Finding Problems Worth Solving

or

“Let the Data Speak”

Most technical founders start with a solution. I founded a company years ago and learned the hard way as all startup schools will tell that is not the way to go. I am not go looking for a problem and I'm doing it the other way around.

I built a small internal tool — AISignals — to systematically surface real problems founders, startups, SaaS companies are struggling with. Not trending topics, not what VCs are excited about. Actual friction that people describe in their own words, in communities like Reddit or Hackernews.

My Approach

The tool runs keyword-driven searches across online communities, looking for specific language patterns: frustration signals, time-waste indicators, workarounds people describe, things they wish existed. Phrases like "I spent too long on," "I wish I had known," or "I ended up writing a script to", etc. These are markers that something is broken enough for people to talk about it publicly.

Each result gets scored across six dimensions: Is this a real problem? Is it addressable? Can I actually build something for it? Is the audience reachable? Will they pay? And so one. This is about finding a scoring filtering removing the noise. Anything below a 50% signal threshold gets dropped. What remains goes into a Kanban pipeline for manual review.

I went through three iterations in a week. Version 0.1 was a basic search-and-save workflow. Works, hardly useful and it still required me to do all the heavy lifting. Version 0.2 added LLM-based cross-comparison of results, but lacked enough signal depth. Version 0.3 is where it got really interesting and I had a great aha moment I haven't had in a long time.

The Clustering Discovery

I ran a local LLM against the full database using cosine similarity clustering — finding entries that aren't identical, but semantically close. Different people, different communities, describing related pain points from different angles.

The epsilon parameter controls how tight the clustering is. Think of it as a sensitivity dial. At 0 you'd need exact semantic matches. At 1, everything is unrelated. At 2, you're comparing opposites.

I started at 0.25, which was too loose. The clusters were noise, grouping things that had surface similarity but no real connection. Dropped to 0.1, guess what too tight. Almost nothing clustered, because the threshold demanded near-identical meaning. I was sitting there and thinking why was is it not working, but this is where Opus gave me some clarification and explained the underlying mechanics. At 0.2, something clicked. Clear, coherent problem clusters emerged from hundreds of scattered signals. The reason the value is low is, because most of the content we picked up from the search is already pretty similar.

The validation moment: these clusters independently surfaced the same pattern I had been noticing when I had to sort manually over the previous days to train the LLM. Technical founders consistently struggling with business execution — marketing, positioning, reaching the right customers. Now this is a common, well known problem, but that the algorithm saw it by comparing everything against everything allows me now to look for possibly yet undetected signals.

What's Next

The tool works, but still has a lot of room to improve. Now it's about two things: fine-tuning the algorithms and running my own judgment against the clustering output. Early days mean, the human mind is still the best filter, but soon AISignals should do the job. This isn't about building the perfect research tool. It's about finding one problem worth solving and having evidence, not just instinct, to back the decision. What I learned when I worked at Lyft as manager and it was a great mantra "let the data speak".