SAMPLE

Building Tomorrow's Prompt Engineers

The Eval's Keeper in the Machine Era

Aiman JoeHanz

Free sample: Introduction & Chapter 1

putrabooks.github.io

Buy the full book:

Carousell carousell.sg/p/building-tomorrow-s-prompt-engineers-ebook-pdf-1471386262

Ko-fi

Payhip

Contents

Introduction

Chapter 1 — The Eval

Chapter 2 — The Territories

Chapter 3 — The Way In

Chapter 4 — The Working Day

Chapter 5 — The Eval

Chapter 6 — The Drift

Chapter 7 — The Conversation

Chapter 8 — The Instruments

Chapter 9 — The Year Whole

Chapter 10 — The Long Arc

Introduction

An instruction to a machine has no quality until somebody has decided how to tell whether it worked, and that decision is the work.

What follows concerns how the people who compose those instructions actually work day to day: the habits the craft relies on, the way the roles split, what the career pays out for, and the shifts arriving as models grow more capable while judging what they produce grows harder. It is written for people considering the role, for engineers and domain specialists moving into it, and for the managers who commission work whose output nobody can easily check.

The claim is narrow. Prompt engineering is not the discovery of a phrase that unlocks behaviour, and it is not the pursuit of a score on a public test. It is the maintenance of an eval: a set of cases, drawn from what the system is actually for, that can be run repeatedly to answer the only question that matters, which is whether this version is better than the last. Everything else, the instruction, the examples, the structure, the retrieval, the guardrails, the post-processing, either improves that answer or is decoration nobody can defend.

Two folklores obscure this. The first is the incantation, which treats the instruction as a spell whose wording has power. It is attractive because it is dramatic and because occasionally a rewording does transform behaviour. It fails because a phrasing that works on one model version is not knowledge, and because a practitioner who cannot say why it worked cannot say when it will stop.

The second is the benchmark. Public evaluations are real, they are useful for comparing systems, and they are not your problem. A team that optimises for a public set produces a system that scores well on a distribution nobody in its user base occupies, and the difference between the benchmark and the task is where the craft lives.

Ten stages then follow: what the trade is, where it gets practised, the routes by which people arrive, a working day, the craft, its strain, the conversations carrying it, the tooling, one composite year, and lastly the long view of a career.

One thing belongs here rather than at the end. This is the youngest occupation described in this collection, and its practitioners are frequently the first people in an organisation to discover that a model has changed. The chapter on strain

is not about burnout in the ordinary sense. It is about a specific condition of this work: that it can degrade while nobody touches it, and that the person who notices is often the person with the least standing to say so.

Nothing here assumes a particular product, vendor or model. The mechanisms described hold for any system whose behaviour is produced by instruction rather than by code, and where they differ, the text says so.

Chapter 1 — The Eval

What the Eval Is

An eval is a set of cases with expected outcomes, run against a system, whose purpose is to answer whether a change made it better.

The set is drawn from the work the system is for, not from the work that is easy to generate. It contains the ordinary case, the awkward case, the case that a real user produced at an inconvenient moment, and the case that broke the last version. Each has an expectation attached, and the expectation is written by somebody who understands the task rather than by somebody who understands the model.

The purpose is comparison. An eval is not a certificate and it is not a grade; it is an instrument for telling whether one version of an instruction is better than another. A team without one can change the system and cannot know what the change did, which means every subsequent decision is made on the impression of whoever last looked at an output.

This is why the eval, rather than the instruction, is the trade's keeper. The instruction is the artefact that gets shipped. The eval is the thing that makes shipping a decision rather than a hope.

The Two Folklores

Every young occupation acquires stories that make it legible from outside, and this one has two.

The incantation says that the instruction is a phrase with power, and that a practitioner's skill is in finding the phrase. Its evidence is real: a rewording does sometimes transform behaviour, and the transformation is memorable. Its failure is that the practice it encourages cannot accumulate. A person who cannot say why a phrasing worked cannot say when it will stop working, and the vocabulary of magic phrases is a vocabulary of anecdotes rather than of knowledge.

The benchmark says that the field has a scoreboard, and that progress is movement on it. Its evidence is also real: public evaluations exist, they are comparable across systems, and they are useful for choosing between them. Its failure is that the scoreboard measures a distribution and the product serves a task. A system tuned to score well on a general test can be worse at the specific job, and the gap is invisible unless somebody built a set for the specific job.

Both folklores replace the eval with something tidier: one with a lucky phrase, the other with somebody else's test. Neither survives a model update, which is the ordinary condition of this trade.

What Is Actually Held

Ask practitioners what they are responsible for and the answers vary by employer; underneath the variation is one thing.

It is the correspondence between what the system is asked to do and what has been demonstrated about its behaviour, and the knowledge of where that correspondence stops. An engineer holds it for a pipeline, a product manager for a workflow, a domain specialist for the professional judgement the system is supposed to reproduce. None of them holds the model. Each holds a claim about behaviour, and every claim has an edge.

So the chief artefact of this profession is the record, not the prompt. It holds the eval cases, the version history, the result of each run, the failure that was examined and understood. A prompt says what was tried. A record says what was established, and the second is what a practitioner needs when the system misbehaves in front of a customer.

Practitioners who have understood this treat the eval as the instrument rather than the paperwork. They write cases because they will need them, not because a process asked for them.

The Three Commitments

The argument rests on three commitments, which set out good practice as currently done rather than arguing for change.

The first is to build the eval before fixing the instruction. An instruction written without a set of cases is a draft whose quality is a matter of opinion, and the opinion belongs to the most senior person in the room rather than to the task.

The second is to examine the failure rather than the average. A score of eighty per cent is a fact about a distribution and says nothing about which twenty per cent failed or why. The craft is in reading the failures, classifying them, and deciding which ones matter for this system with these users.

The third is to record what the system was, as well as what it did. Models change, frequently without notice, and a result is meaningless without the version that produced it. A practitioner who cannot say which model version produced a result has a number rather than a finding.

The Written Eval and the Real One

Two evals exist for every system, and the distance between them decides what the team can know.

The written eval is the set of cases the team maintains: the file, the criteria, the score it produces, the dashboard a manager looks at. It is precise where somebody thought to write a case and silent everywhere else, and its silences are exactly the behaviours nobody has tried to protect.

The real eval is the set of cases the users produce. It contains the awkward input, the contradictory instruction, the request in three languages, the paste of an entire document with no question attached. It contains everything the users do, at their prevalence, in their proportions, and almost none of it is in the written set.

The practitioner's job is to keep the distance small, which means deliberately harvesting from the real set. A team that reviews a sample of production inputs every month and adds cases from them keeps its instrument honest. One that builds its set once at the start measures the system's behaviour on the day it was built, which is a historical document wearing a scoreboard's clothes.

The Four Dimensions of a Case

A case has four dimensions, and a set that omits one of them measures something narrower than the team believes.

Input is the obvious one: what the system is given. Context is the second: what else is available, the retrieved documents, the conversation history, the user's account state. Expectation is the third: what a good output is and how it will be judged. Provenance is the fourth: where the case came from, which is what tells a reviewer whether the set samples the task or the team's imagination.

Provenance is the dimension most frequently lost. A case that arrived from a real conversation carries the user's phrasing, which is where the difficulty lives. A case written by the team carries the team's phrasing, which is a paraphrase of the task rather than the task. Six months later nobody can tell the two apart, and the set quietly becomes a test of the team's own vocabulary.

The remedy is a field in the case record: source, date, and what it represents. The field costs a minute and it is what allows a reviewer to say that half the

set is real and half is invented, which is a fact about the instrument's authority.

The Score Is a Claim

A score is not a measurement. It is a claim with a scope, and the scope is always smaller than the number suggests.

A score of eighty-one per cent means that on this set, with these criteria, applied this way, eighty-one per cent of cases were judged acceptable. It says nothing about the other nineteen per cent in terms of consequence, nothing about cases outside the set, and nothing about whether the criteria captured what the users need.

The practitioner's discipline is to state the scope whenever the number is quoted upward. A manager who hears eighty-one per cent and understands it as a general quality measure will make decisions the number cannot support, and the practitioner who allowed that understanding is responsible for the consequence.

The second discipline is the comparison. A score is useful only against another score, produced by the same set, the same criteria and the same model version. A number compared with a number from three months ago, across a model change and a set revision, is a comparison of two different instruments, and the difference between them is not a change in the system.

What Verification Actually Establishes

Verification in this trade establishes behaviour under the conditions that were sampled, and practitioners who forget the qualifier over-claim on their own work.

A run establishes that the system, at that version, on those cases, produced those outputs, judged by those criteria. It does not establish that the system is good, that the users will be satisfied, that the failure rate is acceptable in production, or that the behaviour will hold next month.

The craft is in carrying the limits forward. A result is recorded with its set version, its model version, its criteria version, and the date. The record is what allows a later practitioner to know what was established rather than what was believed, and the distinction is the whole value of the document.

The second craft element is the honest scope statement. What this run addressed and what it did not: the classes it cannot detect, the inputs it does

not sample, the failure it cannot see because the case does not exist. A verification note that includes those lines is usable by somebody who was not present, and that is the only test of a record's quality.

The Case That Was Written Down Too Late

Every team owns a set of cases that were added after an incident, and their lateness is the point.

The case exists because something reached a customer with nobody watching for it. It is written afterwards, from a transcript, by somebody who already knows what the answer should have been. It joins the set and it will prevent that exact failure in future, which is genuinely useful and is also the most expensive possible way to acquire it.

The pattern is worth naming because it is structural rather than careless. Nobody writes a case for a failure they have not imagined, and imagination follows experience. A team's set therefore always lags its incidents, and the lag is closed only by effort that has no deadline attached.

The remedy is the scheduled hunt, which is an hour a month spent reading production traffic for things that could have gone worse. The hunts produce cases for near misses: the input that was answered correctly by luck, the question that the system handled but only because the user rephrased. Those cases cost an hour and prevent the incident that would otherwise supply them.

The Instruction and Its Four Readers

An instruction is a document written for a machine and read by people, and the second audience decides whether it survives.

Its human audience is four people: whoever edits it next, whoever signs it off, whoever audits it, and whoever inherits it after everybody else has gone. None of them can read it the way the model does, and each needs something different: the colleague needs structure, the reviewer needs the change isolated, the auditor needs a statement of what it does and must not do, the successor needs the reasons.

The craft is therefore composition rather than phrasing. Task first, then constraints, then examples, then the output contract, each separable, each with a line explaining why it exists. A reader who can find the section that governs their interest in ten seconds is a reader who will maintain the document instead of replacing it.

The second element is the comment. A constraint with a reason survives review; one without is removed by the first person who finds it inconvenient and cannot see what it protects. The reasons are the only thing in the instruction that a machine does not need and every human reader does, which is precisely why they are the first thing omitted under time pressure.

The Idiom of the Trade

This occupation has developed a vocabulary quickly, and its words carry assumptions that are worth unpacking.

An instruction is called a prompt, which suggests a single utterance rather than a document with sections and a history. The output is called a completion, which suggests finality in something that is frequently a draft. A failure is called a hallucination, which attributes to the system a mental state and discourages the more useful question of what evidence it had.

The words matter because they shape the work. A team that thinks in prompts writes one paragraph and iterates on wording; a team that thinks in instructions keeps a document with a reason for each clause. A team that thinks in hallucinations looks for a model defect; one that thinks in unsupported claims looks for a retrieval gap.

None of this argues for abolishing the vocabulary. It argues for noticing it, because a metaphor that has become a technical term stops being examined, and this trade's youngest words are the ones most in need of it.

The Refusal to Measure

Occasionally a team decides not to build an eval, and the reasons given are worth examining because each contains a real constraint.

The first reason is speed. There is no time before launch, and the instrument is deferred until after the deadline. The constraint is real and the consequence is that the deadline is met with a system whose quality is unknown, which means the deadline was met for the launch and not for the product.

The second reason is the task's nature. The work is creative, or the output is judged by taste, or the users are forgiving. Each of these can be true and none of them removes the question; they change the instrument. A creative task is evaluated with exemplars and comparative judgement, which is slower and still possible.

The third reason is the model's pace. By the time the set is built the model will

have changed, so the effort seems wasted. This is the most seductive and the most wrong. The cases survive model changes; the criteria survive them; the taxonomy survives them. What dates is the technique, and the instrument is the part of the work with the longest life.

The fourth reason is the absence of authority. Nobody asked for it, and building one is not in anybody's objectives. This is the honest reason and it is the most common. It is also the reason the trade's practitioners should treat the instrument as part of the job rather than as a deliverable, because a deliverable that nobody requested will be cut first.

The Ninety Minutes That Establish the Task

There is a session that produces most of a project's definition in ninety minutes, and it has a shape worth describing.

It begins with inputs. Twelve real examples, read aloud, with the person who owns the task saying what they would have done with each. That half hour produces the criteria more reliably than any requirements document, because the owner is reacting to specifics rather than describing an abstraction.

It continues with the boundary. Two examples that are out of scope, deliberately chosen at the edge, and the question of what the system should do with them. The answer is a routing rule or a refusal, and either is better than the ambiguity that would otherwise be discovered in production.

It ends with the consequence. What happens today when this is done badly, and what the cost looks like. That produces the priority among the criteria, which is what the eval needs in order to resolve the conflicts that will arrive in the second week.

Ninety minutes, twelve inputs, two exclusions, one consequence. Everything after that is execution, and the projects that skip the session spend months discovering its content in arguments.

The Small Team's Instrument

A small team cannot afford a large evaluation, and the instrument it can afford is different in shape rather than merely smaller.

The shape is a small set kept current. Forty cases, of which ten are the common task, ten are the awkward variants, ten are the dangerous class, and ten are the ones that broke in the past. Refreshed quarterly from real traffic, reviewed by one qualified person, run on every change because the run is

cheap.

That set costs a day a quarter and it catches most of what a large set would catch, because its composition is deliberate. What it cannot do is measure a rate precisely, and the small team should therefore never quote a percentage from it as though it were one. It is an instrument for direction, not for precision, and knowing which is which is the discipline.

The second property of the small team's instrument is that it must be somebody's. In a team of three, an eval that is everybody's is nobody's, and the assignment is what keeps it current. One name against the instrument is the difference between a set that is maintained and a document that was created once.

The Cost of a Case and What It Buys

A case costs between ten minutes and an hour, depending on whether it needs a reference output, and the cost is worth examining against what it prevents.

Ten minutes buys an input and a criterion, which catches the class of failure that is obvious once described. An hour buys a reference output and a qualification, which catches the class of failure that only a professional would notice. The second is where the dangerous failures live and it is forty times rarer in a typical set, which is the bias the book argues against throughout.

The practical arrangement is a budget rather than a wish: a stated number of expensive cases per quarter, chosen by consequence rather than by convenience. Five a quarter is twenty a year, and twenty well-chosen reference outputs will find more dangerous behaviour than two hundred criteria.

The second consideration is who pays. A reference output written by an engineer without the domain produces a standard calibrated to plausibility. The hour belongs to somebody qualified, which means the case's cost is partly a scheduling cost, and scheduling is what makes teams substitute criteria for references and then wonder why the set never catches the dangerous class.

ABOUT THE AUTHOR

Aiman JoeHanz writes the Building Tomorrow's People series, one volume per working role. This one belongs to the people whose craft is making a model behave on purpose: the prompt engineer who writes the instruction and then proves it, the eval that decides whether the instruction survived a model change, and the harness that turns a demo into something a business can depend on. The book treats prompting as instrument work rather than incantation, which is why it spends its pages on evaluation, versioning and regression instead of clever phrasing. He writes from the conviction that the roles which survive this decade will be the ones that can be audited from the outside.