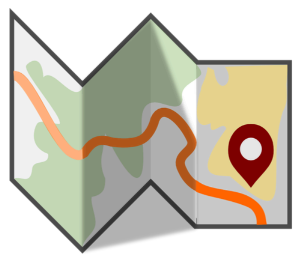
Strategy guide for the understaffed search team

Maintained By Doug Turnbull - http://softwaredoug.com,
Contact Me - http://linkedin.com/in/softwaredoug

Frankly the economy sucks right now. Search teams are not excluded from this mess. Still, your leadership has said something like "Improve search. But we can't give you more resources to do so. Good luck!" 🙃.

You've spawned in some weird, alien landscape. Maybe you happen to be in a desert biome, or some forest. (Sorry, not sorry, I have kids, so Minecraft metaphors abound.)

This guide is for all those teams tasked with the impossible. It's a beginners guide, to help you take those early steps, to gain confidence so you can keep exploring deeper and deeper into *your* specific, weird search landscape.



Where and how you walk through the landscape depends on where you ~~spawn~~ start. Some will be easier, some harder, for your specific situation. As you go through, see if you can identify areas that might be good next steps for your team. Don't feel like you ought to read this linearly - just scan the table of contents, and take whatever next steps help you.

Second, I don't get into nitty-gritty implementation details - like "how to boost by matching early in text". A good google search can help you with *your* specific tech stack. I want to make this as "scannable" as possible.

Finally, it's OK to feel overwhelmed! If you did everything in this guide you'd be busy for ***years***. We all take time to get there… we're all on a journey, and never "done"!

Getting oriented - what’s your stakeholder’s definition of success?

Agree on a measurable definition of success with your stakeholders. If you’re an underfunded search team, well, that means to survive and grow you have to focus on the value-add to the overall org. Try to agree on a concrete measurable that a non-search stakeholder would understand.

Does your team contribute to:

- Sales / conversions?
- Increased customer satisfaction (perhaps scores like Net Promoter Score, etc)
- Increased usage, like DAU (Daily Active Users) or time spent in your app?
- More sensitive statistics that predict these stats? Like add-to-carts instead of total dollar amount in sales. Or leading statistics like satisfaction you might study correlate to longer-term, lagging stats - customer retention and product renewals
- The farther you go down the funnel away from search for the conversion, the less sensitive it will be to a search intervention.

Most of these are fairly recognizable software industry statistics. That’s intentional - the “search nerd” stuff is for your team internally, not for the outside.

Ideally you would be able to A/B test relevance (and other!) changes against these stats. That might not be entirely possible. Maybe you don't have enough traffic. Perhaps your measure of success is not sensitive enough to an A/B test. Or you don't have any kind of A/B testing setup.

It's also important to note there are product considerations beyond just optimizing for one, easily measurable metric. Brand safety is one important measure - you don't want your brand associated with inappropriate content that, maybe converts well, but might put your company on the front page of the New York Times for something horrible.

For more information - I recommend OpenSource Connection's search quality training.

Your map and compass - knowing if your changes are going in the right direction

You'll usually want to evaluate a change before going to an A/B test to measure the business metric you discovered above. Or, if you can't A/B test, this could be even more central to how you evaluate your search.

For evaluation of a relevance change before going to an A/B test, look at two types of stats:

Optimize ranking stats (NDCG, Judgments, etc)

The first question we ask when working on relevance: did we move relevant results up higher / push irrelevant results lower? Usually this can help us improve ***precision*** within the set of results labeled relevant or not. We call these kinds of labels of what's relevant for a query a *"judgment list"*. The 'labels' might come from which results convert for that query, or they might be hand labeled (or both!). Metrics such as NDCG evaluate a given query's ranking on a scale of 0-1 to see how close that query comes to the ideal ranking as specified for the judgment list.

Learn more about judgment lists and offline evaluation:

- What is a judgment list?
- What is test-driven search relevancy?

Optimize for 'bad query exploration'

You might not have easy access to good judgments. Further, judgments are heavily limited by one of search's most pernicious problems: presentation bias - users only click or convert what they see. Even if judgments are created from hand labels, you only have so many that can be labeled.

Still, it's common to know what queries perform well, and which need improvement. Perhaps through your analytics, you have a sense of the class of queries you need to improve, but don't want to upset the applecart on the good ones.

Another way of looking at search relevance evaluation is to simply try to measure

1. Did we minimize ranking changes to the *good* queries?
2. Did we make the change we expected (as in mechanically, like a unit test)?
3. Did we change the bad queries we expect to change?

"Change" here doesn't mean good or bad. It simply means *change*. It means our subsequent A/B test will measure what we expect it to measure.

Learn more about this methodology in the article NDCG is overrated

Taking your first steps - good early search experiments

Things to try that often work to improve a lot of search interfaces. These have consistently (though not always) been consistent performers for A/B tests across multiple projects.

Boost generally popular results

It's not bad to put generally popular stuff first that matches the query. Especially stuff that ***converts*** well ***in the search UI***. Though other sense of global popularity like overall sales rank, etc can help. The definition of "generally converts well in search" can itself be a complicated problem to solve. But you can start simple with whatever sense of global popularity you have and iterate.

Explicitly downboost poor performers

You may hear of the idea of "signal boosting" in search. This simply means to observe which results get a high number of clicks and conversions, and place them higher in the search results.

However, signal boosting amplifies search's most pernicious problems - presentation bias. It keeps us in a local maximum, and doesn't give us a chance to try other results for that query.

I prefer a different approach. - Instead of up-boosting what works well for a query, I prefer signal ***downboosting***: if you know per-query, what has gotten a lot of impressions in the search UI, but underperforms its peers with high confidence - move these results towards the bottom of the ranking.

By definition, this improves the precision of the results, while helping improve result diversity. It also helps overcome presentation bias by giving other results a chance in the top N.

Use ‘relevance feedback’ techniques

Rocchio and similar approaches learn from what users click / convert for a query. By observing that when a user searches for “shoes” they end up clicking on items in the category “footwear” we can learn that “shoes” should perhaps boost items in that category. This is just one example, any property of a document can be associated with a query simply by observing the relationship between the query and relevant documents.

This kind of query -> document(s) feedback can be a core feature of successful relevance approaches.

- See https://en.m.wikipedia.org/wiki/Rocchio_algorithm

Invest in embedding based retrieval (ie vector search)

Get familiar with embedding based retrieval - text, image, etc. there are a lot of preexisting, open source models for doing vector search (query text->image). While your current infra may not yet support this, the value is worth the upgrade to latest versions of Solr /ES or a vector search DB to use these techniques. This is one area a bit of upfront infrastructure investment likely could payoff if done well.

This approach can be combined with rocchio-type approaches to build up even more accurate query embeddings, especially for short queries. You can learn that “shoes” tends to correspond to product images that correspond to a certain area in your embedding space.

- See Vector search for the uninitiated
- Sease (http://sease.io) has a lot of training on vector based retrieval

Weave together results with interleaving and rank fusion -

Not an experiment, per-se, but interleaving *different* ranking strategies can add diversity to the search results. It can also be an effective A/B testing strategy. Reciprocal Rank fusion combines different rankers by ranking by the sum of 1/rank of each underlying ranker. This type of combination is important for combining vector search with traditional retrieval strategies.

Carefully apply manual overrides: rules and synonyms, etc

It’s better to override a weird one-off behavior through a rule, boost, synonym, etc than it is to fix it with global ranking. Consistent statistics have shown in my experience that actually hand-managed recommendations and search results can out-perform algorithmic results! It’s also important to patch over embarrassing or potentially legally questionable search results to protect your company. Unfortunately rules can be insanely time-consuming, and can go out of date. So tread carefully.

See Querqy (https://querqy.org/) for one such system of managing query rewriting rules.

Dealing with lots of text? Try these simple text matching hacks

For traditional keyword based retrieval, a few types of text matching techniques work pretty well:

- **Optimize for BM25's expectations** - BM25 - the core scoring algorithm in your search engine - works best with *article* length text. Not book length or tiny snippets. Break your large chunks of text up accordingly to maximize its effectiveness. Eliminate noisy text, lists of tags, or other types of text that don't organically correspond to an article.
- **Proximity matters** - You can usually score terms higher that occur closer together, in my experience this has shown to matter. You can turn this on with a phrase query with a slop.
- **Earlier in the text matters** - Typically the most important topics occur earlier in an article, so the matches here should be counted more valuable

Query understanding hacks - collocations and compounds -

You can get started understanding queries without building some giant, complicated knowledge graphs

- **Use collocations** - The first step is just understanding terms that statistically go together "Palo Alto" should be considered a single unit, not two distinct concepts. Same with perhaps "Albert Einstein" or "Mazda Miata" - perhaps these are distinct concepts. Really all you're doing is recognizing important phrases, then doing a phrase search with them. See The Unreasonable Effectiveness of Collocations
- **Compounds -** you can find collocations from your text that searchers frequently append can point at commonly compounded and de compounded words (ie back pack and backpack)

Mine query refinements for corrections

How do users correct their queries in session? This might point at common confusions or misspellings. If they happen a lot, you could short circuit and offer those corrections yourself.

Relax the query to increase recall

Issued a search and got 0 results? Try again with a less strict query. Perhaps turning the query from AND to OR, turning on fuzzy search, or interleaving embedding based retrieval, or showing users recommendations, or strategically dropping a term in subsequent search passes.

Don't sacrifice performance: Go faster! Jump higher!

It's important to note, speed is a crucial factor in how users perceive relevance. Don't sacrifice performance to give perfectly accurate search results.

See an introduction to search quality.

Communicate relevance in the search results page

Sometimes users will report search as not giving relevant results simply because the UI does not effectively show users *why* a result is relevant. Search results highlighting is one such classic solution, but not the only one. Simply showing the important attributes of an item (images, relevant snippets, etc) can help demonstrate to users why an item is relevant.

See an introduction to search quality.

What's in your backpack? - infra considerations

You want to make iterating on your production search stack really easy. So do everything you can to build resilience - a giant "UNDO" button - so that you can resolve any bad configuration that ends up in search instantly.

Make (re)indexing *seamless*

you should be able to discard and rebuild your indices easily to incorporate new features. A lot of teams using streaming systems like flink or beam to keep things up to date but also allow batch processing of historical stuff

Have multiple versions of your index in prod

Be more resilient by being able to go back to your last reindex if need be. Most systems have collection aliases explicitly for this purpose.

Make search indices disposable

don't freak out about a weird index. Just delete it and reindex / point to an existing index. Search indices should be as disposable as possible.

Reindexing: optimizing for changes to search index config

In this case, the actual data being added to the index is unchanged. It's nice in these cases to have a staging database of your documents, such as a document database like MongoDB, flat files on S3, etc, so you can just reindex into the search engine easily.

Reindexing: optimizing for adding data from upstream data warehouse

in this case, you want to make it really easy to connect to whatever upstream system exists and pull the data into your search engine (or staging database).

Tools and community resources

See the "Awesome Search" repo for - Grumpy Search Dev's awesome search list has lots of great tools, resources, books, training, conferences and more (https://github.com/frutik/awesome-search)

Strategy - ship now in your competence areas, but work to get deeper leverage

The most important thing

Most teams have different levels of competence in the above areas. It can feel overwhelming. The #1 thing you can focus on is how fast can you deliver an improvement to relevance? The quickest way to get a project canceled is to promise, promise, promise but never deliver.

Note there's actually ***two*** statistics here: *speed* and *improvement*. In other words, you're not just throwing crap out to the search, you're iterating confidently, continuously learning and improving your processes.

Blazing new paths to production

I like to think in terms of paths, roads, etc when it comes to going to production. Some roads are well established. You have a team of snowplows, maintenance workers, traffic engineers, etc available to manage that "road" connecting a set of systems in prod-land. The highway system of your company can respond to issues with that road. But proposing a new road means ramping up new infrastructure, placing load on existing communities, and convincing all the adjacent land-owners, stakeholders, businesses, etc it's a good idea. Good luck!

As a concrete example, what I mean is you'll have organizational competency in perhaps evolving a search service, with engineers and others that don't worry / fear about dealing with that service. If it breaks, many people know how to fix it. But when you propose a new thing - like a brand spanking, new way of indexing to production from the data warehouse - it takes time to build that muscle internally. It means new, unknown technologies. It means unique complications. It means creating new load on existing systems. So it takes time to align all these systems - and most importantly aligning people to accept and even be enthusiastic about the change.

To increase your sophistication, you ***need*** to blaze these new paths to gain deeper leverage on the relevance problem. Otherwise you may be stuck in a local maximum. BUT you can't depend on a risky, complicated, potentially political wrestling match to get your new system established.

This means you need to consider two things:

1. **What can I improve *now*** given my maturity and resources? Maybe reindexing takes weeks, and there's a distant, promised, new indexing project that might fail, due to be delivered months down the line - so what can you do on the query side easily. Work there!
2. **Blazing new paths to production -** new patterns of working in production (like a new indexing system, a new way of working with the search cluster, a search API, etc) require building resilience and trust with your colleagues in infra, SRE, and the services being interacted with. So "blazing new paths" takes time.

How to effectively blaze new paths

Blazing new paths in an organization is IMO a very specific skill that requires a mix of human / political skill and strong technical competence. You need someone who can get buy-in from stakeholders, while simultaneously knows the technology to actually deliver.

Some tips on how to do this well:

- **Make partner teams part of the solution** - share success liberally, don't hoard it for yourself. Especially infrastructure teams don't get enough credit in use-facing solutions, make sure they get lots of credit. Unless they're sociopaths, they'll pay such success forward
- **Show, don't tell** - A truism, don't endlessly describe designs and systems. Show people what it can do with a prototype.
- **Find a way to ship ASAP** - How can you get into prod NOW? Greenfield systems that stay in the lab too long often fail. Stakeholders lose interest. You want to find a way to get into prod in some gradually, incremental fashion ASAP. Perhaps by taking a small percentage of traffic, or working in a side-area, or shipping a skeleton behind a feature flag. Anything that wears that path to prod fast
- **Discuss options and natural consequences with stakeholders -** While a good vision is really powerful and important, it's also a great idea to take a step back and communicate to stakeholders in terms of their options. This creates a participatory environment and builds trust
    - For a reindexing system:

    - If you stick with the existing system, we won't be able to add new features to the search index, I expect we would only be able to make a tiny, low single digit improvement to relevance
    - If you do this new thing, it will require upfront investment $X, but we would probably be able to improve search in double digit percentages…. And second we can validate this system by shipping on this side search surface in 4 weeks, to mitigate the risk and confirm its value.

Final thoughts - you can do it!

You're embarking on an exciting journey. A healthy team can do this. You can make improvements, even with limited resources. I'm happy to chat with most people for 30 minutes if you want an ear or bit of advice, don't hesitate to get in touch - http://linkedin.com/in/softwaredoug