SAMPLE CHAPTER

With applications for Solr and Elasticsearch

Relevant SEARCH

Doug Turnbull
John Berryman

FOREWORD BY Trey Grainger

MANNING

Relevant Search
With applications for Solr and Elasticsearch

by Doug Turnbull
John Berryman

Chapter 1

brief contents

1 The search relevance problem

This chapter covers

- The ubiquity of search (search is all around us!)
- The challenge of building a relevant search experience
- Examples of this challenge for prominent search domains
- The inability of out-of-the-box solutions to solve the problem
- This book's approach for building relevant search

Getting a search engine to behave can be maddening. Whether you're just getting started with Solr or Elasticsearch, or you have years of experience, you've likely struggled with low-quality search results. Out-of-the-box settings haven't met your needs, and you've fought to deliver even marginally relevant search results.

When it comes to relevance ranking, a search engine can seem like a mystical black box. It's tempting to ignore relevance problems—turning the focus away from search and toward other, less mystical parts of the application such as performance or the UI. Unfortunately, the work of search relevance ranking can't be avoided. Users increasingly need to work with large amounts of content in today's applications. Whether this means products, books, log messages, emails, vacation rentals, or medical articles—the search box is the first place your users go to explore and find answers. Without intuitive search to answer questions in human terms, they'll be hopelessly lost. Thus, despite the maddening, seemingly mystical nature of search, you have to find solutions.

Relevant Search demystifies relevance. What exactly is relevance? It's at the root of the search engine's value proposition. *Relevance* is the art of ranking content for a search based on how much that content satisfies the needs of the user and the business. The devil is completely in the details. Ranking search results for what content? (Tweets? Products? Beanie Babies?) For what sorts of users? (Doctors? Tech-savvy shoppers?) For what types of searches? (Written in Japanese? Full of grocery brands? Filled with legal jargon?) What do those users expect? (A shopping experience? A library card catalog?) And what does your employer hope to get out of this interaction? (Money? Page views? Goodwill?) Search has become such a ubiquitous part of our applications, creeping in inch by inch without much fanfare. Answering these questions (getting relevance right) means the difference between an engaging user experience and one that disappoints.

1.1 Your goal: gaining the skills of a relevance engineer

How will you get there? *Relevant Search* teaches you the skills of a relevance engineer. A *relevance engineer* transforms the search engine into a seemingly smart system that understands the needs of users and the business. To do this, you'll teach the search engine your content's important features: attributes such as a restaurant's location, the words in a book's text, or the color of a dress shirt. With the right features in place, you can measure what matters to your users when they search: How far is the restaurant from me? Is this book about the topic I need help with? Will this shirt match the pants I just bought? These search-time ranking factors that measure what users care about are called *signals*. The ever-present challenge, you'll see, is selecting features and implementing signals that map to the needs of your users and business.

But technical wizardry is only part of the job (as shown in figure 1.1). Understanding what to implement can be more important than how to do so. Ironically, the relevance engineer rarely knows what "relevant" means for a given application. Instead, others—usually nontechnical colleagues—understand the content, business, and users' goals. You'll learn to advocate for a *relevance-centered enterprise* that uses this broader business expertise as well as user behavioral data to reveal the experience that users need from search.

We refine these concepts later in the chapter (and throughout this book). But to help set the right foundation, the remainder of this chapter defines the relevance problem. Why is relevance so hard? What attempts have been made to solve it? Then we'll switch gears to outline this book's approach to solving relevance.

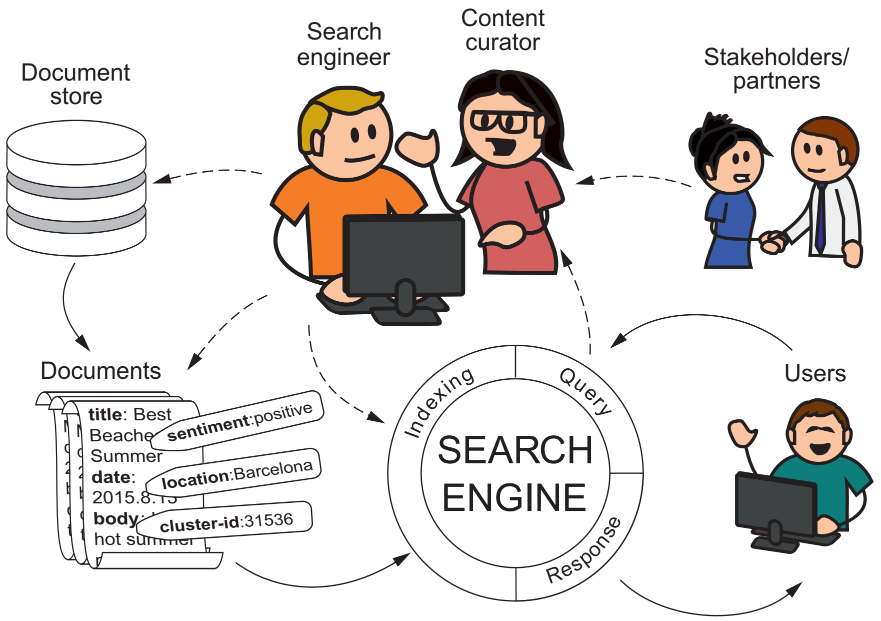


Figure 1.1 The relevance engineer works with the search engine and back-end technologies to express business-ranking logic. They collaborate on relevance closely with a cross-functional team and are informed heavily by user metrics.

1.2 Why is search relevance so hard?

Search relevance is such a hard problem in part because we take the act of *searching* for granted. Search applications take a user's search queries (the text typed into the search bar) and attempt to rank content by how likely it will satisfy.

This act occurs so frequently that it's barely noticed. Reflect on your own experiences. You probably woke up this morning, made your coffee, and started fiddling with your smartphone. You looked at the news, scanned Facebook, and checked your email. Before the coffee was even done brewing, you probably interacted with a dozen search applications without much thought. Did you send a message to a friend that you found in your phone's contact list? Search for a crucial email? Talk to Siri? Did you satisfy your curiosity with a Google search? Did you shop around for that dream 50-inch flat-screen TV on Amazon?

In a short time, you experienced the product of many thousands of hours of engineering effort. You engaged with the culmination of an even larger body of academic research that goes back a century in the field of information retrieval. Standing on the shoulders of giants, you sifted through millions of pieces of information—the entire human collection of information on the topic—and found the best reviewed and most popular TV in mere minutes.

Or maybe you didn't have such a great experience. It's just as likely that you found at least some of your search experiences frustrating. Maybe you couldn't find a contact on your phone because of a simple spelling mistake. Maybe the search engine didn't understand your idea of a dream TV. In frustration you gave up, uninstalling the application while thinking, "Why should a reasonable search be so difficult?"

In reality, a "simple" search that appears "reasonable" to users often requires extensive engineering work. Users expect a great deal out of search applications. Our search applications are asked, within the blink of an eye, to understand what information users want based on a few hastily entered search terms. To make it worse, users lack time to comb through dozens of search results. Users try your search a few fleeting times, quickly getting frustrated if it seems the search doesn't bring back what they're looking for. Your window for delivering relevant search results is small and always shrinking.

You might be thinking, "Sure the problem seems hard, but why isn't it easily solved?" Search has been around for a while; shouldn't a search engine such as Solr or Elasticsearch always return the right result? Or why not just send users to Google? Why won't a canned, commercial solution such as Amazon's A9 solve your search problems?

1.2.1 What's a "relevant" search result?

We're easily tricked into seeing search as a single problem. In reality, search applications differ greatly from one another. It's true that a typical search application lets the user enter text, filter through documents, and interact with a list of ranked results. But don't be fooled by superficial appearances. Each application has dramatically different relevance expectations. Let's look at some common classes of search applications to appreciate that your application likely has its own unique definition of relevance.

First, let's consider *web search*. As the web grew, early web search engines were easily tricked by unsavory sites. Shady site creators stuffed phrases into their pages to mislead the search engine. At best, early search engines returned any old match for a user query. At worst, they led users to spammy or malicious web pages.

Google realized that relevance for the web depended on trust, not just text. Users needed help sifting through the untrustworthy riffraff on the web. So Google developed its PageRank algorithm[1] to measure the trustworthiness of content. PageRank computes this trustworthiness score by determining how much the rest of the web links to a site. Using PageRank, Google brings back not only content that matches the user's search, but content that's seen as reliable and trustworthy by the rest of the web. This emphasis on returning trustworthy content continues today as Google plays a cat-and-mouse game with malicious websites that continually attempt to game the system.

[1] Read more at "The Anatomy of a Large-Scale Hypertextual Web Search Engine" by Sergey Brin and Lawrence Page at http://infolab.stanford.edu/~backrub/google.html.

Now let's contrast web search to *e-commerce*. A site such as Amazon, which has complete control over the content being searched, lacks the dire trustworthiness concern. Instead, what's relevant to e-commerce users is the same thing that matters to any kind of shopper: affordable, highly rated products that will satisfy them. But it's not just the shoppers that matter to a store. E-commerce sites have their own selfish interests. They must also return search results that generate profit, clear expiring inventory, and satisfy supplier relationships.

Search becomes the e-commerce site's salesperson. The same priorities that matter to the in-store sales experience must be programmed into the e-commerce search by the relevance engineer. The relevance engineer hopes to build a search that understands what shoppers want, so that they'll leave the store with satisfactory purchases. To e-commerce, relevant means not just leading users to satisfactory purchases, but also making a buck.

Still another kind of search, prominent in medicine, law, and research, digs deeper into text for its definition of relevance. This *expert search* depends on understanding jargon entered by specialists such as lawyers or doctors. These solutions must understand the subtle, domain-specific relationships—for instance, that "Heart Attack" is the same thing as "Myocardial Infarction". Or that acute "Myocardial Infarction" is a specific type of "Heart Attack".

Just as e-commerce search mirrors a shopper's interactions with a salesperson, expert search parallels a searcher's conversation with a research librarian. These librarians understand the lingo of specialized researchers. When asked a question, they guide specialists toward data and related research that specialists couldn't easily find on their own.

The basic definition of relevant to these search applications depends on solutions originally intended to organize information for libraries. For example, in medicine, the Medical Subject Headings (MeSH) taxonomy shown in figure 1.2 organizes medical concepts to help retrieve information on synonymous, more-specific, or less-specific subjects. To expert search, *relevant* means carefully linking subjects and topics between

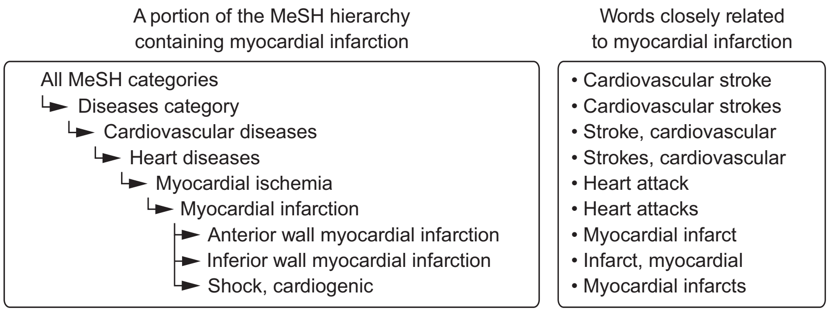


Figure 1.2 MeSH categorization of "Myocardial Infarction" (left) along with several MeSH topics closely related to "Myocardial Infarction"

search queries and content. A relevant result is something that delivers an "Aha!" moment to stuck researchers—a sudden insight they couldn't easily find on their own.

1.2.2 Search: there's no silver bullet!

The classes of search problems we've just discussed only scratch the surface in the amazing diversity of search. Is real-estate search a kind of e-commerce search? Certainly there's a resemblance (satisfying users with a satisfactory purchase), but many other factors come into play for a house buyer (good schools, neighborhood, number of bedrooms). What about a local restaurant search application? Or searching for groceries? Ordering food from a restaurant's menu? Searching volunteer opportunities? Or searching for someone to shovel the driveway after a snowstorm? What about intranet search? And what about your application? How do you define what's relevant?

Given this dramatic diversity of relevance requirements, it's surprising to find so many vendors eager to deliver a surefire, silver-bullet solution. Your definition of relevant is likely far more unique than you realize. Your users have expectations they may not even be aware of. Your content and business carry challenges you haven't appreciated yet.

Indeed, be grateful that Solr or Elasticsearch don't work well for your problem out of the box. You didn't choose a programming language because your product is just a module to import from its standard library. If that were true, there'd be nothing unique about your product! Rather, think of Solr or Elasticsearch as a search programming framework. An open source search engine lets you program *your* understanding of what's relevant into the search engine. We'll teach you just that: the art and science of delivering a relevance solution by using open source search technologies that satisfy users and meet business goals.

1.3 Gaining insight from relevance research

Okay, so you see that your application has its own definition of what's relevant. But why is there no universal, defined practice for delivering relevant search results to users? Search the web, and you'll find any number of one-off solutions that solved any author's problem particularly well. What you're not left with is a sense that search relevance has any holistic grounding or common engineering principles but is instead a bag of tricks that can't be generally applied.

In reality, there *is* a discipline behind relevance: the academic field of information retrieval. It has generally accepted practices to improve relevance broadly across many domains. But you've seen that what's relevant depends a great deal on your application. Given that, as we introduce information retrieval, think about how its general findings can be used to solve your narrower relevance problem.[2]

[2] For an introduction to the field of information retrieval, we highly recommend the classic text *Introduction to Information Retrieval* by Christopher D. Manning et al. (Cambridge University Press, 2008); see http://nlp.stanford.edu/IR-book/.

1.3.1 Information retrieval

Luckily, experts have been studying search for decades. The academic field of information retrieval focuses on the precise recall of information to satisfy a user's information need. What's an *information need*? Think of it as a *specification* of the ideal content that would satisfy the user's search. This specification goes beyond the search string itself. For example, consider a programming problem you're attempting to solve. You might be trying to figure out why the Java library function `sort` throws a `NullPointerException`. The information need could be specified as follows:

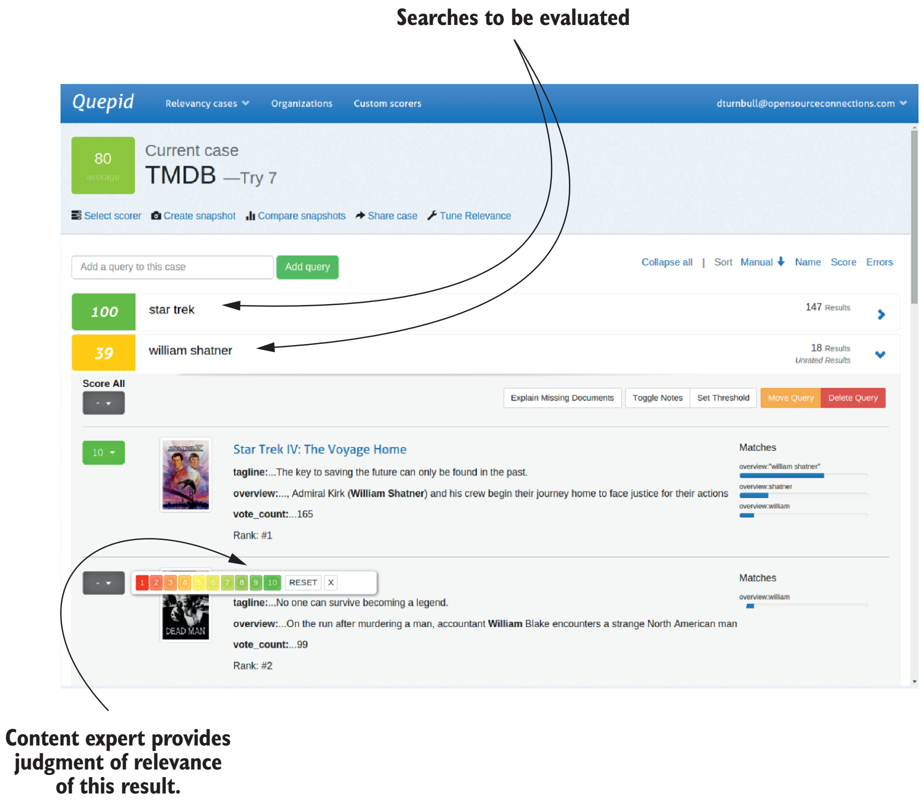
> A solution as to why my particular use of the `sort` method causes a `NullPointerException`. (Though I won't admit it to myself, it'd be nice to have some code to copy-paste that solved my problem so I can go to lunch!)

To satisfy this information need, you're likely to formulate search queries to find solutions to your particular problem—for example, "`sort` method `NullPointerException`" or "<code snippet> `NullPointerException`." If you're fortunate, you'll find a result addressing a problem similar to your own. That information will solve your problem, and you'll move on.

In information retrieval, *relevance* is defined as the practice of returning search results that most satisfy the user's information needs. Further, classic information retrieval focuses on text ranking. Many findings in information retrieval try to measure how likely a given article is going to be relevant to a user's text search. You'll learn about several of these invaluable methods throughout this book—as many of these findings are implemented in open source search engines.

To discover better text-searching methods, information retrieval researchers benchmark different strategies by using test collections of articles. These test collections include Amazon reviews, Reuters news articles, Usenet posts, and other similar, article-length data sets. To help benchmark relevance solutions, these collections have been heavily annotated in an experimental search setting, grading which results are most relevant for a given query. For example, when searching for "Mitt Romney," news articles about his 2008 or 2012 presidential run would be considered highly relevant. Perhaps articles about Romney's early management consulting work would be considered moderately relevant. Articles that discuss his father, George Romney, likely would be graded much less relevant. These annotated lists of search results that are relevant with respect to a set of queries are known as *judgment lists* (see figure 1.3).

Using judgment lists, researchers aim to measure whether changes to text relevance calculations improve the overall relevance of the results across every test collection. To classic information retrieval, a solution that improves a dozen text-heavy test collections 1% overall is a success. Rather than focusing on one particular problem in depth, information retrieval focuses on solving search for a broad set of problems.



Star Trek IV: The Voyage Home

tagline:...The key to saving the future can only be found in the past.

overview:..., Admiral Kirk (**William Shatner**) and his crew begin their journey home to face justice for their actions

vote_count:...165

Rank: #1

tagline:...No one can survive becoming a legend.

overview:...On the run after murdering a man, accountant **William** Blake encounters a strange North American man

vote_count:...99

Rank: #2

Figure 1.3 Example of making a relevance judgment for the query "Rambo" in Quepid, a judgment list management application

1.3.2 Can we use information retrieval to solve relevance?

You've already seen there's no silver bullet. But information retrieval does seem to systematically create relevance solutions. So ask yourself: Do these insights apply to your application? Does your application care about solutions that offer incremental, general improvements to searching article-length text? Would it be better to solve the specific problems faced by your application, here and now?

To be more precise, classic information retrieval begs several questions when brought to bear on applied relevance problems. Let's reflect on these questions to see where information retrieval research can help and where it might stop being helpful.

- *Do we care only about information needs?* For many applications, satisfying users' information needs isn't the only goal. Search exists just as much to satisfy the business behind the search application. You saw this with e-commerce earlier. Although it's often said "the customer is always right," it's also true that businesses can't function without selling ads, making a profit, satisfying suppliers, and moving inventory. Many incentives exist in any search experience that puts business needs above the user's information needs. Just like the used-car salesmen trying to move an overpriced clunker off the lot, relevance engineers must work with these factors to keep their employer in business.

- *What besides text reflects information needs?* Classic information retrieval focuses on a generic, one-size-fits-all measure of text relevance. These factors may not matter—at all—to your application. You need to focus with greater care on your specific problems. We discussed one example: how Google revolutionized web search by incorporating a numerical website trust measure (PageRank). Google uses PageRank to get around pure text-based measures easily gamed in its domain. Even text search doesn't always neatly fit into information retrieval's focus on article-length text. Good results for short text snippets such as tweets or titles require different thinking. You, not information retrieval researchers, must decide which factors matter to *your application*, and implement those. An approach that does poorly against the Reuters test set may be exactly what you need to satisfy your users.
- *What does the user experience imply about information needs?* Often the promises of the application itself influence what users consider relevant. We discussed expert search earlier. Consider two medical search applications. Both serve the same users (doctors). Both hold the same content (medical articles). But there's one important difference: one helps doctors serve sick patients at their bedsides, and the other allows doctors to explore their research interests casually in their offices. These dramatically different expectations mean a different understanding of what's relevant for the same search queries. A search for "heart attack" at the patient's bedside must provide actionable, reliable solutions to a dire, life-and-death problem. The research application allows for more variety: doctors search for "heart attack" to explore interesting and new research findings less tied to solving specific problems.

  Often the hardest part of being a relevance engineer is understanding the relationship between context and information needs. User searches arrive at your search engine with a great deal of baggage attached. This baggage comes in part as additional data, perhaps geolocation or user session. But other baggage is entirely implied in the promises made by the search application. Is the application built, sold, and marketed for sitting casually at one's desk and performing research? Or is it instead billed as almost an expert system, ready, willing, and able to solve any problem asked of it, including helping a doctor save a life?

Considering these questions, you can see that information retrieval builds a foundation for applying generally useful relevance measures to extremely broad classes of problems. Your job is to solve relevance for your application. As you'll see, much of this exists outside the realm of search technology and speaks to broader product strategy questions: Who are our users? What do they expect from this application? What implied and unspecified information needs will search need to address?

In fact, before we move on, let's refine our definition of *relevance* to what it takes to solve an applied relevance problem:

> *Relevance* is the practice of improving search results *for users* by satisfying *their information needs* in the context of a particular *user experience*, while balancing how ranking *impacts our business's needs.*

1.4 How do you solve relevance?

Informed now by information retrieval, let's focus on how to solve your relevance problems. Open source search engines recognize that what's relevant to your application depends on a broad range of factors. Many of these are application-specific (how far the user is from a restaurant, for instance). Others are broader, generic, text-ranking components from information retrieval.

Given the capabilities of open source search, how do you solve an applied relevance problem? What framework can we define that incorporates both the narrower, domain-specific factors alongside broader information-retrieval techniques?

To solve relevance, the relevance engineer:

1. Identifies salient *features* describing the content, the user, or the search query
2. Finds a way to tell the search engine about those features through extraction and enrichment
3. At search time, measures what's relevant to a user's search by crafting *signals*
4. Carefully balances the influence of multiple signals to rank results by manipulating the ranking function

This process is shown in figure 1.4.

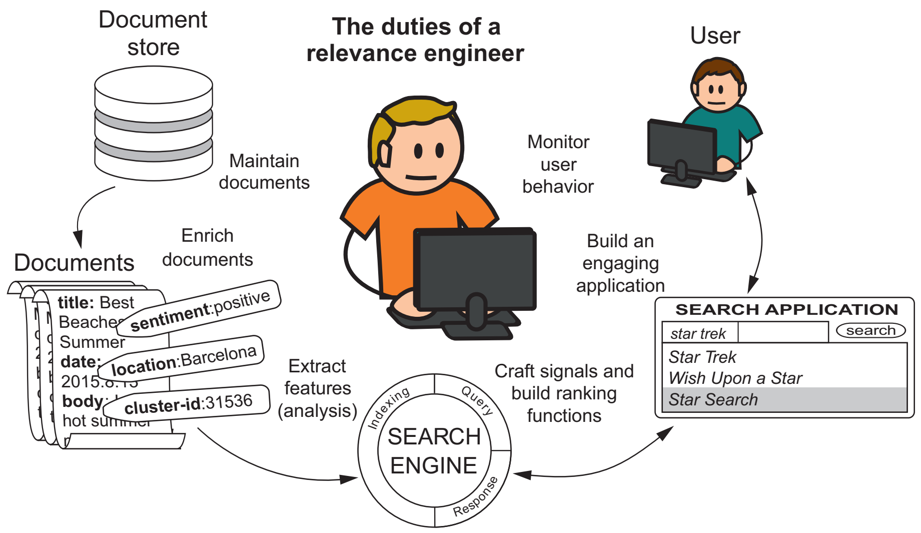


Figure 1.4 Relevance engineers select, enrich, or create important features from back-end systems and express ranking signals in terms of those features.

That sounds a bit abstract. What exactly do we mean? We discussed an example earlier: how Google susses out the feature of PageRank for websites (step 1). This feature is encoded in Google's search engine alongside each web page (thus achieving step 2). When you issue a search, Google measures many factors that you, with this search, consider relevant (step 3). For example, Google uses PageRank directly as a trustworthiness ranking signal. Other signals could include how frequently your search string is mentioned in a page's title/body or personalization factors using knowledge about your preferences. Google blends all of these signals (step 4) into a bigger ranking computation that orders search results in a way that it hopes you'll find satisfactory.

We discussed these ideas earlier in the chapter. But let's lay down some more-precise definitions. A *feature* is an attribute of the content or query. Features drive decisions. Much of the engineering work in search relevance is in *feature selection*—the act of discovering and generating features that give us the appropriate information when a user searches.

Those familiar with machine learning or classification may see something recognizable in these features. When performing classification, you identify new features of your data to make better classification decisions. Is a fruit a banana or an apple? If you know the color is yellow, there's a reasonable chance it's a banana. If you add data about the shape—round or long—then you can make an even more definitive decision. As you'll see, these features also help search solutions make definitive decisions about data.

Features describe, but what happens when users search? With *signals*, you program the search engine to rank by using your definition of what's relevant. Signals measure whether items are relevant for a given search (using features, of course!). For example, in our fruit search engine, the user might search for "yellow fruit." The search engine must evaluate whether a `Golden Delicious apple` might be relevant for this user. We know color matters to fruit shoppers, so one signal might measure how much this fruit's color corresponds to a color being searched for.

It's rare to have only one signal that measures relevance. More often, multiple signals combine to rank search results in the search engine's *ranking function*. For example, in addition to matching on color, perhaps the fruit shopper considers the freshness of produce. Or the user might recall preferred brands, using that as an additional signal. We'll teach you how to control the search engine's ranking function to rank results in a way that seems eerily "smart"—factoring in all the considerations (signals) that your users factor into their definitions of *relevant*.

Fear not—we know these ideas are abstract right now. As you get your hands dirty in future chapters, you'll begin to have the Aha! moment you need to grok what we mean. But to get the general idea, let's consider examples of features, and how they can be used as ranking-time search signals:

- *Sales data, user ratings*—Features used to signal popular results that users will probably be happier with.
- *Text with positional information*—Used to signal when phrases from the user's query match the content.

- *Text with synonyms*—Whether synonyms of query terms match the content.
- *Geolocation*—Whether something is near or far: Is the searcher close to the content? Is the sushi restaurant next to the user or in Manhattan?
- *Machine learning/classification features*—Is the search more easily classified into one type of content (a search for movies) and not easily classified into other types (a search for lawn equipment)?
- *Personalization/recommendation*—Has the user shown an affinity for any particular kind of content over others? Can you identify other users who are similar to the user making a search? Perhaps the historic preferences of the user issuing a search could be used as a signal to influence the search results.

As you work through future chapters, you'll see an approach that systematically improves search relevance based on selecting features and programming ranking signals. To form a foundation for this work, we'll first give you an overview of the search engine's internal mechanics and how to debug them in chapters 2 and 3. Chapters 4–7 get at the meaty problems of building features and signals. In chapter 8, we point out alternate strategies to guide users to relevant content when search by itself won't do.

Throughout this book, we use Elasticsearch as our example search engine. Elasticsearch is a modern search engine built upon Lucene, a commonly used Java search library. This book also applies to Solr, another search engine based on Lucene. Though our examples focus on Elasticsearch, these ideas are generally applicable. Solr readers in particular should follow along with appendix B, which helps map features between the two search engines.

1.5 More than technology: curation, collaboration, and feedback

Is a technical foundation enough to solve the search relevance problem? Armed with new skills from this book, you might be hungry to improve your employer's search. Targeting what you think are the biggest relevance problems, you deliver to your users what you consider to be an amazing search experience. You release your updates without much fuss; to the organization, that's yet another one of those heads-down, backend tasks that engineers go off and just figure out. It's something akin to squeezing more performance out of the SQL database, right?

Unfortunately, shortly after the release, your boss is at your door. Things look pretty grim. Despite your best efforts, something is deeply amiss. Somehow, users aren't making purchases. They can't find the information they need. Instead, they're giving up and going to the competition. With revenue headed south, your boss grits her teeth. In desperation, she looks at you square in the face and pleads for you to "make it more relevant!" In other words, fix the bug, implement the feature—stay all weekend if you have to; just make it work!

"Make it more relevant"? Let's recall our definition of relevance. Perhaps if you meditate on this definition, you'll see how the organization in this story misses the mark:

> *Relevance* is the practice of improving search results *for users* by satisfying *their information needs* in the context of a particular *user experience*, while balancing how ranking *impacts our business's needs.*

When you think about this definition, you quickly see that *relevance engineers have no idea what relevant search should be*! To satisfy your users' information needs, you need to understand their goals, their domains, and the context of their searches. These could vary wildly, from a doctor helping a struggling patient to a grandparent shopping for baby shower presents. Satisfying these users means getting inside their heads. Understanding these users goes far beyond search technology, touching nearly every competence in the organization. This is especially true as you work to understand business needs such as politics, profit, business goals, and other internal factors.

Solving the search relevance problem requires shifting the organization's culture to emphasize cross-functional collaboration. How can the organization teach relevance engineers to understand the users' vernacular and what they expect from search? What happens when the application is built for doctors or lawyers? Who helps the engineer understand these users' domains? How does the organization teach a relevance engineer what makes the company the most money? Which suppliers should be kept happy? What content has "premium" access in search (and what's that even supposed to mean)?

Even seemingly mundane search applications can be fraught with these complications. Consider a restaurant search application. Your marketing colleagues worked hard to bring users "into the doors" of your application. Now the search, acting as the site's salesperson (or perhaps concierge?), needs to satisfy them and make them eager to come back for more.

Relevance engineers, though, aren't the sales department. When a user types "sushi" into the search bar, what restaurants does that user expect? Takeout? High-end restaurants? Nearby ones? Depends on the user? Others in the organization, *not* the relevance engineer, understand what goals users hope to achieve. The relevance engineer is working in isolation to define relevance ranking and might as well be painting a house blindfolded.

Further, this collaboration goes beyond simply educating the relevance engineer. *Curation*, the manipulation of content to be easily found by user searches, can matter just as much as teaching a relevance engineer. Recall the expert search examples earlier in this chapter. Here the expertise of the librarian can help you build better search by organizing content to make it easier to find. Often this organization requires a close meeting of the minds between those who understand the content deeply and the relevance engineers who grok how the search engine works.

Rooted in these forms of collaboration is the notion of feedback. An effective organization strives to bring relevance engineers accurate and quick feedback to inform and guide their efforts. You can visualize several important feedback loops as a series of increasingly focused circles, as shown in Figure 1.5. Starting on the outermost loop, the search developers operate within an organization, blissfully unaware of the impact of search relevance. As the organization evolves, it moves to inner, more mature forms of feedback: incorporating user behavioral data and expert feedback. Finally, the organization encodes its wisdom into relevance tests, enabling test-driven relevancy practice—the most mature organizational form.

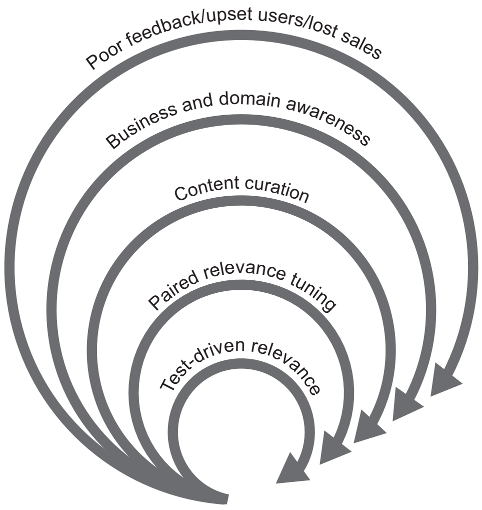

Poor feedback/upset users/lost sales
Business and domain awareness
Content curation
Paired relevance tuning
Test-driven relevance

Figure 1.5 Forms of search-relevance feedback

This book primarily teaches you about the technical craft of relevance engineers. But reflecting on what you *should* be doing hopefully echoes in your mind as you learn these technical lessons. In many examples, we state unequivocally that a particular search result is what users want to see. We do this to teach you technical skills to manipulate the search to get those results. As you work through those examples, remember the examples in this section before applying lessons directly to your relevance problems. We'll dive deeper into organizational challenges in chapter 10.

1.6 Summary

- Relevance problems are pervasive. Even established domains such as web search, e-commerce, and expert search continue to struggle to improve the relevance of search results.
- Bringing users to relevant search results can turn into a multibillion-dollar business advantage; failing to do so can mean losing out to the competition.
- *Information retrieval* is the academic field of bringing users to content that satisfies their information needs, largely as specified in search queries.

- In practice, *relevance* is more than satisfying information needs as specified by searches. It also means satisfying business needs. Further, understanding a user's information needs often depends on implicit information, such as the application's context, purpose, marketing, and user experience.
- Relevance can be achieved by identifying the valuable *features* of your content, and using those features to compute relevance signals.
- Technologists can't do it alone. Based on business needs, the user audience, and the content domain, the relevance engineer often doesn't have the skills to evaluate what content is relevant for user searches.
- Feedback is vital. From the perspective of the relevance engineer, measuring the impact of relevance changes helps avoid delivering poor search to users.

SEARCH

Relevant SEARCH

Turnbull • Berryman

Users are accustomed to and expect instant, relevant search results. To achieve this, you must master the search engine. Yet for many developers, relevance ranking is mysterious or confusing.

Relevant Search demystifies the subject and shows you that a search engine is a programmable relevance framework. Using Elasticsearch and Solr, it teaches you to express your business's ranking rules in this framework. You'll discover how to program relevance and how to incorporate secondary data sources, taxonomies, text analytics, and personalization. In practice, a relevance framework requires softer skills as well, such as collaborating with stakeholders to discover the right relevance requirements for your business. By the end, you'll be able to achieve a virtuous cycle of provable, measurable relevance improvements over a search product's lifetime.

What's Inside

- Techniques for debugging relevance
- Applying search engine features to real problems
- Using the user interface to guide searchers
- A systematic approach to relevance
- A business culture focused on improving search

For developers trying to build smarter search with Elasticsearch or Solr.

Doug Turnbull is lead relevance consultant at OpenSource Connections, where he frequently speaks and blogs. **John Berryman** is a data engineer at Eventbrite, where he specializes in recommendations and search.

To download their free eBook in PDF, ePub, and Kindle formats, owners of this book should visit www.manning.com/books/relevant-search

“One of the best and most engaging technical books I've ever read.”

—From the Foreword by Trey Grainger
Author of *Solr in Action*

“Will help you solve real-world search relevance problems for Lucene-based search engines.”

—Dimitrios Kouzis-Loukas
Bloomberg L.P.

“An inspiring book revealing the essence and mechanics of relevant search.”

—Ursin Stauss, Swiss Post

“Arms you with invaluable knowledge to temper the relevancy of search results and harness the powerful features provided by modern search engines.”

—Russ Cam, Elastic

Free eBook
SEE INSERT

ISBN-13: 978-1-61729-277-4
ISBN-10: 1-61729-277-X
54499
9 781617 292774

MANNING $44.99 / Can $51.99 [INCLUDING eBOOK]