An early-access test of TypeSafe's Jev: calibrated judgments for half a cent

Emil Lindfors | September 18, 2026

Here is the last paragraph of a letter a salmon farmer on Senja sent to the Ministry of Finance in January 2023:

> Det er fra flere hold reist tvil om grunnrenteskatt er den mest optimale form når samfunnet skal hente inn mer skatt fra havbruk. Vi deler denne tvilen og hadde gjerne ønsket at vi som samfunn kunne tatt oss bedre tid til å utrede.

They share the doubt about the tax, and they wish there had been more time to study it. The four pages before that are about the norm price and a tax-free allowance of 7,250 tonnes. So is this company against the tax, or for it with changes?

I sent the letter to Jev, a new kind of model from a company called TypeSafe, and asked. Jev is in early access, and I got a key this week. This is the whole answer:

```
stance        against 0.62   for_with_changes 0.37   no_clear_stance 0.01
respondent    company 1.00
tax_design    0.98
```

```
process       0.83
jobs          0.72
substance     2.89 of 3
```

No text and no explanation. Eleven questions, eleven typed answers with probabilities, 4,995 input tokens, 0.31 seconds. The reference label says `for_with_changes`, so Jev got this one wrong. It also told me it was not sure, and that turned out to be the interesting part.

This post is an early-access test: one model version (`jev-1.13.0`), one day, 24 Norwegian documents. The model, the prices and the limits can all change before it opens up, so read the numbers as a snapshot from 18 September 2026.

What Jev is

Jev is TypeSafe's first "System One" model. You give it some state (a text, or JSON) and a set of questions, and each question is one of three primitives:

Primitive	Answers	Returns
Choice	which one of these options	the option, a probability per option
Noul	does this condition hold	one probability of yes
Score	where on this ordered scale	a position, a probability per level

It does not generate text. Your code asks, gets numbers back, and decides what to do with them. All questions over the same state go in one request and run in parallel. During early access the price is $42 per billion input tokens and output is free. That is $0.042 per million, about a seventh of the cheapest chat model I compare it with below.

Why I wanted to look at it

I have always been curious about the alternatives to the way we build language models today. This comes from my evolutionary background, where one should favor variation, as we don't know the outcomes. LLMs may be a local optimum, and it is not certain that the current transformer architecture will succeed.

So when something new and innovative comes along I want to have a look at it, to see how it differs from what we already have, and maybe try to tease out the actual components or "routines" that differ across technologies. That is why I asked for early access to Jev.

Variations exist at several levels. Some change the architecture, like state-space models, or diffusion models that write text. Others change how the model is trained. TypeSafe does not publish what is inside Jev, so I can't place it on the first level. What they do describe is the training that comes after pre-training, and that is where Jev differs.

Chat models get their manners from RLHF, reinforcement learning from human feedback. TypeSafe's primer puts it this way: "RLHF teaches a model to say things that people prefer", and "an output can be compelling to a person without being reliable enough for unattended automation." The same page notes that RLHF "was co-invented by Diogo Almeida, cofounder of TypeSafe". So this is a variant from someone who helped build the design it varies from.

Their alternative is called RLCD, reinforcement learning for calibrated decisions. It "trains TypeSafe to return decisions and calibrated probabilities instead of generated text." RLHF aims at the response a person likes best. RLCD aims at a probability that holds up when you count.

They have not published a paper on it, so I can't say whether it is novel in the literature. But innovation is an idea taken into practice, and this one has an API and a price list. So I take RLCD as their claim and test the part I can test. TypeSafe has not seen this post or the numbers in it.

My first thought was: it's a good and cheap classifier, so that may be a good test. But then we need unruly unstructured data, I suppose.

The docs say Jev "excels in English" and handles other languages "with varying accuracy". My unruly data is Norwegian. So the first question was whether it can read it at all.

The data: 412 opinions about a salmon tax

In September 2022 the Norwegian government proposed a 40 percent resource rent tax (grunnrenteskatt) on sea-based salmon and trout farming. The hearing got 412 published responses. If you want unruly text, a Norwegian høring is great:

- a three-sentence "nei takk" typed into a web form
- long legal commentary from law firms and the bar association, as PDFs
- municipal council minutes with the case number, the vote and the archive code still in them
- bokmål and nynorsk, from fish farmers, mayors, unions, a shareholder association and the Directorate of Fisheries

Every response carries a category the sender picked themselves, and that is unruly too. On the first page of the list, a private person is filed under "Kommune".

I ended up with 25 responses, and here is why. My scraper got a 403 on the first request. The reason was the User-Agent I had written: `jev-horingssvar-eval (research; ...)`. The site's firewall saw `eval (` and took it for an injection attempt. The project name was the attack!

Then I read `robots.txt`. It disallows the `?uid=` pages that hold the individual responses. These are public documents, and `robots.txt` is a convention and not a law. But the repo and this post are public too, and I did not want 800 scripted requests against a stated

preference on my name. So I fetched a sample of 25, spread over the sender categories, at one request a second. One was a scanned PDF with no text layer. That leaves 24.

Twenty-four documents and an early-access model make a first look and not a benchmark. Keep that in mind for every number below.

The questions

One request per document, eleven questions:

- **stance** (Choice): against, for, for with changes, or no clear stance
- **respondent** (Choice): company, municipality or county, interest organisation, private person, political party, public body or research. The sender's name is withheld, because "Lurøy kommune" gives the answer away.
- **eight arguments** (one Noul each): jobs and settlement, investment, the municipal share, the process, tax design, fairness, environment and fish welfare, suppliers
- **substance** (Score): from a slogan to a detailed analysis with figures

The instructions are in English and the documents go in as they are. Here is one of the Nouls:

```
Noul(instructions=
    "... Does the response itself discuss this topic as part of what it "
    "argues, in either direction, beyond a passing mention or a quotation "
    "from the proposal: what host municipalities or counties receive of "
    "the revenue, including Havbruksfondet and the production fee?")
```

One thing to note here. Jev's tokenizer gets about 2.06 characters per token on Norwegian. The limit for the state is 32k tokens, so that is roughly 64,000 characters of Norwegian and not the 120,000 you would guess from English. My longest document was 53,000 characters and fit. Check yours before you plan around the limit.

Who decides what is right?

I don't know if I want to manually label.

So I didn't. I wrote a label guide first, before any label existed. Then Claude (Fable 5.1) labelled all 24 documents twice, in two independent passes that saw only the text. No sender name, no category, no Jev output.

The two passes agreed on 24 of 24 stances, 23 of 23 respondent types and 97 percent of the 192 argument labels. That measures consistency. It does not measure correctness, because both passes are the same model with the same blind spots.

This changes what the numbers mean. Everything below is *agreement with a frontier model's labels*. Where the reference is wrong and Jev is right, Jev scores as wrong.

The results

The comparison is DeepSeek V4.1 Flash through OpenRouter, with the same questions in one prompt and JSON out. I ran it twice, with reasoning on and with reasoning off.

	Jev 1.13	DeepSeek, reasoning off	DeepSeek, reasoning on
Stance, 4 options	20 of 24	20 of 24	22 of 24
Respondent, 6 options	21 of 23	22 of 23	23 of 23
Arguments, 192 yes/no	0.86	0.89	0.88
Substance, exact level	19 of 24	14 of 24	14 of 24
Cost per 1,000 documents	$0.22	$1.31	$3.08
Median latency	0.32 s	2.7 s	26 s
Slowest request	1.3 s	17.9 s	250 s

With 24 documents, the 95 percent interval on a stance number is about 15 points either way. So on stance and arguments the three columns are the same. One respondent type is "undeterminable" in the reference, which is why that row has 23.

So yes, Jev reads Norwegian. It read council minutes extracted from a PDF, archive codes and all, and found the municipality's demand for 70 percent of Havbruksfondet at the end of twelve pages.

The clear differences are these:

1. **Cost and speed.** Jev read all 24 documents for half a cent. DeepSeek with reasoning needed seven cents and wrote 47,000 tokens of thinking to fill in 24 forms. Its slowest request was a letter of 1,800 characters from a transport association. It thought about that for 7,437 tokens and four minutes.
2. **Substance.** The ordered scale is where Jev is clearly ahead, 19 against 14. That is the Score primitive doing the thing it was built for.
3. **Respondent type.** Jev called two private persons a company and an interest organisation. One of those texts is three sentences long. I am not sure there is a right answer in three sentences.

Does the probability mean anything?

This is the more interesting part, because it tests the one thing RLCD is supposed to buy. TypeSafe's primer states the target plainly: "Outcomes assigned a probability of 0.8 should occur about 80% of the time." With 192 argument judgments I can check that, roughly.

Jev said	Judgments	Reference said yes
0.0 to 0.1	14	0%

Jev said	Judgments	Reference said yes
0.1 to 0.3	56	4%
0.3 to 0.7	41	34%
0.7 to 0.9	38	97%
0.9 to 1.0	43	98%

The direction is right everywhere, and it is a bit underconfident at both ends. For the Choice questions I split on the top probability:

	Top probability 0.9 or more	Below 0.9
Stance	14 of 15 agree	6 of 9 agree
Respondent	20 of 20 agree	1 of 3 agree

That table is how you would use Jev in practice. You accept the confident two thirds, and you send the rest to something slower or to a person. TypeSafe's own cookbook on SEC filings reports 27 of 30 correct at 0.9 or above, in English. I got 14 of 15 in Norwegian.

DeepSeek's stated confidence is a different thing. With reasoning on, 84 of its 192 argument answers were above 0.9. When it said something between 0.7 and 0.9, the reference agreed 48 percent of the time.

Careful wording made it worse

I ran Jev twice. The first run used short draft questions: *Does the response raise this argument or topic: the tax threatens jobs, settlement or communities along the coast?* Then I wrote the label guide and rewrote the questions to match it, with the qualifiers about passing mentions and quotations. No labels existed yet, so the rewrite was not fitted to anything.

	Draft wording	Careful wording
Argument agreement	0.89	0.86
Calibration error (ECE)	0.040	0.116
Judgments in the 0.3 to 0.7 bin	24	41

The longer instruction pushed the probabilities toward the middle. I had read TypeSafe's page on Jev's known weaknesses before I started. It says the model reads instructions at face value and that indirection costs accuracy, and the careful version of my question is one long sentence with three qualifiers in it. I did it anyway.

If you are about to write questions for Jev, write them the way you would ask a colleague across the desk, and keep the fine print in your own code.

Reasoning bought two labels

Look at the first two columns of the results table again. With reasoning off, DeepSeek agrees with the reference on stance exactly as often as Jev does, and it gives the same answers as Jev on two of the documents they both miss. Turning reasoning on bought two stance labels and one respondent type, for ten times the latency.

Two labels out of 24 is inside the noise. There is also a reason to distrust the direction. My reference is a model that reads the whole text and reasons its way to a label. So does DeepSeek with reasoning on. Jev does not. Where a label is a judgment call, two reasoning models are likely to make the same one.

Here are the four stance documents where Jev and the reference disagree:

Sender	Reference	Jev	The deciding phrase
Directorate of Fisheries	no clear stance	for with changes, 0.98	"legger til grunn at grunnrenteskatt vil bli innført"
Akademikerne	for	for with changes, 0.65	"i all hovedsak positiv"
NHO Logistikk og Transport	against	for with changes, 0.68	asks for a transitional arrangement
The salmon farmer on Senja	for with changes	against, 0.62	"hadde gjerne ønsket ... bedre tid til å utrede"

A person could make every one of those calls either way. I have not settled them yet. Until I do, "Jev was wrong four times" and "Jev disagreed with an LLM four times" are the same number.

Variation before selection

The theory I keep coming back to is generalized Darwinism. Hodgson & Knudsen (2006) argue that evolution in economic and social systems runs on the same three principles as in biology: variation, selection and retention. Something has to differ, something has to favour some of the variants, and what is favoured has to be carried forward. Essletzbichler & Rigby (2007) brought this into economic geography, which is my field. Without variety, selection has nothing to work on.

Two things follow that are easy to forget. Selection climbs to the nearest peak and not to the highest one, so a population can sit on a local optimum for a long time. And you cannot tell from the inside whether you are on one.

The unit that varies is the *routine*. Nelson & Winter (1982) use the word for the regular, repeatable things an organisation knows how to do. Routines play the part genes play in biology. A firm is a bundle of them, each doing its part of the work. Firms differ because their routines differ, and the routines that work are kept and copied.

A large language model is in a sense a bundle of routines too, each doing a different part of the process. Variation comes from switching out certain parts, and some of those variants are selected and retained.

Here is the bundle, roughly: a tokenizer, an architecture that mixes information along the sequence, pre-training on a lot of text, post-training that shapes the behaviour, and a way of producing the output. The history of the field reads well this way. Attention replaced recurrence in 2017 and was retained. RLHF was a new post-training routine in 2022 and was retained across the field. Reasoning tokens are a recent variant that is being selected right now. State-space models such as Mamba (Gu & Dao, 2023) swap the sequence-mixing part and have not displaced attention so far.

RLCD is a new variation on generative AI. Everything up to post-training can stay as it is. What gets switched out is the generating itself, at the end of the process: the model is trained to return a decision and a probability, where a chat model is trained to write a response people like. Nelson & Winter wrote about firms and not about software, so this is an analogy and I use it as one. It is a useful one, because it tells you what to test. If one routine was swapped, test what that routine is supposed to do.

That is what the experiment above did, for two variants at once:

- **RLCD.** Its job is a probability that comes true. When Jev said about 0.8, the reference agreed at least 80 percent of the time. When DeepSeek with reasoning on wrote down 0.8, the reference agreed half the time. Dropping text generation is also why the price is a sixth to a fourteenth.
- **Reasoning tokens.** Their job is better answers. On this task they bought two labels out of 24, for ten times the latency.

Whether either variant is retained is not up to me or to TypeSafe. Retention is enormous around the chat design. Every prompt library, every API client and every engineer's habits are built for text in and text out. That is great for getting better at the design we have, and it says nothing about whether it is the highest peak.

What the theory recommends here is short:

1. Look at the odd variants while looking is cheap. This one cost half a cent.
2. Ask which part was swapped, and test that part. "Is Jev better than DeepSeek" has no answer. "Which one tells me when it is unsure" does.
3. Keep your own selection environment. My questions, label guide and metrics script did not change while three models sat behind them in one day. If those are yours, you can test the next variant too.
4. Know what you are selecting for. Agreement, calibration, latency and price pull in different directions, and the winner changes with the one you pick.

Experimentation is your friend here. You should try different variants in your own selection environment to figure out what works best, and you have to know what you are selecting for.

This also argues against sending every problem to one model. It is tempting to take the biggest frontier model, Fable 5.1 or whatever leads this quarter, and use it for everything. I expect it would do fine on this task. The price per document would be many times Jev's, and a chat model does not tell you which of its answers to doubt.

Look at where each model fits in this small project:

Job	Model	Why
48 careful reference labels	Fable 5.1	few documents, judgment calls, cost does not matter
All documents, every question	Jev	cheap, fast, and says when it is unsure
A second opinion on the unsure third (not built yet)	DeepSeek, or a person	slower and dearer, so only where it is needed

Three jobs, three different tradeoffs. I only know which goes where because I ran the variants side by side.

What to try

If you have a pile of text in a small language and a classifier-shaped problem:

1. Write the questions down before you pick the model. Short ones.
2. Decide what the right answer is for the ugly cases first. The three-sentence anonymous letter will show up.
3. Measure characters per token on your own language before you plan around a context limit.
4. Run the cheap model on everything and look at what it is unsure about. That list was the most useful thing I got out of the day.

What happens next for this one: I settle the four stance documents by hand, pin DeepSeek to one provider (OpenRouter spread my runs over 13, so the latency numbers above are a mix), and ask the ministry whether I can have all 412. And since Jev is still in early access, there will be a newer version to test. The questions are fixed and a full run costs half a cent, so I will rerun it and see what moved. The questions, the label guide, the predictions and the scripts are in the repo.

References

- Nelson, R. R., & Winter, S. G. *An Evolutionary Theory of Economic Change*. Harvard University Press, 1982.
- Gu, A., & Dao, T. "Mamba: Linear-Time Sequence Modeling with Selective State Spaces". *arXiv preprint arXiv*:2312.00752, 2023. [link]

- Hodgson, G. M., & Knudsen, T. “Why we need a generalized Darwinism, and why generalized Darwinism is not enough”. *Journal of Economic Behavior & Organization*, vol. 61, no. 1, pp. 1-19, 2006. doi:10.1016/j.jebo.2005.01.004
- Essletzbichler, J., & Rigby, D. L. “Exploring evolutionary economic geographies”. *Journal of Economic Geography*, vol. 7, no. 5, pp. 549-571, 2007. doi:10.1093/jeg/lbm022