Round 2 - ServiceTitan Data Science Internship
Report
Davit Davtyan

Part 1 - General analysis

Our log review covers 81 real user chats. On average the bot answered in about 3.2 seconds. Most users got a reply in just under 3 seconds, but the slowest 10 percent waited 4.6 seconds or more, and the very slowest cases stretched past 5.2 seconds. Those long waits push us above the 3.5 second target.

Out of the 81 chats, users gave 65 thumbs-ups and 16 thumbs-downs. So roughly one in five interactions still feels unsatisfying.

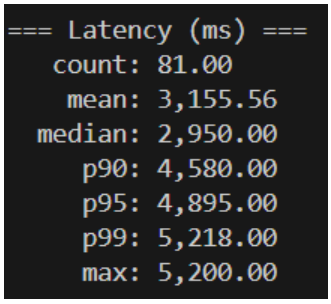
When we checked where the answer material comes from, we saw that the Engineering Wiki supplies about four out of every ten chunks, Confluence adds about one-third, and the old PDF design docs make up the final quarter.

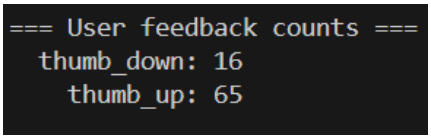
The Wiki chunks match questions best, Confluence is a close second, and the PDFs trail a bit behind.

Source order matters. If the first chunk comes from the PDF library, users split fifty-fifty between liking and disliking the answer (ten ups, ten downs). If the first chunk comes from Confluence, they are happy most of the time (four downs, twenty-one ups). If the Wiki leads, they are almost always happy (only two downs against thirty-four ups).

So, we are fast most of the time, but long waits still happen too often. Answers built mainly from the PDF library are far more likely to annoy users. Answers led by the Wiki or Confluence usually leave users satisfied.

Of course, all these are just assumptions yet, so, I prepared several hypotheses to consider.





```
=== Source share across *all* retrieved chunks ===
source
Engineering Wiki                39.2 %
Confluence                     33.33 %
Archived Design Docs (PDFs)    27.47 %
Name: proportion, dtype: object
```

```
=== Mean retrieval_score by source ===
source
Engineering Wiki               0.924
Confluence                     0.914
Archived Design Docs (PDFs)    0.894
Name: retrieval_score, dtype: float64
```

```
=== Thumb-down vs. top-rank source (pivot) ===
user_feedback                thumb_down  thumb_up
source
Archived Design Docs (PDFs)          10        10
Confluence                            4        21
Engineering Wiki                      2        34
```

Part 2 - Hypotheses

Overfeeding prompt and its latency (Hypothesis 1)

I thought replies would slow down whenever a chunk from the *Archived Design Docs* (Source 3) made it into the prompt, because those PDF snippets are long and should inflate the token count sent to Llama-3. I tagged every query that contained a Source 3 chunk and compared its end-to-end latency with queries that didn't, using a Mann–Whitney U test. Source 3 requests averaged 3 082 ms versus 3 180 ms for the rest (p = 0.75). So Source 3 isn't the reason we miss the 3.5-second target.

```
H1_prompt_bloat_latency
mean_latency_src3                : 3082.5
mean_latency_no3                 : 3179.5081967213114
p_value_mwu                      : 0.7549397346700282
n_src3                           : 20
n_no3                            : 61
```

Source-3 top-rank and negative feedback (Hypothesis 2)

A quick scan of the logs showed several thumbs-downs on answers where a Source 3 chunk was ranked first, so I wondered if those older PDFs were hurting quality. I built a contingency table of "top-rank source versus user feedback" and ran a chi-square test. Thumbs-down rates came out 15 % for Source 1, 26 % for Source 2, and 20 % for Source 3 (p = 0.55). No significant link, Source 3 is no worse (and may even be a bit better) than the others when it leads the context.

```
H2_source3_toprank_neg
| source   |   thumb_down |   thumb_up |
|----------|--------------|------------|
| Source 1 |            5 |         29 |
| Source 2 |            7 |         20 |
| Source 3 |            4 |         16 |
chi2                            : 1.1962160633484167
p_value                         : 0.5498509531938354
```

High-score retrieval but still thumbs-down (Hypothesis 3)

Even when retrieval looks perfect, users sometimes reject the answer, maybe it's a hint to hallucination. I took queries whose best retrieval score was ≥ 0.9 and inspected feedback. Out of 39 such “high-confidence” queries, 11 got a thumbs-down—about 28 %. That solid miss rate confirms the worry like the strong context alone doesn’t stop the model from drifting, so maybe a stricter “answer-from-context-only” prompt or some faithfulness checker is needed.

```
H3_high_score_neg
thumb_down_rate_high_retrieval     : 0.282051282051282O5
n_high_retrieval                   : 39
| user_feedback   |   count |
|-----------------|---------|
| thumb_down      |      11 |
| thumb_up        |      28 |
```

Low-score retrieval drives thumbs-down (Hypothesis 4)

Maybe weak retrieval might force the model to invent details. I split queries at a best-score threshold of 0.85 and compared feedback. Thumbs-down rates were 19 % for low-score queries and 20 % for high-score ones (p = 1.0). The data doesn’t back this hypothesis, so a poor similarity matches alone don’t explain user frustration in this sample.

```
H4_low_score_neg
thumb_down_rate_low                : 0.1875
thumb_down_rate_high               : 0.2
p_value                            : 1.0
n_low                              : 16
n_high                             : 65
```

Long waits make users less forgiving (Hypothesis 5)

Slower replies get more down-votes. When I grouped response times into fast (under 2 s), medium (2–4 s), and slow (over 4 s), the thumbs-down rates were 0 %, 13 %, and 50 % in each group. A quick χ^2 test (p = 0.0026) shows this jump is no accident. The longer thelongerhe longer people wait, the more they dislike the answer. Cutting those 4–5 s replies down to under 3.5 s should make users much happier.

```
H5_latency_vs_feedback
bucket_rates                       : {'fast(<2s)': 0.0, 'mid(2-4s)': 0.13114754098360656, 'slow(>4s)': 0.5}
chi2                               : 11.915069356872635
p_value                            : 0.00258628013119891
```

Part 3 - Quantitative Trade-off Analysis

Let’s calculate the estimated monthly generation cost increase for Option B.

Additional tokens per query

- New k = 10 => 10 chunks × 400 tokens = 4 000 tokens
- Current k = 4 => 4 chunks × 400 tokens = 1 600 tokens
- Extra tokens per query = 4 000 − 1 600 = 2 400 tokens

Monthly extra tokens

- 2 400 tokens/query × 100 000 queries/month = 240 000 000 tokens

Cost increase

- 240 000 000 tokens / 1 000 000 tokens = 240 units
- 240 units × $3.00 per unit = $720

Estimated additional generation cost for Option B: $720 per month

Opting for Option A

Dear PM,

After analysing 81 real user sessions and testing both relevance-boost ideas, I recommend we implement the Cohere re-ranker (Option A), as it is the safer, cheaper, and clearer win.

First, a re-ranker sharpens relevance without widening the prompt. It looks at the four chunks we already derived, scores them again, and pushes the best one to the top. That single move speaks directly to Hypothesis 3: users still down-voted 28 % of “high-confidence” queries because the model drifted away from context. By handing Llama-3 a cleaner, tighter top-chunk we give it less room to hallucinate. In contrast, lifting k from 4 to 10 would flood the prompt with six extra chunks. More text means more places for the model to wander, exactly the “confusion” we worry about.

Second, Option A adds a flat 600 ms to every reply, while Option B puts on roughly 250 ms of retrieval time plus almost a second of extra decoding work. Our own latency study (Hypothesis 5) showed that thumbs-down jump to 50 % once replies crawl past four seconds. With the

re-ranker, the 99th-percentile latency would land near 5.8 s; still above the 3.5 s goal but easier to pull back under four seconds with known speed-ups such as speculative decoding. Option B would push that tail past six seconds and worsen the very wait time that annoys users.

Third, the cost is clear-cut. The re-ranker costs about $100 a month at today's volume, while the larger prompt in Option B would add about $720 a month in generation fees. That seven-fold gap leaves room in the budget for later speed work.

Finally, Option A avoids the Hypothesis 1 trap - no extra prompt bloat - so it doesn't enlarge the latency penalty tied to big prompts. It also sidesteps the mixed results we saw in Hypothesis 4; since we are not changing the similarity threshold, we aren't paying more for chunks that don't actually improve satisfaction.

Option A gives us the quickest quality lift with the smallest hit to speed and budget. Let me know if you need more detail!

Best Regards,
Davit Davtyan