You are wasting 40% of your AI tokens.

TOON fixes that.

Vitthal Mirji · Open Source Summit 2026 Mumbai, India · Linux Foundation

THE LINUX FOUNDATION
OPEN SOURCE SUMMIT INDIA

Vitthal Mirji

Staff Software Engineer - Data platforms

- From Germany, grew up here in Mumbai
- Previously: Walmart & Deloitte
- Data & platform Engineering for entire 12 year career
- Blog/articles vitthalmirji.com
- Maintainer open source SDKs & libraries.
- Few important ones: **flowforge**, **compile-time-contracts**, **toon4s** - TOON for the JVM

```
{
  "id": 1,
  "name": "Alice",
  "dept": "Engineering",
  "salary": 95000,
  "active": true
}
```

One record. Harmless.

```
{
  "employees": [
    {
      "id": 1,
      "name": "Alice",
      "dept": "Engineering",
      "salary": 95000,
      "active": true
    },
    {
      "id": 2,
      "name": "Bob Chen",
      "dept": "Sales",
      "salary": 72000,
      "active": true
    }
  ]
}
```


... × 100 records

id · name · dept · salary · active - repeated 100 times.

Payloads(JSON) was built in 2001 to move data between two machines.

We are using it to talk to a reader who charges by the word.

WHY?

we are using the wrong tool for the job

Tokens are the new bill

127,063 tokens

for 100 employee records, pretty JSON

× 1,000,000

calls a day. Real money, on bytes that carry no information.

It is not only money. It is attention.

Repeated keys are repeated distractions - on every single row.

What if the cheaper format were also the one the model reads *better*?

(hold that thought - we will come back to it)

Why JSON cannot help itself

- JSON objects are self-describing **by design**
- Every object restates its keys - so a parser can read it out of context
- That is exactly right for an API reading **one** object
- It is waste when a model reads **a thousand** in sequence

Shape data for how a model reads -

not for how two servers exchange.

HOW

three moves that make it work

Move 1 - say the schema once

Declare length and fields once. Then stream bare rows. Like a table.

```
{
  "employees": [
    {
      "id": 1,
      "name": "Alice Patel",
      "dept": "Engineering",
      "salary": 95000,
      "active": true
    },
    {
      "id": 2,
      "name": "Bob Chen",
      "dept": "Sales",
      "salary": 72000,
      "active": true
    }
  ]
}
```

Move 1 – say the schema once

Declare length and fields once. Then stream bare rows. Like a table.

becomes

```
employees[3]{id,name,dept,salary,active}:
  1,Alice Patel,Engineering,95000,true
  2,Bob Chen,Sales,72000,true
  3,Carol Singh,Marketing,68000,false
```

Move 2 - drop the scaffolding

Braces and quotes rebuild structure that layout already carries. Use indentation.

```
order[1]:
  - id: 1001
    customer: Ada Lovelace
    items[2]{sku,qty}:
      A1,2
      B2,1
```

This is where TOON leaves CSV behind - it nests, and it keeps types.

Move 3 - make it self-checking

```
employees[100]{id,name,dept,salary,active}:
  ...
```

- `[100]` is not just compression. It is a **contract** - a receipt.
- The model knows how many rows to expect. So do you.
- Drop or hallucinate a row → the count breaks → you catch it.
- JSON ships no receipt.

CSV that grew up.

JSON that went on a diet.

Either way, you land here.

TOON is *not*

- Not a schema language
- Not better for deeply irregular, one-off nested data
- Not a replacement/competitor for CSV when data is purely flat
- Not magic

Admitting the limits is why you can trust the numbers next.

WHAT

the format, the proof, and where it runs

TOON v3.3 – an open spec

Open

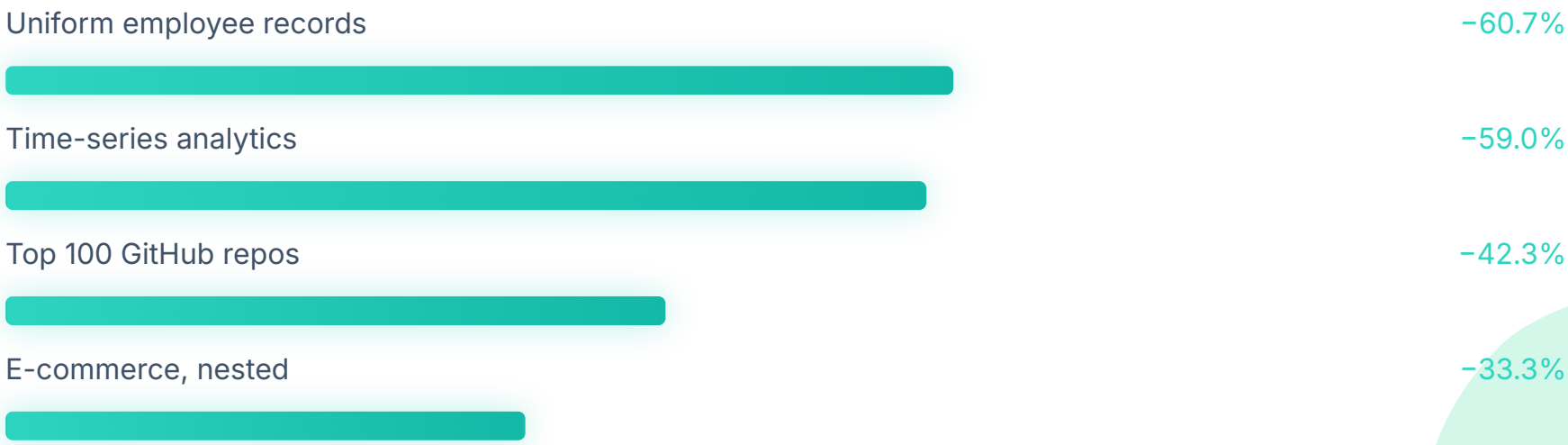
Published spec toon-format/spec

Lossless

Drop-in for the JSON data model

Polyglot

The proof - tokens



Average across all shapes: **~40% fewer tokens**

The proof - accuracy

Remember the question from earlier?

Raw accuracy

TOON **76.4%** · JSON **75.0%**

a near tie

5,016 LLM calls · 209 questions · 6 formats · 4 models

Per 1,000 tokens

TOON 27.7 · JSON 16.4

69% more answers per token

Where it ties or loses - say it plainly

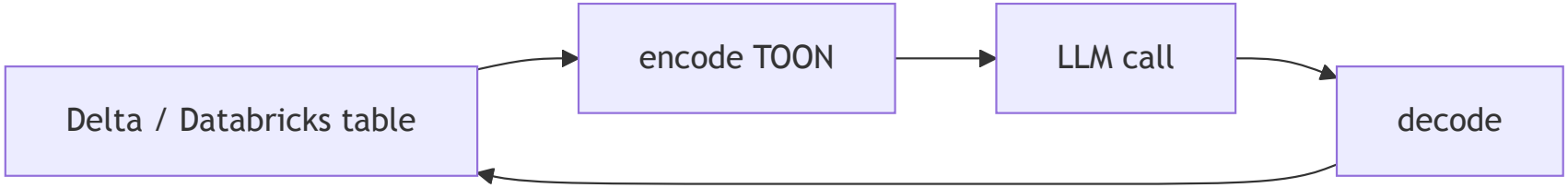
- **Gemini**: TOON and JSON are essentially equal
- **Semi-uniform logs**: TOON can cost *more* than compact JSON
- **Purely flat data**: CSV is still smaller

You spent credibility admitting limits earlier. This is where it pays back.

Where it is paying off

- **Production at scale** - one team reports 50%+ fewer tokens across thousands of daily ChatGPT/Claude calls, with lower latency. *Same prompts, smaller bill.*
- **RAG context** - pack more uniform records into the same window. *More signal per token.*
- **Agent tool I/O** - pass DB rows, metrics, and logs between steps. *Cheaper hand-offs.*
- **Data engineering on Spark** - encode large uniform DataFrames (events, logs, catalogs) before the model. toon4s' home turf.

Where it runs - the JVM loop



toon4s - the Scala implementation. Spec-complete, zero dependencies, Spark 3.5.

Spark and Databricks: encode at the boundary

Your rows are already in Spark. TOON-encode them on the way to the model - no pipeline change.

- **JVM (Scala/Java/Kotlin):** `df.toToon(options)`
  gives a `Dataset[String]` of TOON chunks
- **Spark SQL:** register once, then call `toon_encode_row(struct(*))`
- **Databricks AI Functions:** drop it straight inside `ai_query` / `toon_query`

```
SELECT toon_query('my-endpoint', toon_encode_row(struct(*)))
AS answer FROM employees
```

Streaming on Delta Lake

Delta Change Data Feed → TOON → model, one micro-batch at a time.

```
DeltaLakeCDC.streamDeltaCDCToToon(spark, DeltaCDCConfig(
  tableName          = "transactions",
  checkpointLocation = "/checkpoints/toon-cdc",
  triggerInterval    = "10 seconds"))
```

- `readChangeFeed` → `foreachBatch` → encode each batch as TOON → your LLM → write back
- Real-time fraud, anomaly, and data-quality checks on the change stream

Cost to try: two lines

```
val toon = Toon.encode(json) // send to the model
val back = Toon.decode(output) // parse the reply
```

No migration. No lock-in.

If it does not pay off, you delete two lines.

It is an ecosystem - and an invitation

Already real

- Implementations across languages
- Tools and playgrounds
- I maintain the Scala/JVM one - **toon4s**

toonformat.dev/ecosystem

A place to start

- Spec is small, MIT, finishable in an afternoon
- "Done" is unambiguous - a conformance test suite
- Whole languages still have no implementation

A real open-source thing to put your name on. Come find me after.

Same data. 40% fewer tokens. 69% more answers per token.

And it tells you when a row goes missing.

If you send the same shape a hundred times to a model,

you are paying to repeat yourself.

Stop.

toonformat.dev · github.com/toon-format/spec

github.com/com-vitthalmirji/toon4s · contributors welcome

Thank you

Questions?

Vitthal Mirji · vitthalmirji.com

toonformat.dev · github.com/toon-format/spec

github.com/com-vitthalmirji/toon4s · contributors welcome