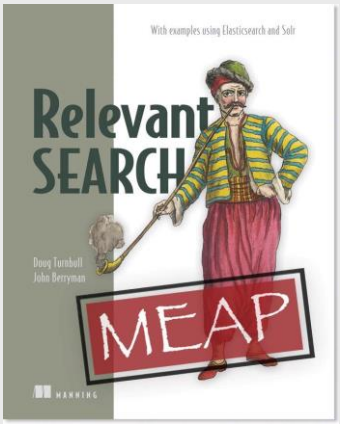

Solving an applied relevance problem

By Doug Turnbull

How do you solve an applied relevance problem? What process can you define that incorporates both some of the narrower, domain-specific data points that influence your relevance along with Information Retrieval techniques? In this article, we discuss the applied relevance problem.

This article is excerpted from Relevant Search. Save 39% on Taming Search with code 15dzamia at manning.com.

Modern search engines recognize that a broad range of factors come to bear in creating a relevant search experience. Information Retrieval techniques form the foundation, providing many of the text-based techniques. However, we've certainly seen that just relying on generic text-based factors may not entirely reflect what we consider a relevant search experience. Thus the search engine also provides tools ranking content with other factors, allowing us to incorporate richer domain-specific features into results ranking.

Given this combination of factors, how do we solve an applied relevance problem? What process can we define that incorporates both some of the narrower, domain-specific data points that influence our relevance along with Information Retrieval techniques?

The technical work of a relevance engineer includes these three primary tasks:

1. Identify important pieces of data in your content or about users that drive relevance
2. Find a way to tell the search engine about them
3. Carefully balance the impact of each piece of data with other pieces of data when asking the search engine to perform relevance ranking for a user's query

What exactly do we mean? It can be argued that often narrower, domain-specific methods or data metrics can sometimes matter more than Information Retrieval metrics.

Just as Google applied PageRank to enhance relevance, other search applications similarly have additional pieces of data we can add to enhance the search experience. In e-commerce, we could add user ratings, sales data, or other pieces of data to content to sell users valuable

products. Text, of course, can have a great deal of prominence. Search engines extract a great deal of information out of text, including how often a given word appears and where it appears in the text.

All of these items add a dimension to our relevance ranking. Those familiar with Machine Learning or Classification may see something recognizable in these pieces of data that drive decisions. When performing classification, for example, we similarly try to identify and incorporate new aspects of our data into our algorithms in order to make a better classification decision. Is a fruit a banana or apple? If we know the color is yellow, there's a reasonable chance it's a banana. If we add data about the shape – round or long – then we'll have even more information to make a definitive decision because lemons are also yellow. Each aspect of the data allows classification systems make more robust statistical decisions.

In the context of classification and machine learning, these different aspects of the data are known as features. A ***feature*** is an attribute of content that be used to make reliable decisions about that content. Features drive decisions – with a feature such as "fruit color" helping decide whether a fruit is a banana or an apple. Much of the engineering work in machine learning and search relevance is in ***feature selection*** – the act of discovering and generating appropriate features that give our algorithms enough information to accomplish a task.

In the search domain we need to use features to decide whether a piece of content is relevant for a user's search. With search, features can, in principle, be anything that concisely describes an aspect of the content (e.g., color, shape, taste in the case of fruit), but within the realm of search the most obvious features are the words themselves. Also there's an important difference between using features and machine learning. With search, features of content are used along with features about a user, the text query, and the user experience. Used together, features help decide which search results are relevant or irrelevant for a user and their search query.

PageRank is one such feature. Other features of content influence ranking by being compared to the user's query through a similarity measure. For example, Information Retrieval contains any number of relevance ranking formulas to determine the similarity between a user's query and a piece of text content – attempting to judge whether the text is about the user's query. Other features we might apply to search could include

- Sales ranking data, user rating – to drive users to products they'll be happier with
- Text features – used to determine whether a word in a user's query matches the content and how strongly
- Text w/ Positional Information – used to determine whether phrases from the user's query match the content
- Text w/ Synonyms – if we define a set of synonyms, used to define whether words in the query or synonyms of those words match the content

- Geolocation – Is the user close to the content? Is the sushi restaurant I found next to me or in Manhattan?
- Machine Learning/Classification Features – Is the query 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 their historic preferences couple be used to influence the search results.

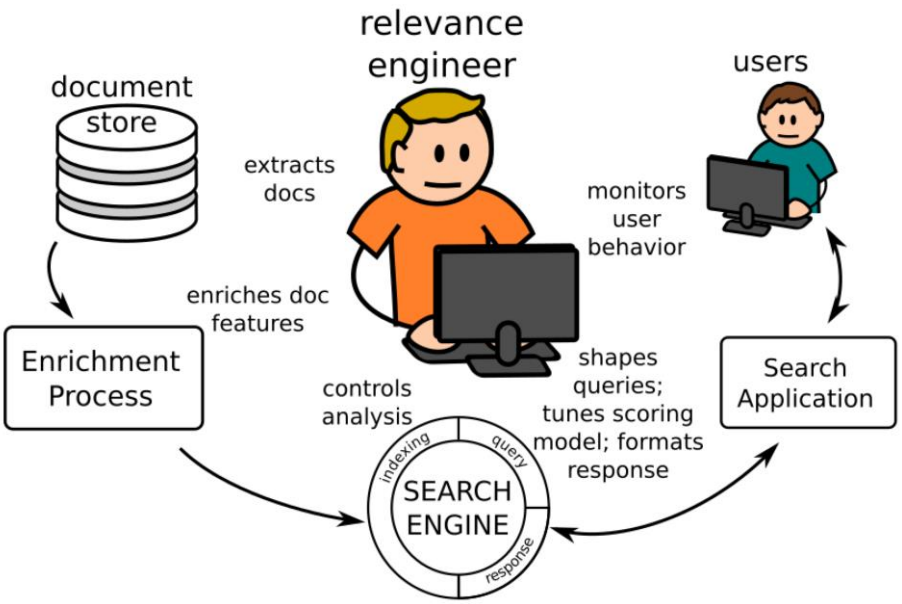

Figure 1 -- The search application. Relevance engineers select, enrich, or create important features from backend systems and express relevance ranking in terms of those features and information about the user

After pulling in the appropriate feature, the relevance engineer must then combine those features with the user's query to define relevancy ranking rules. Search engines are in part highly scalable ranking systems. This is particularly true with text search. Search engines have sophisticated data structures for finding and ranking content matching a search query. However, this is also true of other forms of ranking. Search engines expose a rich query DSL to express relevancy ranking and sorting in terms of multiple features that have been developed.

A Systematic Approach Using Features

Tying this all together, the job of the relevance engineer is:

1. **Identify Features:** Identify important features from your text or data you'd like to place into the search engine
2. **Extract and Transform Features:** Extract features from backend systems. Enrich these features with other systems, data sources, algorithms, or text analytics.
3. **Rank with Features:** Combine the user's query, other information about the user, to express relevancy ranking in terms of features the search engine's query interface.

The book *Taming Search* presents a systematic approach for identifying features that maximize the value of your search results. Further, as part of this systematic approach it points out ways you can derive interesting features from your text and data and finally load those features into the search engine. Sometimes those features will be simple metrics taken straight from your database and placed into the search engine, such as sales ranking data or user ratings.

Much of the time those features are simply text – text massaged into various representations that expose specific features. Verbs in text may be normalized to their root form so that a user searching for *run* can match documents that contain the word *running*. Occasionally, statistical or machine-learning approaches like classification and sentiment analysis are used to discover and extract additional features that provide important features to our relevance ranking.

Once important features are placed in the search engine the final problem becomes balancing and regulating their influence. Should text-based factors matter more than sales based factors? Should exact text matches matter more than synonym-based matches? What about metadata we glean from machine learning – how much weight should this play?

Throughout *Taming Search*, we use Elasticsearch as our search engine. Elasticsearch is a modern search engine built upon Lucene, a commonly used Java search library. You'll learn more about the underlying Lucene data structures, and how to leverage them to build features using Elasticsearch's analysis and indexing APIs. Further we use Elasticsearch's rich Query DSL to balance and regulate the influence of various content features on ranking the search result set. Though we focus on Elasticsearch, we argue this systematic approach to improving relevance will remain generally applicable.