



Designing for Probabilistic Outputs

AI UX design 01 | Output variability in the interface, confidence display options, choosing defaults (suggest vs auto-apply) with an expected-value rule, graceful degradation when the model is slow, unsure or down

#UX #Variability #Confidence #Defaults #Fallback

🕒 Outputs that vary and are sometimes wrong

A probabilistic output

In one sentence
●○○○

Practitioner
●●○○

Expert
●●●○

Under the hood
●●●●

Ask the same thing twice and the AI may answer differently; most answers are fine, a few are wrong, and they all look equally confident.

The interface must make variation useful (alternatives, regenerate) and errors cheap (preview, edit, undo), and must say how sure it is only when that signal is reliable.

- vary where creativity helps
- freeze where consistency matters

Each design choice is a bet on accuracy p: auto-apply pays off only above a break-even accuracy set by the cost of an undetected error and the cost of a review.

- defaults by expected value
- fallback when p is unknown

Variability also comes from outside sampling: silent model upgrades, retrieval changes and batch-dependent numerics, so a feature must version outputs and degrade gracefully.

- log model and prompt version per output
- regression test on upgrades

Analogy

A weather forecast, not a clock: useful every day, wrong some days, and trusted because it says '70 % chance of rain' rather than pretending to be certain.

Probabilistic outputs at four depths

```
graph TD; subgraph Must_know [Must know]; direction TB; M1[Temperature, top-p sampling  
Learn: llm (decoding)]; M2[Calibration, why scores mislead  
Learn: responsible-ai 06]; end; subgraph Helpful [Helpful]; direction TB; H1[Precision, recall, thresholds  
Learn: machine-learning 07]; H2[Latency: TTFT, p95  
Learn: metrics-and-kpis 07]; end; subgraph Advanced [Advanced]; direction TB; A1[Model routing and fallbacks (serving side)  
Learn: mlops 04]; end; Must_know --> Design[Design for probabilistic outputs]; Helpful --> Design; Advanced --> Design;
```

🔄 Variability: use it or freeze it

Context: "Reply to the customer about the late delivery:" → next token?

		T = 0 (greedy) H = 0 bits kept 1 · H = 0 bits	T = 0.7 H = 1.65 bits kept 4 · H = 1.51 bits	T = 1.2 H = 2.04 bits kept 5 · H = 2.04 bits
token	logit z			
▶ Thanks	2.4	100% → 100%	56% → 57%	41% → 41%
Sorry	1.9	0% cut	27% → 28%	27% → 27%
Hello	1.2	0% cut	10% → 10%	15% → 15%
Unfortunately	0.6	0% cut	4.3% → 4.4%	9.2% → 9.2%
We	0.3	0% cut	2.8% cut	7.2% → 7.2%

▶ greedy pick: the highest logit, at every temperature · cut: top-p = 0.95 on the tempered probabilities; light bar = before the cut, dark bar = renormalised

Softmax with temperature: $p = \exp(z/T) \div \sum \exp(z/T)$ over all tokens
Entropy: $H = -\sum p \cdot \log_2 p$ (bits; 0 = certain, $\log_2 N$ = uniform over N tokens)
Renormalised: $p_{\text{kept}} = p \div (\text{sum of } p \text{ over the kept tokens})$
where z = logit, the model's raw score of a token; T = temperature (< 1 sharpens, > 1 flattens, 0 = always the top token); p = probability of a token;
top-p = keep the fewest most likely tokens whose probabilities add up to at least p (nucleus)

Why two runs differ: the first word of a drafted email reply. At temperature 0 the reply always starts 'Thanks'; at 1.2 'Sorry' or 'Unfortunately' start about 1 reply in 3, changing the tone. Pick the setting per field, not per product

Pattern	What the UI does	Use when
Several alternatives	show 2-4 options side by side; user picks	creative output: headlines, images, replies
Regenerate	button for a new output; keep earlier ones in a pager (1/3)	user may prefer another phrasing
Edit in place	output lands in an editable field, not a read-only bubble	output is a draft the user owns
Pin and version	accepted output is stored; replay shows the stored one	reports, records, anything audited
Low temperature + schema	structured output, temperature 0, validated fields	extraction, classification, form filling
Consistency check	warn when a regenerated answer contradicts the first	factual Q&A; disagreement signals uncertainty

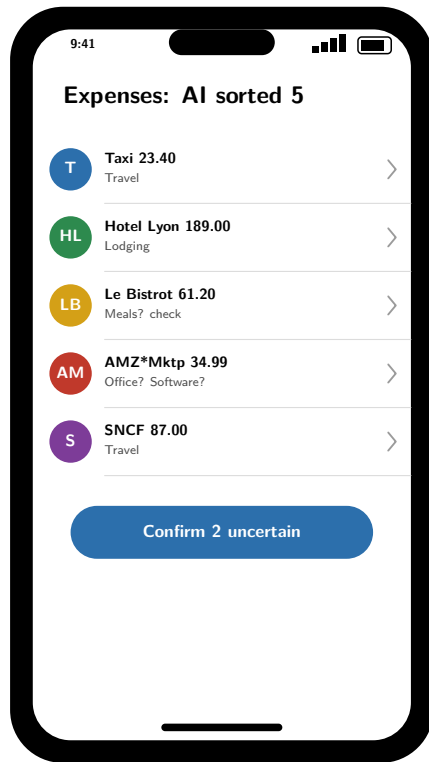
Variability patterns: where variation is a feature, expose it; where it is a bug, freeze it

⚠️ Temperature 0 is not a determinism guarantee

Many hosted APIs still return slightly different outputs at temperature 0 (batching and floating-point order on GPUs, provider-side model updates). If a value must never change, store the accepted output and the model version; do not regenerate it on every page load.

🕒 Tadata | tadata.fr | contact@tadata.codes

Showing confidence



Expense app: the AI categorises receipts. Only uncertain rows ask for a tap; confident rows stay quiet. Bands come from calibrated scores (sheet 02), never from the raw model output

Display	Looks like	Good for	Risk
None	output only	very high accuracy, cheap errors	users cannot tell bad from good
Flag on low only	'check' badge on uncertain items	triage lists, forms	unflagged errors are trusted blindly
Bands	high / medium / low	most consumer and business UIs	band edges must be calibrated
Percentage	'83 % match'	expert users who act on odds	false precision; users anchor on it
Ranked alternatives	top 3 with the first preselected	ambiguous inputs	choice overload past 3-5
Span highlight	uncertain words underlined	transcripts, OCR, translations	visual noise if overused

Confidence display options, from least to most information. More information is not better: show the least that changes the user's next action

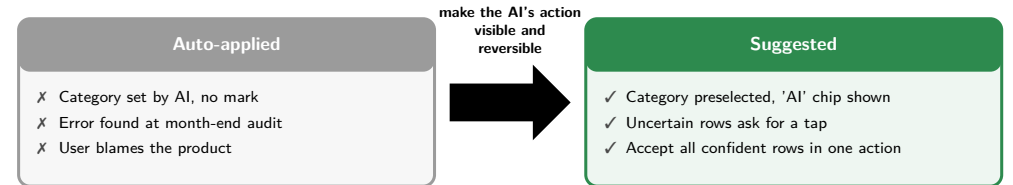
Words and numbers

People read verbal probability terms very differently (in surveys 'likely' is read as anything from about 55 % to 85 %). If you use words, define each band once in a tooltip; if you use numbers, round them (80 %, not 83.4 %) and only for expert users.

Defaults: suggest or auto-apply?



From suggestion to automation. Moving right saves effort per item and raises the cost of each undetected error



Same model, different default: the after version turns silent errors into one-tap decisions

PROBLEM

An AI fills the cost centre of invoices. It is right with probability $p = 0.95$. A correct fill saves $g = 30$ s. An undetected wrong fill costs $c = 600$ s to find and fix later. In suggest mode the user checks each item in $r = 5$ s and fixes wrong ones on the spot (no later cost). Which default saves more time, and above which accuracy does auto-apply win?

GIVEN

- $EV_{\text{auto}} = p \times g - (1 - p) \times c$ (seconds saved per item)
- $EV_{\text{suggest}} = p \times g - r$ (wrong items are fixed during the check: no saving, no later cost)
- $p = \text{accuracy (0-1)}$; $g = \text{seconds saved by a correct output}$; $c = \text{seconds lost per undetected error}$; $r = \text{seconds of review per item}$

1 Auto-apply

$$0.95 \times 30 - 0.05 \times 600 = 28.5 - 30 = -1.5 \text{ s}$$

Why: errors cost 20 times what a success saves

2 Suggest

$$0.95 \times 30 - 5 = 23.5 \text{ s}$$

Why: a short check catches every error

3 Break-even accuracy

$$(1 - p) \times c = r \rightarrow p = 1 - r / c = 1 - 5/600 = 0.9917$$

Why: set the two EVs equal

ANSWER

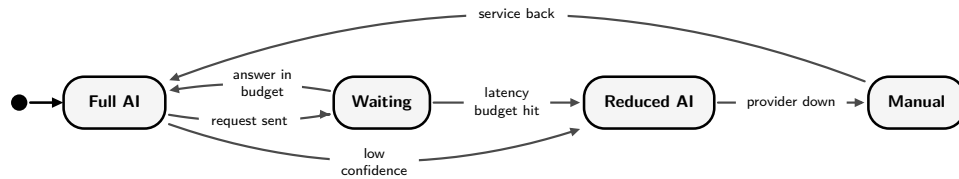
Suggest saves 23.5 s per item; auto-apply loses time. Auto-apply only above 99.2 % accuracy, or if the undo cost c can be cut (for example by flagging uncertain items)

CHECK

✓ With $c = 60$ s (error caught the same day) the break-even falls to $1 - 5/60 = 91.7$ %: making errors cheap to find is as powerful as making the model better

When does auto-apply beat suggest? Compare expected time per item

Graceful degradation



Interface states of an AI feature. Each failure moves the user to a reduced but working state, never to a blank screen or a spinner without end

Trigger	Fallback	What the UI says
Slow (over latency budget)	stream partial result, or smaller faster model	'Still working, here is a first draft'
Low confidence	show alternatives or abstain	'Not sure: pick one or fill it in'
Provider outage, rate limit	cached answer, rules, or manual form	'AI suggestions are unavailable; you can continue manually'
Policy block	explain and offer an allowed path	refusal copy (sheet 06)
Context too long	summarise or ask to narrow	'This file is too long; choose a section'

Degradation ladder: what the user sees at each trigger

Degradation checklist

✓ baseline / mandatory □ recommended – verify in your context

- ✓ The manual path still exists and is one click away *never remove the non-AI way to do the task*
- ✓ Every AI call has a timeout and a defined fallback
- ✓ Partial results are kept, not discarded on error
- **MUST** Fallback states are labelled so users know quality changed
- **SHOULD** Outage copy tested like any other screen

Myths, recap and self-test

MYTH

✗ Showing a confidence percentage always builds trust

✗ Variability is a bug to remove

✗ Auto-apply is always faster

✗ If the model fails, show an error

FACT

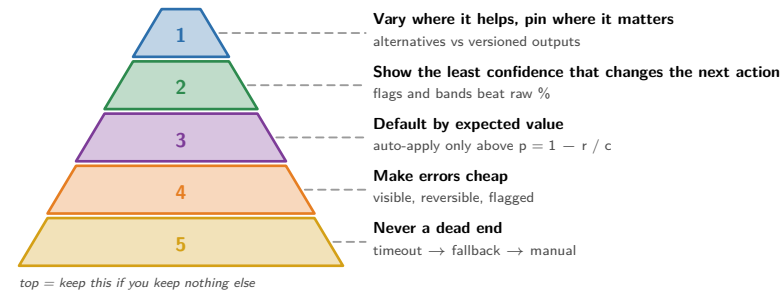
✓ Uncalibrated numbers mislead and users anchor on them
Show bands derived from calibrated scores, or flag only the uncertain cases.

✓ For creative tasks it is the feature
Expose it with alternatives and regenerate; freeze it only where consistency or audit matters.

✓ It is faster only above the break-even accuracy
 $p = 1 - r / c$: with costly hidden errors that can be above 99 %.

✓ Fall back to a reduced but working path
Cached results, a smaller model, rules or the manual form keep the task moving.

Remember



Q1 Name four ways to make variability useful.

A Several alternatives, regenerate with history, edit in place, consistency check across runs.

Q2 Why is temperature 0 not enough for a value that must never change?

A Hosted inference can still vary (batching, numerics, model updates); store the accepted output and model version.

Q3 Break-even accuracy for auto-apply vs suggest?

A $p = 1 - r / c$, with r review seconds per item and c cost of an undetected error.

Q4 $r = 5$ s, $c = 600$ s: break-even accuracy?

A $1 - 5/600 \approx 99.2$ %.

Q5 Flag-on-low-only display: main risk?

A Unflagged errors get trusted blindly.

Q6 Why round displayed confidence?

A False precision invites anchoring; 80 % conveys the same decision value as 83.4 %.

Q7 Four triggers for degradation?

A Latency budget hit, low confidence, provider outage or rate limit, policy block (plus context too long).

Q8 Ghost text vs auto-apply?

A Ghost text is opt-in (accept with Tab, ignore by typing); auto-apply takes effect unless undone.

Q9 What must every fallback state tell the user?

A That quality changed (reduced or manual) and how to continue.