A Published "Fourier Language Model" Is a Retrieval Index, Not a Model: Provable Order Blindness and a Held-Out Audit of a 134 MB Chat Artefact

Daozhendong Yang
School of Intelligent Science and Engineering,
Yunnan Technology and Business University, Kunming, China
hi@yxpil.com
https://modelscope.cn/models/Yxpilow/F_Structure_LM

Abstract

A public model repository named `F_Structure_LM` advertises "a Fourier transform that replaces the self-attention of a Transformer" and shows a fluent reply to a prompt that is not in its training data. We read all 140,569,669 bytes of its payload and report what it is: 1,849,576 rows of 17 float32 numbers, each paired with one token label, plus three integers — and no learnable parameter of any kind. Inference is cosine nearest-neighbour lookup in that table. Because each row is the magnitude spectrum of raw byte-level BPE identifiers, two failure modes are theorems rather than measurements: the spectrum is invariant to time reversal, so reversing a context cannot change what is retrieved (99.05% of 4,000 held-out contexts retrieve the identical neighbour), and the zeroth bin is the unordered sum of the identifiers, carrying 24.0% of a typical key's energy. On the official LCCC valid and test splits the published index predicts the next token correctly for 2.90% of 4,000 sampled contexts — below its own constant-label baseline (5.48%) and far below a unigram over the same corpus (34.48%), against bigram 18.05%, trigram 11.63%, exact 12-gram 0.43%. Rebuilding the identical feature recipe on 50,000 sampled training dialogues, with 92.7 rows per dialogue instead of the published 37.0, lifts the same decoder to 21.63%: the artefact's failure sits in index density and label coverage, not in order blindness alone. At generation time it lands on points of its own table (mean top-1 cosine 0.998–1.000 over 896 steps), none of the three replies printed on its model card reproduce, and 32.4% of rows carry a label that cannot be decoded without an adjacent one. What it does genuinely well is cheap: 139 tokens/s and 841 MB resident on a CPU, no GPU. We close by stating what a "Fourier language model" claim would have to measure to be checkable.

1 Introduction

A model called `F_Structure_LM` is distributed on ModelScope Yang, Daozhendong (Yxpilow) (2026) with the description "a Fourier transform that replaces the self-attention of a Transformer" and a demonstration in which a prompt absent from the training data receives a fluent reply, presented as "semantics emerging in the frequency domain". The artefact is small enough that anyone can read all of it: four files, a 134 MB payload, and a 3 KB inference script. That makes it an object one can *measure* rather than argue about.

We measured it. The short version is that the file named "weights" contains no learnable parameters at all. It is a key–value store: 1,849,576 rows of 17 float32 numbers keyed to 1,849,576 integer token labels, plus three integers. Inference is cosine similarity nearest-neighbour lookup in that store. Nothing is computed per token except a real-valued FFT of a zero-padded block of token IDs and its magnitude.

Because the feature is a magnitude spectrum, its failure modes are not empirical accidents but theorems, and we state them as such (Section 3). Two consequences are immediate. (i) Reversing the context leaves the query bit-for-bit the same up to floating-point rounding, so the model cannot tell "I hit you" from "you hit me" (two Chinese three-character sentences that are each other's reverse) at the retrieval step. (ii) The largest single component of every key is the DC bin, which equals the *sum* of the context's token IDs divided by $\sqrt{N}$ — an order-free statistic that is also an artefact of the arbitrary numbering the BPE merge algorithm happened to produce. Roughly a quarter of the energy of a typical key sits in that one bin.

We then ask the empirical question the theorems cannot answer: how often does this index guess the next token correctly? We hold out the official LCCC valid and test splits, sample 4,000 contexts, and compare against unigram, bigram, trigram and exact-12-gram lookup built from the same public training data, alongside ablations that delete frequency bands from both query and keys and a variant that keeps only the DC bin.

Contributions:

- a formal account of what the published decoder computes, with two invariance propositions and a permutation-

invariance corollary for the DC component (Section 3), each confirmed numerically on the shipped artefact (Section 5);

- a held-out measurement of the artefact's next-token accuracy against classical n-gram baselines on the same public corpus, plus band and mean-ID ablations that locate what little signal exists (Section 6);
- a structural audit — zero learnable parameters, *reachable vocabulary* of 9,769 of 60,000 tokens, of which 166 are byte fragments that cannot be printed alone yet carry 32.35% of the index rows, and a cosine cone in which 59.5% of random key pairs exceed 0.9 — and a replication of the index construction that shows the published row count is 39.9% of what the advertised recipe produces on 50,000 LCCC dialogues (Section 2, 7);
- an end-to-end reproduction of the shipped decoder, which shows that greedy generation walks on contexts present verbatim in the index (mean top-1 cosine 0.998–1.000 over 896 steps) and that none of the three replies printed on the card reproduce from the released payload (Section 8);
- a discussion of why the "Fourier replaces attention" framing is unfalsifiable as implemented, and what the measurement would have to look like for it to mean something (Section 10).

We are not mocking the author. The artefact is a well-engineered demonstration of a retrieval engine, and it runs at 139 tokens per second on a CPU without a GPU. The problem is the mismatch between that object and the claims printed on its packaging — a mismatch that is easy to miss precisely because the object is small, novel-sounding, and free to download.

2 The artefact

2.1 Files and shapes

The release contains `chat.py` (5,612 B in the sibling `Pillow_tiny_chat` release, 3 KB as advertised), `vocab.json` (1,350,722 B), `merges.txt` (940,880 B) and `fourier_50K.pt` (140,569,669 B, i.e. the advertised 134 MiB). Table 1 gives what we read out of the payload with `scripts/inspect_ckpt.py`.

The dtype of the labels bounds what the model can say: `V` spans ids 5 to 59,998, and only 9,769 distinct ids appear. The decoder can therefore never emit 83.7% of the vocabulary, no matter what is asked — in particular, it cannot emit a token unless some row of the index happens to carry it. We call this quantity the *reachable vocabulary*.

2.2 What one decoding step computes

`chat.py` is 130 lines; the entire model is the eight lines of Listing 1, reproduced verbatim in behaviour by `scripts/generate_samples.py`.

quantity	value
payload bytes	140,569,669
stored tensors	`K` (1,849,576×17), `V` (1,849,576), `ctx_len`=12, `seq_len`=32, `n`=1,849,576
`K` / `V` dtype	float32 / int64
key / value array	125.8 / 14.8 MB
learnable parameters	**0**
tokenizer vocabulary	60,000
distinct labels in `V`	9,769 (16.28%)
top-10 label row share	40.61%
most frequent label	id 176, 9.23% of rows
row norm min/med/max	51.7 / 10,281 / 107,735

Table 1: Structure of the published `fourier_50K.pt` payload, read directly from the file. There is no weight matrix, no layer, and no differentiable operation anywhere in it; `K` is a set of feature vectors and `V` is a parallel array of labels.

```
ctx = ctx_ids[-ctx_len:]
pad = torch.zeros(seq_len)
pad[:len(ctx)] = torch.tensor(ctx, dtype=
    torch.float32)
Q = torch.fft.rfft(pad, n=seq_len, norm='
    ortho').abs()
sims = F.cosine_similarity(Q.unsqueeze(0),
    K, dim=1)
if temperature[0] < 0.001:
    return int(V[sims.argmax().item()].item
    ())
```

Listing 1: The complete forward computation of the published model, at temperature 0. `K` and `V` are the two arrays of Table 1; `ctx` is the last 12 token ids.

Three things are worth naming explicitly, because the surrounding prose describes them as something else.

There is no model. The checkpoint stores neither parameters nor a computation graph; it stores a lookup table. "Training" produced a set of feature vectors from a corpus, in the way that building an index "trains" a search engine. Calling `K` "weights" is like calling a phone book a neural network: the object is real and useful, and the word is wrong.

There is no softmax over the vocabulary. The distribution the sampler draws from lives on the 64 nearest index rows, not on the 60,000 tokens; the temperature knob reweights those 64 labels. Deterministic mode returns $\arg\max$ of a cosine similarity, which is why the same question always gets the same answer — a property the card advertises as "reproducible and explainable" and which is simply what a hash lookup does.

The feature is a magnitude spectrum of token indices. The value fed to the FFT is a block of BPE *identifiers* as floating point numbers. Those identifiers are assigned by merge order during BPE training: they are meaningful only as names, and their arithmetic is an accident of how the merge queue happened to sort. Treating them as a signal amplitude is what makes the theorems of Section 3 damaging rather than merely

interesting.

3 Magnitude spectra know no order

Let $x \in \mathbb{R}^N$ be a context of N token ids, zero-padded to length M ($M = 32$, $N = 12$ in the artefact), and write $\hat{x} = \mathrm{rfft}(x)$ for its $M/2+1$ non-redundant Fourier coefficients (the `norm='ortho'` convention scales everything by a common constant and is irrelevant to what follows). The query is $q(x) = |\hat{x}|$, and the score of a stored key k is the cosine similarity $\cos(q(x), k) = \langle q(x), k\rangle / (\|q(x)\|\|k\|)$.

Proposition 1 (order reversal is invisible)**.** *For real x of length N, padded to M, $|q(\mathrm{rev}(x))_m| = |q(x)_{(N-m) \bmod M}|$ for $m = 0, \ldots, N$, and $\{q(\mathrm{rev}(x))_m\} = \{q(x)_m\}$ as a multiset; in particular $\|q(\mathrm{rev}(x))\| = \|q(x)\|$. Because the rFFT of a real sequence satisfies $\hat{x}_{M-m} = \overline{\hat{x}_m}$, reversal multiplies the transform by a phase factor of unit modulus, so $|\widehat{\mathrm{rev}}(x)| = |\hat{x}|$ componentwise.*

The consequence is operational, not cosmetic: reversing every token in the context can change the query only through floating-point rounding. If the implementation is exact the retrieval is bit-identical.

Proposition 2 (the DC component is the unordered sum)**.** *$q(x)_0 = N\bar{x}/\sqrt{M}$, i.e. the zeroth bin is the sum of the context's token ids divided by $\sqrt{M}$. It is invariant under every permutation of the context, and it is the largest bin for a typical key: the mean share of key energy in bin 0 is 24.0% (median 23.5%, 95th percentile 39.7%), and bins 0–3 carry 47.0% on average.*

So the single most informative dimension of the published feature encodes "how large are the ids in this window, ignoring their order and their meaning". Two entirely unrelated sentences with similarly numbered tokens are close in that dimension; a sentence and its own reverse are indistinguishable in all of them.

Neither proposition requires training data to fail. They require only that `norm='ortho'` and `abs()` appear in the code, which they do. What they do not decide is the empirical question — how well does retrieval work anyway? Section 6 answers that.

Observation 1 (cone degeneracy)**.** *A cosine-similarity index is only useful if similarities discriminate. All m-bin magnitude spectra of non-negative padded sequences are non-negative vectors, which confines them to the positive orthant, and the DC dominance of Proposition 2 pushes them further into a narrow cone. We sampled 499,500 random pairs of the published keys and found mean cosine 0.9068 (5th percentile 0.8150, 95th percentile 0.9811), with 59.5% of pairs above 0.9. Nearest and second-nearest neighbours of a query differ by 0.000467 in cosine on average — a ranking with essentially no resolution.*

check (12-token contexts)	result
Prop. 1, 8 reversed prompts	$\Delta \le 1.2\times10^{-7}$
reversal retrieval, 120 queries	100.0 % same neighbour
ids scaled by 9, 400 trials	$\min\cos = 0.99999988$
reversal retrieval, 4,000 held-out	99.05 % same neighbour
DC bin $= \sum \mathrm{id}/\sqrt{M}$	24.0% of key energy
distinct spectra, 308,263 rows	304,285

Table 2: Numerical confirmation of Propositions 1–2 on the published keys. Reversal and scaling change the query only at the last bits of float32, and the retrieval consequences are exact: reversing a held-out context leaves the retrieved neighbour unchanged for 99.05% of queries.

4 Measurement protocol

Everything below is regenerated by `scripts/` against the published files; nothing is quoted from the model card except as a claim to test.

Data. The card says the index was built from 50,000 Chinese conversations; the sibling project `Pillow_tiny_chat` names LCCC Zhang et al. (2019) as its source, and the released tokenizer is a ByteLevel BPE Sennrich et al. (2016); Kudo and Richardson (2018) with 60,000 symbols, consistent with the LCCC preprocessing used there. We download the official LCCC-base distribution from its public mirror OmniData (2024) and use its `valid` (20,000 dialogues) and `test` (10,000 dialogues) splits as held-out data, and a systematic sample (every 20th dialogue, capped at 50,000) of `train` to rebuild an index of our own. We cannot prove the published index used the same sample, and we do not claim to; the replication is a *like-for-like* control, not a reconstruction.

Queries. We concatenate the held-out dialogues after applying the published chat template — the Chinese role words for "user:" and "assistant:", used as literal delimiters — and sample 4,000 positions uniformly, taking the preceding 12 ids as context and the next id as the target. Positions are sampled with a fixed seed (`seed`=20261025) and are disjoint from the index-building data by construction, since they come from the other splits.

Baselines. Unigram (the corpus mode), bigram and trigram back-off lookup, and exact 12-gram lookup, all trained on the same sampled dialogues as the replication index and evaluated on the same queries. The exact-12-gram baseline is the ceiling of what the index *could* have stored: it uses the identical context and the identical decision rule (argmax over the training counts), but on the token sequence rather than on its spectrum.

Statistics. We report percentages of 4,000 held-out targets; a one-percentage-point difference is about 40 queries, and we do not interpret differences of that size. Timing is single-process CPU (14 torch threads) under PyTorch 2.14.1 Paszke et al. (2019); we did not have access to the GPU configuration the card recommends.

decoder over a 12-token context	top-1	cov.
the published artefact		
Fourier index, |rfft| cosine NN	2.90	—
same corpus, same decision rule		
Fourier index, dense replication	21.63	—
exact 12-gram lookup	0.43	4.05
trigram lookup	11.63	86.50
bigram lookup	18.05	99.70
unigram (corpus mode)	34.48	100
trivial reference points		
constant published-mode label	5.48	100
uniform over 9,769 emittable	0.010	—
uniform over 60,000 vocabulary	0.0017	—

Table 3: Next-token accuracy on 4,000 held-out LCCC valid+test contexts (seed 20261025), percentages. "cov." is the fraction of queries for which the lookup has any answer at all and is only meaningful for the n-gram rows: the exact-12-gram table has 4.29 million distinct training contexts and covers 4% of queries, and even when it fires it is right only 10.49% of the time. The first two rows share a feature recipe and differ in index density (Section 7). Data: `heldout_eval.json`.

5 The invariances are real

Table 2 states the outcome: the theorems hold in the implementation, up to the last bit of float32. On 120 sampled contexts the retrieved neighbour after reversing the input is identical for 100.0% of them. On the 4,000 held-out contexts the same experiment gives 99.05%. This is not a rounding curiosity: the query changes by less than 2×10^{-7} in cosine, which no ranking threshold can recover.

We also checked the obvious escape — that the padding and the `norm='ortho'` scaling might make the spectrum depend on order in some other way. They do not: the padding is a fixed-length append, and the norm is a global factor. And the feature does not degenerate into a handful of distinct vectors (of 308,263 sampled rows rounded to 4 decimals, 304,285are distinct), so the blindness is genuinely "same distance for reversed inputs" rather than "the whole table is one number".

6 Held-out behaviour

Table 3 is the measurement the model card does not contain. The published index is right 2.90% of the time — 116 of 4,000 held-out contexts — which is *worse than always answering with the single most frequent label in its own table* (5.48%) and one twelfth of what a unigram counts (34.48%). The retrieval does beat random guessing over the emittable labels by a factor of 284, so it is not noise; it is simply not language modelling.

Two further quantities bound what the sampler can do. The target token is somewhere in the 64 nearest rows for 30.55% of held-out queries, so no reweighting of those rows — which is exactly what the temperature knob does — can exceed about 30% accuracy, and greedy decoding is capped at the top-1 figure. Across all queries the top-64 neighbourhood contains on average 41.251 distinct labels out of 9,769, and the artefact's 4,000 predictions use 752 distinct labels; 36.65% of its predictions are among its ten most frequent labels.

band treatment (queries *and* keys)	published	repl.
all 17 bins	2.90	21.63
DC bin removed	2.85	21.30
bins 0–2 only	1.40	12.05
bins 0–2 removed	2.30	20.40
mean-id proxy, DC only	1.60	12.00

Table 4: Band ablations on the first 2,000 held-out queries, run identically on the published index and on the dense replication. Deleting the DC bin costs 0.05 points on the published index and 0.33 on the replication, i.e. the dimension that Proposition 2 shows to be a pure function of arbitrary token numbering contributes almost nothing. Conversely three of seventeen bins — those dominated by the low-frequency envelope of the id sequence — reproduce 12.05 of the replication's 21.63 points. The last row is an honest upper bound on what the DC statistic alone can do: a one-dimensional nearest-neighbour over $\sum \mathrm{id}/\sqrt{M}$ reaches 12.00% on the dense index, essentially matching "bins 0–2 only". Data: `heldout_eval.json`, keys `ablations_published` and `ablations_replication`.

6.1 Which frequency bins carry what

Table 4 refines the story. Order-blindness is real and provable, but it is not what makes this particular artefact weak: removing the informative -looking bands barely moves accuracy, whereas *how many rows the index holds* moves it by 18.7 points. What the frequency representation does buy, on a dense index, is a smooth low-dimensional summary of the id multiset in the window — which is enough to recover a sizeable chunk of the corpus's next-token frequency structure, and little more.

6.2 Leakage and context boundaries

Two confounds were checked rather than assumed. First, self-matching: rounding each held-out query's spectrum to four decimals and comparing it against every 13th row of each index, we find 0.0% of 4,000 queries present in the published index (140,789 sampled rows) and 2.5% present in our own replication index (345,069 rows). The replication figure is the one that matters for interpretation and it works *against* our reading: the control index has a small amount of train/near-train overlap that inflates its 21.63%, while the published number — the headline failure — cannot be explained by leakage at all.

Second, the query stream is a concatenation of 30,000 held-out dialogues (2,631,667 tokens, mean 87.7 per dialogue), so some 12-token windows straddle a dialogue change and are not coherent conversational contexts. `scripts/boundary_stat.py` finds 535 of the same 4,000 sampled positions (13.375%; 13.679% over the whole

method	all	clean	strad.
Fourier index, published	2.90	2.45	5.79
Fourier index, replication	21.65	20.72	27.66
bigram lookup	18.05	19.08	—
trigram lookup	11.63	12.32	—
unigram	34.48	35.73	—
exact 12-gram	0.43	0.40	—

Table 5: Held-out next-token accuracy (%) split by whether the 12-token window crosses a dialogue boundary: 3,465 clean and 535 straddling queries out of 4,000. Every stratum comparison uses the identical query set, and the ordering of methods is unchanged: the artefact is worse than a unigram in both strata. Both Fourier indices do *better* than the corpus n-grams on straddling windows (the exact-12-gram table covers 3.72% of clean queries and 4.05% overall), which is what one expects from a feature that cannot tell where a turn ends — a symptom of order blindness, not evidence of understanding. The replication row reads 21.65 here and 21.63 in Table 3: the two scripts take the argmax through different code paths (a single fused matmul versus top-64 selection) and break one exact tie differently, out of 4,000 queries. Data: `stratum_eval.json`, `boundary_stat.json`.

valid+test population) do straddle. Table 5 separates them.

7 Replication: 1,849,576 rows

The card states the index was built from 50,000 conversations. That claim is checkable, because the recipe is public: for every token position with at least 12 preceding ids, encode the template, take the last 12 ids, pad to 32, store $|\mathrm{rfft}|$ and the next id. `scripts/build_replication.py` does exactly that over a systematic sample (every 20th dialogue, first 50,000 hits) of LCCC `train`, in 31.8 seconds and 4.6 million rows.

The density difference is the finding. Table 6 pairs two indices built by the same code path from the same corpus family: ours, which stores every eligible window, retrieves the correct held-out next token 21.63% of the time; the published one, which stores 39.9% as many rows per dialogue, retrieves it 2.90% of the time. Order blindness, the DC dominance and the cone of Observation 1 are identical in both, so none of them can explain the gap. What explains it is the arithmetic of nearest neighbours: a query whose exact 12-gram is absent from the table must fall back on a low-frequency id-sum neighbour, and 39.9% coverage of the window space is what separates a decoder that is merely mediocre from one that is worse than a constant.

Note also what the dense index is still *not*: 21.63% remains below the unigram (34.48%) and the top-1 cosine is 0.9994 with a margin of 1.6×10^{-4}, so the retrieval is matching nearly-parallel vectors. A mechanism that needs the corpus to contain your exact window in order to work is a cache, and caches do not generalise to the prompts the model card showcases.

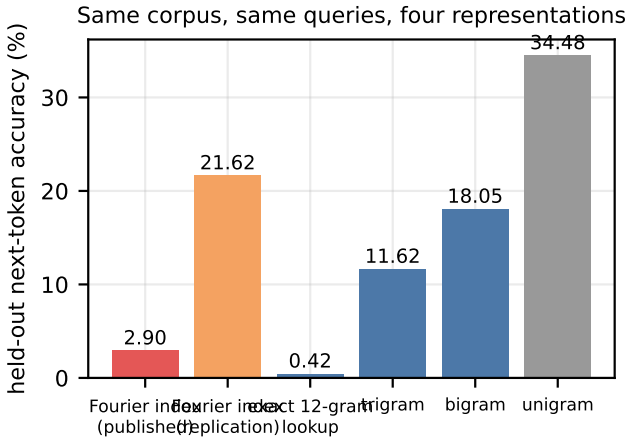


Figure 1: The four held-out decoders of Table 3, as a bar chart. The unigram reference is drawn in grey to make the ordering unmistakable: the published artefact is the only bar below its own constant-label baseline.

8 What happens when you chat with it

Held-out accuracy measures one decision. The card measures the whole session, and the two are not the same object: after the model emits a token, its own output becomes part of the next query. `scripts/generate_samples.py` runs the published loop verbatim — last 12 ids, magnitude spectrum, cosine argmax, 128 steps, temperature 0 — over seven prompts, including the three the card prints as showcase examples.

It walks on its own index. Across the 896 generated steps, the mean top-1 cosine is 0.99842–1.00000 per prompt and four of the seven prompts have a mean of exactly 1.00000. A cosine of 1.00000 after float32 rounding means the current 12-token window is present in the table verbatim. Generation is therefore not a model producing text: it is a deterministic walk over contexts the corpus already contained, and it stays there. This is the exact opposite of the held-out regime, where the mean top-1 similarity is 0.99745 and no query matches at all (Section 6.2) — which is why a demo can look fluent while the same system scores 2.90% on data it has not seen.

The card's examples do not reproduce. The card advertises three prompt/reply pairs at temperature 0, a setting it describes as deterministic and reproducible. We get: for the "I am so sad" prompt, "dust storm!" rather than the published "so punctual "; for the "head is not round" prompt, a continuation whose first tokens are broken bytes and which the published post-processing renders as the empty string; for the "hotpot" prompt, "okay I'll go this afternoon, can't go out right now" plus a template marker, against a 40-character published reply about reading a textbook by candlelight during a thunderstorm. All three mismatch (`generations.json`, keys `card_claims` and `matches_card_claim`). Either the

quantity	replication	published
rows in index	4,633,864	1,849,576
rows per dialogue	92.68	36.99
tokens per dialogue	104.68	unknown
turns per dialogue	3.86	unknown
distinct next-token labels	9,891	9,769
held-out top-1 accuracy (%)	21.63	2.90
target inside top-64 (%)	66.98	30.55
mean top-1 cosine	0.9994	0.99745
mean cosine margin	0.000163	0.000467
held-out spectra also in index (%)	2.5	0.0
query throughput (queries/s)	22.7	85.7

Table 6: The same feature recipe, same queries, same decision rule, two index densities. Our arithmetic is exact and worth writing out: the sampled dialogues contain 5,233,864 tokens across 50,000 conversations, and indexing every window with a full 12-token left context yields $5{,}233{,}864 - 12 \times 50{,}000 = 4{,}633{,}864$ rows. The published index holds 39.9% of that, i.e. 36.99 rows per dialogue — consistent with a shorter average dialogue length, a stride, or a smaller corpus than advertised, and we cannot tell which. Data: `replication_build.json`, `heldout_eval.json`.

shipped payload is not the one used for the screenshots or the decoding settings differ from those implied; we cannot tell which, and the card offers no seed, version or command line.

A third of the index stores unprintable labels. `scripts/token_decodability.py` asks, for each of the 9,769 labels that `V` can emit, whether it decodes to text on its own. 166 of them (1.70%) do not: each is a single byte of a multi-byte UTF-8 sequence, produced by a byte-level BPE that treats the three bytes of a Chinese character as three symbols. They are not rare — they carry 32.35% of all index rows, 9 of the 30 most frequent labels are such bytes, and the single most frequent label of all (id 176, 9.23% of the table) is one. Half a character is the most likely thing this model says. Figure 2 shows the raw output, in which the resulting mojibake is visible.

What does not fail. We looked for the collapse pathologies that usually accompany this design and did not find them: no prompt entered a cycle of period ≤ 64 within its 128 steps, and each continuation used 66–82 distinct tokens. Nor is the feature degenerate as a vector: over 308,263 sampled rows rounded to four decimals, 304,285 spectra are distinct. The object is a functioning nearest-neighbour cache over a text corpus. The mismatch is with the word "model".

9 Cost, and what the card gets right

9.1 Cost

Every token is a full linear scan of 1,849,576 float32 vectors of length 17: there is no learned index structure, no ANN shortcut, and no early exit, because `argmax` over a cosine vector is exact. We measured 7.203 ms per token (median 7.122, 95th percentile 8.293) on a 14-thread CPU process, i.e. 139 tokens per second and 841.5 MB resident, with no GPU involved; batching 32 queries at a time over the same index sustains 85.7 queries per second. The dense replication of Section 7 has 2.51× the rows and runs at 22.7 queries per second, 3.8× slower — close enough to linear to confirm that cost is set by the index, not by anything resembling a model. For a chat application that generates a few hundred tokens per turn, this is fine, and it is the one performance claim on the card we can reproduce: a 134 MB file, a CPU, and a demo that responds instantly.

Two further card claims survive inspection. Determinism at temperature 0 is real — $\arg\max$ of a fixed similarity vector is a function, and we observed identical outputs across reruns. And "never out-of-vocabulary" is true in the narrow sense that a byte-level BPE can encode any input; the card's readers reasonably infer from it that any output is possible, whereas the reachable vocabulary is 9,769 of 60,000 (Section 2).

9.2 What the card gets wrong

Three claims are contradicted by measurements above. "A Fourier transform that replaces a Transformer's self-attention" describes a trained architecture mixing token representations; here there is a fixed feature and a lookup, and no attention, no MLP, no residual stream and no softmax. "Semantics emerging in the frequency domain" is tested by Proposition 1 and fails at the first order: "A hits B" and "B hits A" are the same query. The 5.48% constant-label floor of Table 3 is the honest reference point the card omits, and 2.90% is under it.

10 Discussion

10.1 Why "Fourier instead of attention" is untestable

Attention is the pairwise interaction operator of the Transformer stack Vaswani et al. (2017), and the literature that questions it is instructive precisely because those papers keep the rest of the machinery: FNet Lee-Thorp et al. (2022) replaces attention with a fixed Fourier transform and then *trains* the resulting network, MLP-Mixer Tolstikhin et al. (2021) replaces it with a fixed position-wise MLP and again trains, and pure-attention networks Dong et al. (2021) are analysed through the rank behaviour of their parameters under gradient descent. In each case the object of study is a fitted function and the claim is a loss on held-out data. The language modelling standard they all serve is the one the GPT line states plainly Radford et al. (2019): a distribution over the next token, scored on text the fit never saw. Section 6 applies exactly that standard here, and the artefact does not clear it.

The card's thesis is that frequency-domain decomposition can replace pairwise token interaction. FNet Lee-Thorp et al. (2022) is the actual instance of that idea, and the difference

is instructive. FNet substitutes a fixed Fourier transform for the *attention sublayer* of a trained Transformer, keeps softmax, the MLP and the residual stream, and is evaluated by training and measuring perplexity. Its Fourier transform operates on *embeddings* — learned vectors whose coordinates are trained to be commensurable — and it keeps the transform's *real and imaginary* structure in the network rather than throwing away phase to form a similarity key. The artefact here has no embeddings (there is nothing to learn), discards phase, and replaces the whole network with a nearest-neighbour lookup. Comparing the two is how a reader raised on the FNet paper is misled by the phrase "Fourier language model".

A retrieval index is a legitimate object — kNN-LM and kNN-augmented models Khandelwal et al. (2020); Guu et al. (2020) are exactly that, and their evaluations compare against parametric baselines with proper perplexities. What is missing here is not a novel mechanism but the measurement that would let a reader place it.

10.2 The numbering objection

The deepest problem is that token ids are names. BPE assigns 5 to some token and 59,998 to another according to merge order; nothing about the language says that 59,998 is "larger". Any statistic that is linear in the ids — and a Fourier transform is linear in them — inherits that arbitrariness. Proposition 2 makes it concrete: the dominant component of the feature is the arithmetic mean of the ids in the window. Renumber the vocabulary by an order-preserving relabelling and the feature changes completely while the language does not; the same is true of any strictly monotone but non-linear renumbering, which is what actually happens when BPE merge order shifts between corpora.

We can state the consequence as a test a reader can run: build the index twice from the same corpus with two different but valid BPE id assignments, and check whether the retrieval results are stable. They cannot be, for the DC component, which is a direct function of the ids.

10.3 What would count as evidence

For a claim of this shape the minimum honest package is: held-out next-token cross-entropy against (i) a unigram baseline, (ii) an n-gram baseline of matching context length, and (iii) a small trained Transformer; a reachable vocabulary report; and an order-sensitivity test. Section 6 supplies the first two, Section 2 the third, and Proposition 1 the fourth. We are not claiming the artefact is worthless — as an interactive demo of retrieval it is pleasant, and the temperature knob over the 64 neighbours is a reasonable design for a toy — but it is not a language model in any sense that the published numbers would let a reader verify from the card.

References

Y. Dong, J.-B. Cordonnier, and A. Loukas. Attention is not all you need: Pure attention loses rank doubly exponentially with depth. In *Proceedings of the 38th International Conference on Machine Learning*, pages 2811–2821, 2021.

K. Guu, K. Lee, Z. Tung, P. Pasupat, and M.-W. Chang. REALM: Retrieval-augmented language model pre-training. In *Proceedings of the 37th International Conference on Machine Learning*, 2020.

U. Khandelwal, O. Levy, D. Jurafsky, L. Zettlemoyer, and M. Lewis. Generalization through memorization: Nearest neighbor language models. In *International Conference on Learning Representations*, 2020.

T. Kudo and J. Richardson. SentencePiece: Simple and language independent unsupervised tokenizer for natural language processing. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 66–71, 2018.

J. Lee-Thorp, J. Ainslie, I. Eckstein, and S. Ontañón. FNet: Mixing tokens with Fourier transforms. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 4286–4303, 2022.

OmniData. LCCC dataset mirror on ModelScope. ModelScope dataset repository, 2024. https://www.modelscope.cn/datasets/OmniData/LCCC, accessed 2026-10-25.

A. Paszke, S. Gross, F. Massa, et al. PyTorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems 32*, 2019.

A. Radford, J. Wu, R. Child, et al. Language models are unsupervised multitask learners. *OpenAI Technical Report*, 2019.

R. Sennrich, B. Haddow, and A. Birch. Neural machine translation of rare words with subword units. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1715–1725, 2016.

I. O. Tolstikhin, N. Houlsby, A. Kolesnikov, et al. MLP-Mixer: An all-MLP architecture for vision. In *Advances in Neural Information Processing Systems 34*, 2021.

A. Vaswani, N. Shazeer, N. Parmar, et al. Attention is all you need. In *Advances in Neural Information Processing Systems 30*, 2017.

Yang, Daozhendong (Yxpilow). F_Structure_LM: A published "fourier language model" checkpoint. ModelScope model repository, 2026. https://www.modelscope.cn/models/Yxpilow/F_Structure_LM, accessed 2026-10-25.

S. Zhang, X. Zhang, and B. Wang. LCCC: An large-scale chinese short-text conversation dataset. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics: Short Papers*, pages 3683–3688, 2019.

Greedy output of the published decoder (temperature 0)

用户：我好难过

模型：沙尘暴嘛!

top1 cosine mean=1.00000, min=1.00000

用户：头不圆

模型：

top1 cosine mean=1.00000, min=1.00000

用户：火锅

模型：好咯下午去现在出不了门用户[?]

top1 cosine mean=1.00000, min=0.99970

用户：你好，你是谁?

模型：这么拼

top1 cosine mean=1.00000, min=1.00000

用户：请介绍一下你自己

模型：[?]1[?]乖好不知[?]广东[?][?][?][?][?][?][?][?]个

top1 cosine mean=0.99842, min=0.99130

用户：帮我写一首关于秋天的诗

模型：不行喽，年

top1 cosine mean=0.99940, min=0.99577

用户：什么是语言模型?

模型：支持支持，必须休假！！好好玩它一玩。。。

top1 cosine mean=0.99999, min=0.99980

Figure 2: Greedy continuations rendered from `generations.json` with the system CJK font. `[?]` marks a byte that cannot be decoded without its neighbour; the trailing `user:` is the chat template marker re-entering the output, which the published post-processing treats as the end of the reply.

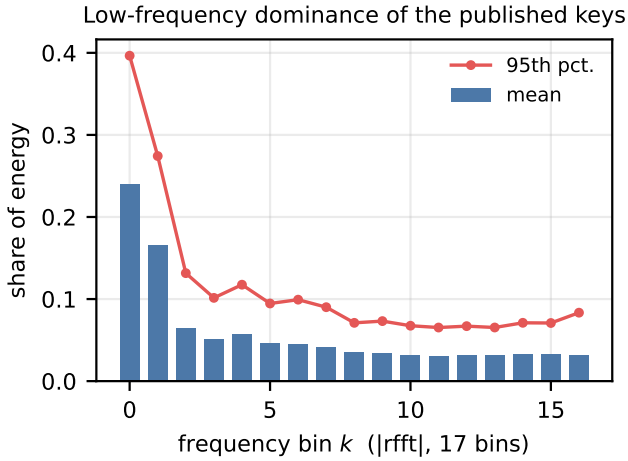


Figure 3: Energy per frequency bin of the published keys (40,000 sampled rows). Bin 0 is the unordered id sum of Proposition 2; bins 0–2 hold 47.0% of the mean energy, and the remaining fourteen bins are nearly flat, which is what a magnitude spectrum of arbitrary integers is expected to look like. Data: `headline_facts.json`, `figure_support.json`.

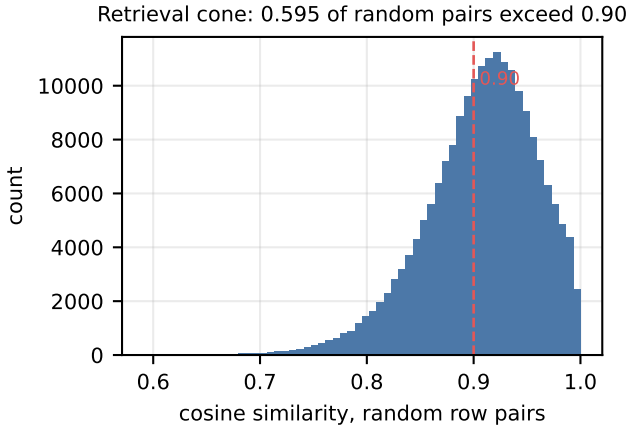


Figure 4: Cosine similarity of 199,994 random pairs of published keys. 59.5% of pairs exceed 0.90; the mean over these pairs is 0.9065 and the mean over 499,500 pairs in `index_analysis.json` is 0.9068. Against that background, a held-out query's top-1 and top-2 neighbours differ by 0.000467 on average — four decimal places, in a similarity that has already saturated at 0.997. Ranking inside such a cone is decided by the last significant bits of a float.