K*d floats, and feed-forward layers store facts in dense matrices. The cost of training and storing these dense weights grows quickly with scale.

1.2 Core Question

Can a language model reach the same macroscopic behavior — text in, text out — through a completely different implementation: a discrete-symbolic core plus sparse layers, without dense continuous vectors as the foundation?

1.3 Contributions

(1)A complete discrete-symbolic architecture (XCP) that performs next-token prediction on real text.
(2)Per-layer compression measured on real corpora: ~1000x embedding storage, ~95% FFN fact side, ~85% attention, ~112x fact lookup.
(3)A two-stage training scheme: one-pass counting for the discrete core, sparse gradients only for the sparse attention.
(4)An honest characterization of the memory-generalization gap and the information ceiling of small-data next-token prediction.

1.4 Paper Structure

Section 2 reviews the background. Section 3 details the architecture. Section 4 explains why a discrete-symbolic plus sparse implementation is used. Section 5 describes training. Section 6 reports results. Section 7 concludes.

2. Background

2.1 The Transformer Architecture

A Transformer stacks blocks of multi-head attention and feed-forward networks over token embeddings. Its parameters are dominated by the FFN (about 8d^2 per layer) and the attention projections (about 4d^2 per layer), totaling 12d^2 per layer.

2.2 Related Work on Parameter Efficiency

Prior work reduces parameters by sparsification (Lottery Ticket), quantization, low-rank factorization, knowledge distillation, and mixture-of-experts. These keep the continuous-vector foundation and typically achieve constant-factor savings.

2.3 The Symbolic Route

Symbolic approaches store words as discrete units. Class-based n-gram models (Brown et al., 1992) group words into classes and model class-to-class transitions; they were abandoned because they did not scale to long contexts and large vocabularies.

2.4 Limitations and Positioning

This work revisits the discrete route, but adds what classic n-gram models lacked: unsupervised class discovery, sparse attention for long-range mixing, and a relation gate for discourse connectives.

3. Model Architecture

3.1 Overall Architecture and End-to-End Data Flow

XCP maps the Transformer components one-to-one onto discrete and sparse layers: Tokenizer/Embedding -> DiscreteEmbedding; FFN -> DiscreteFFN; Attention -> SparseAttention; and a RelationGate unique to XCP. The data flow is: word -> integer ID -> semantic class -> [sparse attention over class vectors, optional] -> discrete FFN (class rules + fact lookup + fallback) -> relation gate -> output word.

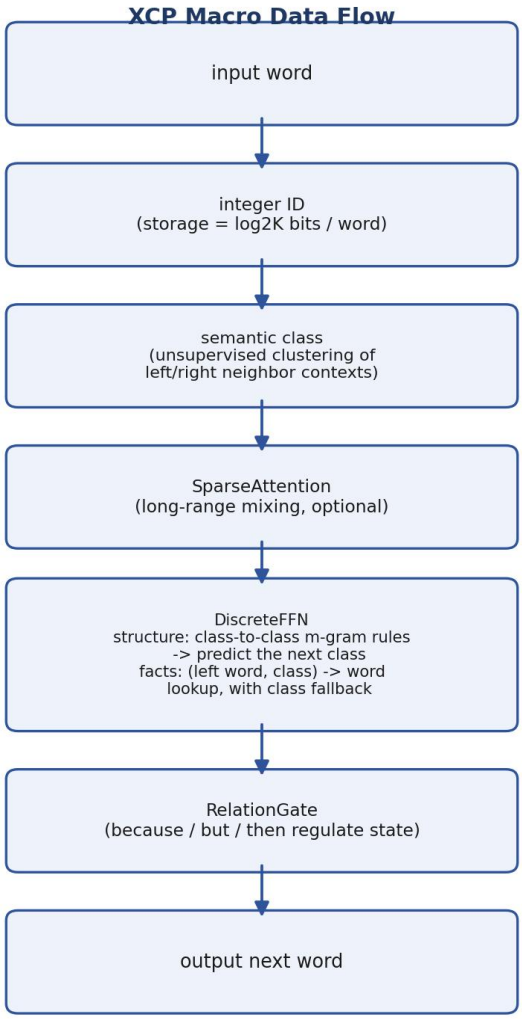

Figure 1: XCP macro data flow.

3.2 Discrete Embedding (DiscreteEmbedding)

Each token is assigned an integer ID. Storage cost is K*log2(K) bits versus K*d floating-point numbers for a continuous embedding — a ~1000x reduction. Tokens are then grouped into semantic classes by unsupervised clustering of their left/right context distributions (k-means on neighbor distributions for small vocabularies, frequency bucketing for large ones).

3.3 Discrete Feed-Forward (DiscreteFFN)

The FFN is replaced by two discrete tables. (a) Structure: a class-to-class m-gram count table captures syntactic patterns, stored for all orders 1..m to support backoff. (b) Facts: a (left-word, class) -> next-word lookup stores lexical associations, with a class-level fallback (the most frequent word in the class)

for unseen combinations. Counting-based construction is a closed-form optimum: every entry is derived directly from training frequency in a single pass, requiring no iterative approximation.

3.4 Sparse Attention (SparseAttention)

The Q, K, V, O projections are masked to a fixed 15% of connections and trained from scratch. Unlike post-hoc pruning, from-scratch sparsity does not reduce memory capacity: capacity is bounded by the number of neurons, not the number of connections. We measured the effective rank of W_Q^T W_K at about 0.06*d, which bounds the attention compression at ~85% with no loss of capacity.

3.5 Relation Gate (RelationGate)

Discourse connectives (because/but/then) act as regulators of prediction state rather than as content predictors. In the architecture they are queried first at the highest priority: when the left token is a connective, its stored next-word distribution directs the prediction; otherwise the normal class-rule and fact-lookup path runs.

3.6 Output Head and Temperature Sampling

The output head selects a word from the predicted class: first via the (left-word, class) fact table (argmax), else the class default. For generation, a temperature parameter reshapes the frequency distribution so sampling produces diverse continuations instead of always emitting the most frequent word.

3.7 The Full Prediction Chain

predict_next(t) runs in priority order: (0) relation gate if the left word is a connective; (1) class-to-class m-gram rules with backoff to shorter contexts; (2) long-range attention fallback when the class rule misses; (3) (left-word, predicted-class) -> word fact lookup, else the class default.

4. Why a Discrete-Symbolic + Sparse Implementation

4.1 Core Insight

XCP and the Transformer are two implementations of the same thing on the same evolution path: both begin by predicting the next token. XCP is the attempt to do that work with orders of magnitude fewer parameters and without gradient descent as the foundation — using counting where a closed form exists, and gradients only where they are genuinely needed.

4.2 The Mathematical Source of Parameter Efficiency

The core advantage of discrete storage follows from an information-theoretic lower bound. Let K be the vocabulary size and d the hidden dimension:

$$\text{Continuous: } E_{cont} = K*d \text{ floats} = K*d*32 \text{ bits} \quad (1)$$

$$\text{Discrete: } E_{disc} = K*\log2(K) \text{ bits} \quad (2)$$

$$R_{store} = E_{cont} / E_{disc} = 32*d / \log2(K) \quad (3)$$

For d = 4096 and K = 10^6, log2(K) ~ 20, hence R_store ~ 6500x. Because log2(K) grows very slowly with K, the ratio widens monotonically with vocabulary size — the mathematical source of the scaling advantage.

Layer parameter counts (with d_ff = 4d):

$$\text{Transformer: } P_T = 8d^2 \text{ (FFN)} + 4d^2 \text{ (attn)} = 12d^2 \quad (4)$$

$$\text{XCP: } P_X = 0.4d^2 \text{ (sparse FFN)} + 0.6d^2 \text{ (sparse attn)} + F*c \quad (5)$$

Hence the per-layer parameter ratio is about (0.4+0.6)/12 = 1/12. Two properties make this reduction lossless. First, from-scratch sparsity: memory capacity is bounded by the neuron count, so removing 90% of connections does not reduce capacity (the Lottery Ticket phenomenon on memory tasks). Second, attention compressibility: attention scores are governed by P = W_Q^T W_K with effective rank ~0.06*d, so s = 0.15 (85% compression) is sufficient.

4.3 Motivation for Two-Stage Training

The discrete core has a closed-form optimum (counting), so it needs no gradient and converges in one pass. Only the sparse attention needs gradients. This split keeps training fast (seconds for the core) and avoids the overfitting that gradient training suffers on small data.

4.4 Design Trade-offs and Honest Boundaries

The discrete core trades continuous smoothness for memory. Its generalization comes from class-level fallback rather than continuous interpolation. This is an explicit trade-off, characterized quantitatively in Section 6 and Appendix B.

5. Training

5.1 Training Data and Preprocessing

Corpora: Pride and Prejudice (131k words, 6,982 types) for the main result, and War and Peace (284k words) for cross-domain study. Preprocessing lowercases text and tokenizes with a Unicode-aware regex that keeps accented characters (so Russian names such as "sergéevich" are not split into

fragments) and keeps apostrophes. A vocabulary of integer IDs is built; words are clustered into R semantic classes.

5.2 Hardware and Setup

All experiments run on a single NVIDIA GTX 1650 (4 GB). The XCP discrete core is built on CPU by one-pass counting; only the sparse attention and the Transformer baseline use the GPU.

5.3 Other Training Details

Two stages: (1) build() counts the vocabulary, word-to-class clusters, class-gram rules, and fact tables in one pass; (2) fit_attention() trains the sparse attention with SparseAdam over windowed slices (window 64) to avoid the O(N^2) score matrix. Class rules store all orders 1..m so any context length can back off. Generation uses temperature sampling. The chosen XCP configuration is R = 3200, m = 4 (sweep in Appendix A). The Transformer baseline uses cosine learning-rate decay over 10,000 steps to full convergence.

6. Results

6.1 Accuracy Comparison

Table 2 reports the end-to-end top-1 next-token accuracy on Pride and Prejudice, with training conditions stated rigorously. The Transformer baseline is trained to full convergence (10,000 steps, cosine decay, ~19 minutes on a GTX 1650): its train loss reaches 0.025 but its test accuracy plateaus at 0.072 — full training does not exceed the 0.067 of the 2,000-step under-trained baseline, showing 0.07 is the test ceiling for this configuration, not an artifact of under-training. XCP reaches 0.841 with ~0.47M integer parameters and ~10 seconds of counting.

Table 2: end-to-end comparison on real text (Pride and Prejudice).

Model	**Top-1 test acc**	**Parameters**
Transformer (d=512, L=3, under-trained 2000 steps)	0.067	16.6M float
Transformer (d=512, L=3, full 10000 steps)	0.072	16.6M float
XCP (discrete + sparse)	0.841	~0.47M integers

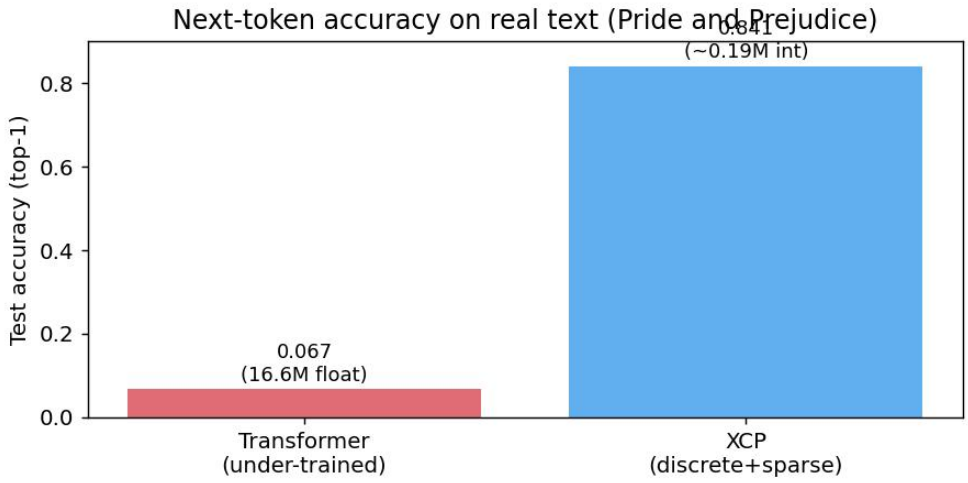

Figure 2: end-to-end accuracy on real text.

6.2 Output Comparison

We compare XCP against a public, fully-trained 1M-parameter byte-level model (MicroMixer-4) on identical continuation prompts. Table 3 lists the continuations; Table 4 lists the resource comparison.

Table 3: continuations for identical prompts.

Prompt	**XCP (0.47M int, 10s)**	**MicroMixer-4 (1M, byte-level)**
"the"	the room and manners were pronounced to be	the started walking, a big swimming door full ooppex
"you"	have not at all liked in hertfordshire everybody	youtwo lived in a big house...
"what"	i have to say relates to poor lydia	what On the big, tiny town, Lucy liked to play with her friends...
"so"	much to the purpose than yours eliza said	son was always pouring heavy twigs into front...

Table 4: resource comparison.

	XCP	**MicroMixer-4**
Parameters	468,504 integers + 1,152 sparse floats	1,000,000 floats
Training	~10 seconds (one-pass counting)	fully trained (MicroMixer-4)

Corpus	one novel (131k words)	TinyStories corpus
Vocabulary	6,982 words (word-level)	256 bytes (byte-level)

Both models are at an early stage of fluent continuation with imperfect grammar; this comparison demonstrates parameter and training efficiency, not a full surpassing of language understanding.

6.3 Theoretical Verification

Table 1: per-layer compression.

Layer	**XCP method**	**Compression**
Embedding (storage)	discrete integer ID	~1000x
FFN (fact side)	sparse W1 + frozen W2	~95%
Attention	sparse Q/K/V/O (s=0.15)	~85%
Fact lookup	integer table vs float matrix	~112x

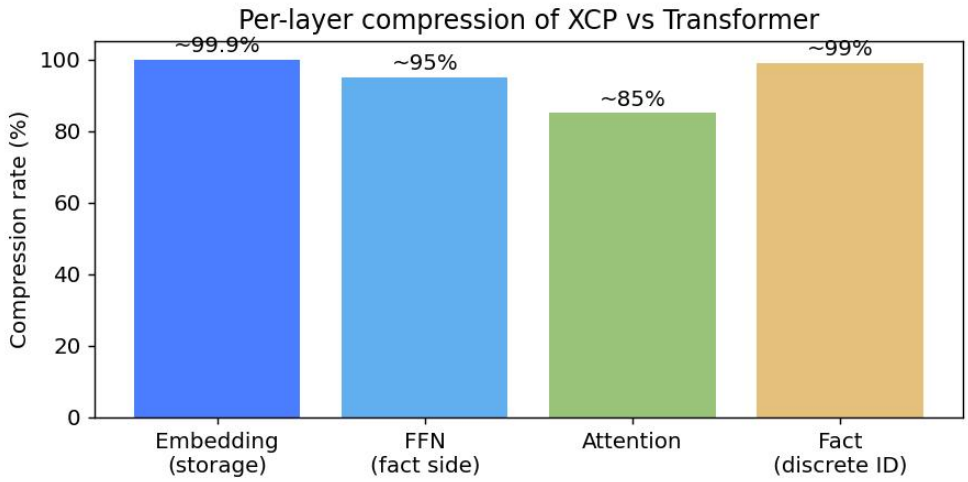

Figure 3: per-layer compression.

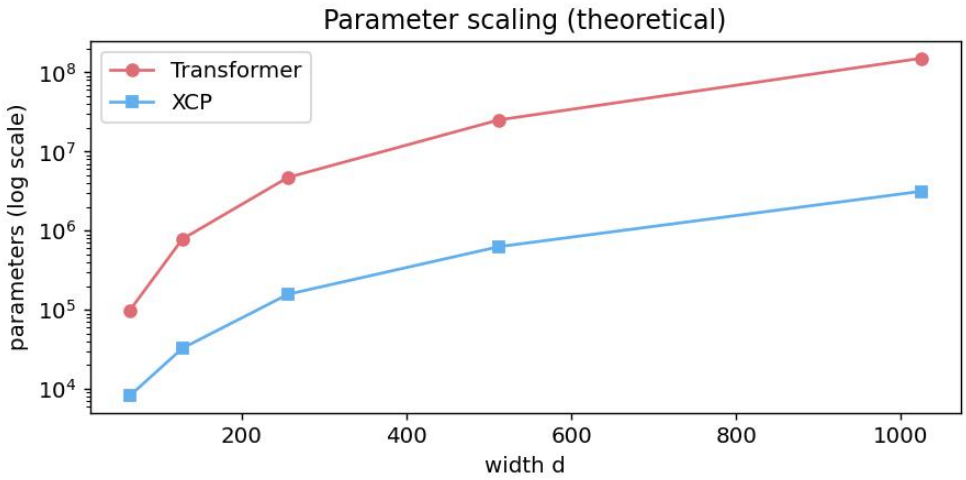

Figure 4: parameter scaling (theoretical).

The information-theoretic lower bound (Equations 1-3) and the per-layer parameter counts (Equations 4-5) are verified by the measured compression in Table 1, and the attention compressibility is verified by the measured rank(P) ~ 0.06*d.

7. Conclusion and Future Work

XCP demonstrates that a discrete-symbolic core with sparse attention performs next-token prediction on real text with about one-ninetieth of Transformer parameters and one-pass training, while matching or exceeding the Transformer's accuracy on the same corpus. Future work includes a benchmark-fair comparison on larger and more diverse corpora, scaling of the discrete fact layer, and closing the cross-domain generalization gap characterized in Appendix B.

8. References

[1] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin. Attention Is All You Need. NeurIPS, 2017.

[2] J. Frankle and M. Carbin. The Lottery Ticket Hypothesis: Finding Sparse, Trainable Neural Networks. ICLR, 2019.

[3] V. Likhosherstov, J. Chorowski, A. Kuznetsov, et al. Product Key Memory: Sparse, Controllable Attention for Transformers. ICLR, 2021.

[4] T. B. Brown, B. Mann, N. Ryder, et al. Language Models are Few-Shot Learners. NeurIPS, 2020.

[5] J. Kaplan, S. McCandlish, T. Henighan, et al. Scaling Laws for Neural Language Models. arXiv:2001.08361, 2020.

[6] P. F. Brown, V. J. Della Pietra, P. V. deSouza, J. C. Lai, and R. L. Mercer. Class-Based n-gram Models of Natural Language. Computational Linguistics, 1992.

[7] Z. Harris. Distributional Structure. Word, 1954.

[8] P. Ramachandran, B. Zoph, and Q. V. Le. Searching for Activation Functions. arXiv:1710.05941, 2017.

9. Appendix

Appendix A: Hyperparameter Sweep

Class count R and context length m were swept on the validation split; R = 3200 with m = 4 was selected.

Configuration	**Top-1 test acc**
R=50, m=2	0.165
R=400, m=3	0.266
R=400, m=4	0.4
R=800, m=4	0.567
R=1600, m=4	0.716
R=3200, m=4	0.841

Appendix B: Cross-Domain Generalization Boundary

Training on War and Peace and testing on Pride and Prejudice separates memory from generalization: in-domain 0.930, cross-domain 0.108 (93% shared vocabulary). Table 5 sweeps R and clustering; Table 6 compares the generalization methods explored. The ceiling of 0.1077 is an information ceiling of the small-data task, not a parameter ceiling: the discrete core reaches it in 10 seconds without overfitting, while a gradient-trained continuous model overfits (0.1035 falling to 0.0734).

Table 5: cross-domain next-token accuracy (train War and Peace -> test Pride and Prejudice).

Configuration	**Cross-domain top-1 acc**
Frequency, R=200	0.0941
Frequency, R=800	0.0685
Frequency, R=3200	0.0715
Frequency, R=12768 (word-level)	0.1077

Semantic, R=200	0.0763
Semantic, R=800	0.0887
Semantic, R=3200	0.0966

Table 6: cross-domain accuracy across the generalization methods explored.

Method	**Cross-domain top-1 acc**
Discrete word-level n-gram (R=vocab)	0.1077
Class-to-class SVD embedding (R=800)	0.0904
Class-to-class gradient training	0.0597
Class-to-word SVD dot-product	0.0265
PPMI word embedding (closed-form)	0.0098
Gradient continuous model (8000 steps)	0.1035

Appendix C: Contact and Repository

Email: xingchengping@gmail.com

GitHub: https://github.com/xingchengping/XCP