Book of Search

The core book team

Chief Editor: Silvija Seres

Kjetil Halvorsen, Tony Hart, Damien Islam-Frénoy, John Puopolo, Pernille Wessel and Carine Zeier

FOREWORD

Dear reader,

We find search exciting. Really. We have seen, time and over again, how great search changes the way businesses operate, and the way people work. We have seen search become, in a matter of weeks, the main key to the most valuable in the enterprise -- its information assets.

In the industrial ages, effective management of natural resources was the key to success of organizations and companies; in this digital age, the same holds true for effective management of information resources. Then as now, is not enough to just *have* these resources; their value can only be judged by the ability to *use* them. It is good search that makes available information usable. It collects in, cross-connects it, analyses and normalizes, prepares and classifies, secures it and structures it. It helps the user express his intentions in a friendly, intuitive, and effective way. It helps the information owner to apply business goals to information, and simplifies the management of information. It moves companies to become market leaders, it creates efficiencies beyond what companies though possible. It satisfies users and grows traffic, finds stocks or jobs, helps prevent crime, finds the best restaurants and the latest ringtones. It enables information sharing, communication and reuse. Forget the "box and button"; think connected people and efficient businesses.

But all this joy of search aside -- to understand the true value of search, we invite you to join us on a technical roundtrip of some of the most important search features. Based on more than one hundred search-focused workshops, with some of the most demanding information consumers and providers in the world, we have built a large collection of Search Best Practices. We offer you a summary of these best practices in twelve short whitepapers, with use case examples, frequently asked questions, and some technical grounding. We have aimed to keep the papers relatively high-level, as an introduction to "how good search works". We have also aimed to be objective and product agnostic; we firmly believe that the more people understand search, the more they will also understand the beauty and power of our search product.

We hope you find the Book of Search enjoyable, educational and useful. If you should have any questions, we are always available to take a deeper dive into the world of search with you.

With best regards,

John M. Lervik

John M. Lervik
CEO, FAST, A Microsoft® Subsidiary

Bjørn Olstad

Bjørn Olstad
CTO, FAST, A Microsoft® Subsidiary

CONTENTS

Introduction

Introducing Search........................8

For the Business minded

Relevancy and Search..........................18
Linguistics and Search............................28
Query and Result Processing.......................................36
Content Refinement..........................46
Business Rules and Search Relevance...............................60
The Usability Factor..72

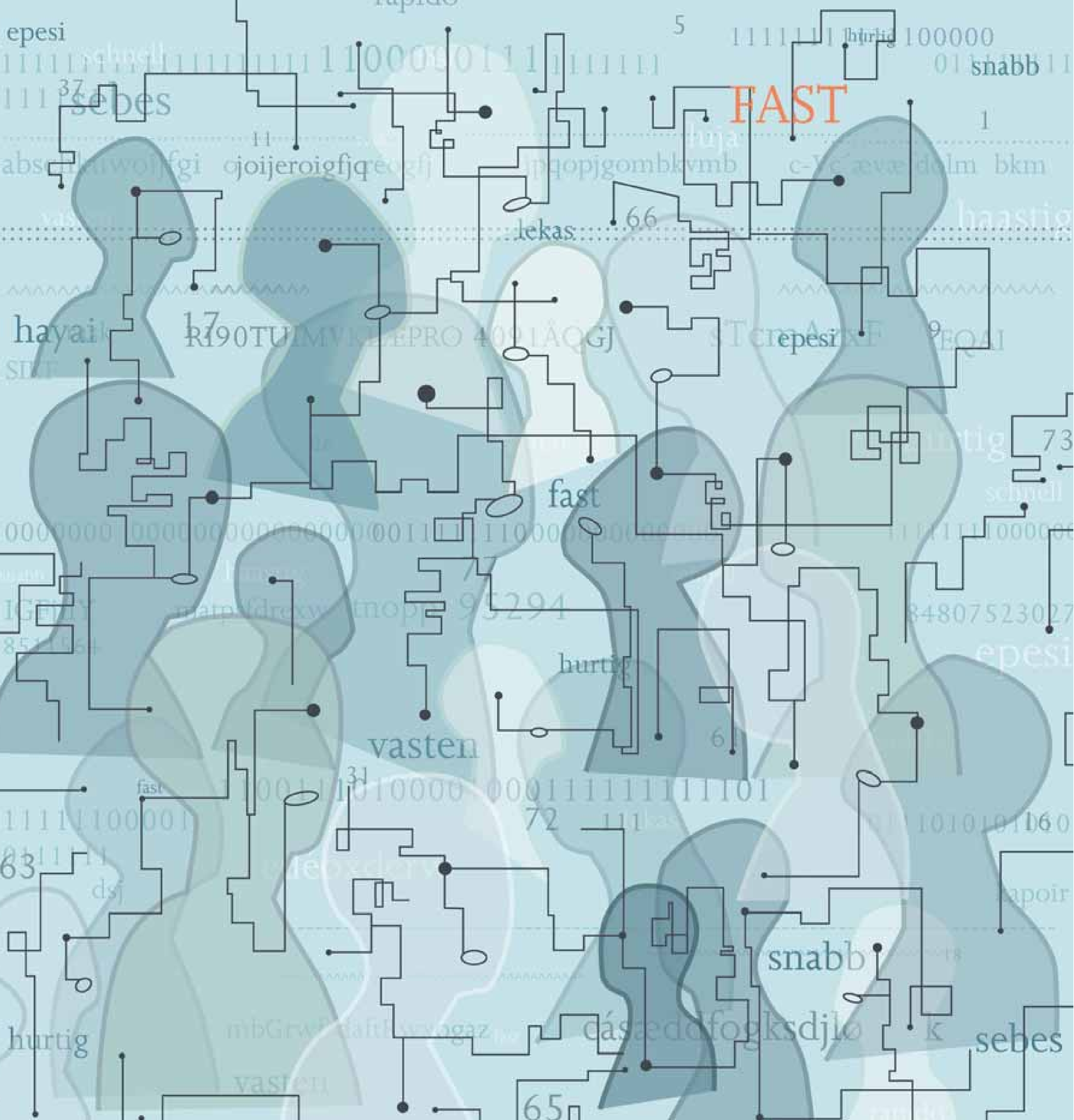
For the Technology minded

Integration..............................84
Security...........................96
High Performance Search....................................104
High-availability Search.................................116
Reporting and Benchmarking Search...126

Dictionary

Dictionary of Search Terms....................................136

Volume 1, Third Edition, 2008
www.fastsearch.com



rapido
epesi
5
snabb
FAST
sebes
joijeroigfjq
lekas
66
haastig
hayai
17
RI90TUIMVKLEPRO 4091AQGJ
epesi
fast
95294
hurtig
epesi
vasten
31
72
snabb
sebes
hurtig
vasten
65

Introducing Enterprise Search

Enterprise search helps organizations build new businesses based on the information they possess. The best of breed solutions are built as platforms, with many levers that make them exceptionally flexible, powerful and user-friendly for a wide variety of applications. The goal of this book if to explore some of these levers and to provide advice on how they can best be exploited.

Think of search. If you are like the majority of Internet users, you have probably thought of a "box and a button" application – a simple tool that transforms a short query into an endless list of results with variable quality. The ubiquity of the technology means that most people will have an idea in their minds of what search is. Words such as "search engine", "crawler" or "inverted index" may be used. Yet whilst at first glance it may appear a trivial and commoditized application, search is actually a flexible and feature rich platform. We would like you to think again – this time not in the general Internet context, but rather in the context of a specific enterprise.

In a setting where an ambitious enterprise needs to convert its most valuable asset – its data – into new business, many new shapes of search come to mind: real-time alerting, market and trend analysis tools, data clustering, advanced mobile services, or automatic video and audio broadcast monitoring. Intranet search has become the centerpiece of Knowledge Management and a unifying tool that binds the many information silos into one coherently searchable unit. Search drives portals, spanning across a range of devices and sources, serving an increasingly demanding and varying user group. Enterprise search is used as the basis for a new generation Business Intelligence solutions, as a user-driven data mining tool with sophisticated real-time analytics of very large data sets, and as an automatic data cleansing tool with advanced fuzzy matching and scoping abilities. It allows contextual insight into all the enterprise data, at an unprecedented level of precision, scale, and user friendliness.

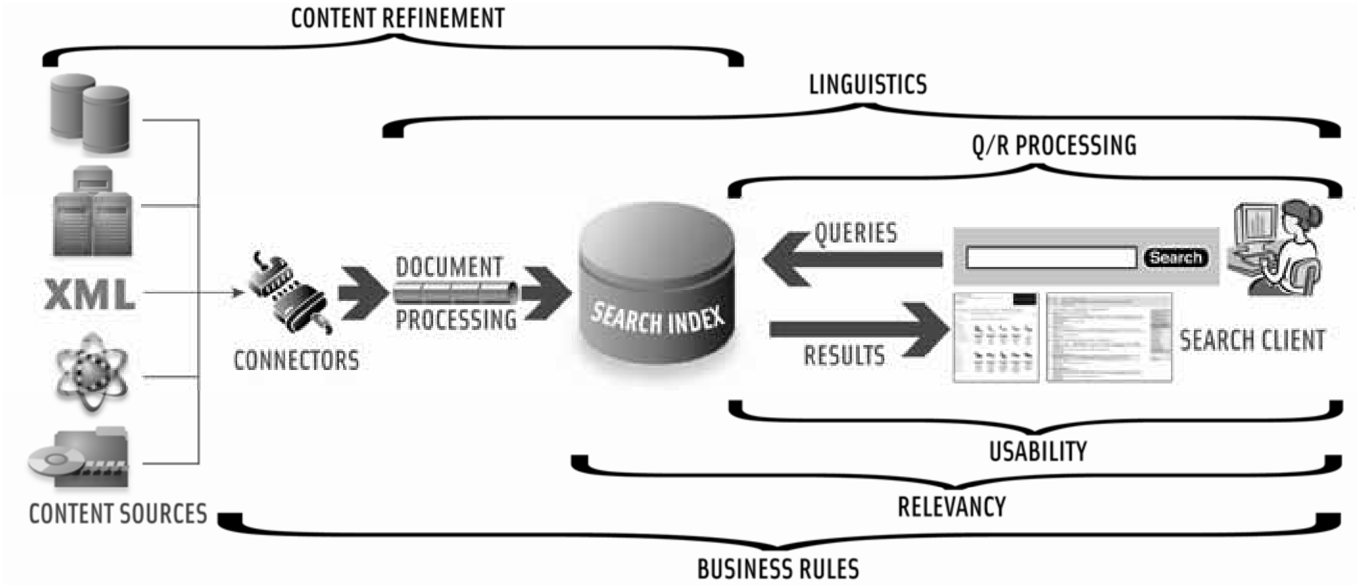
Enterprise search is far more than a "box and a button" application; for many enterprises, this rich technology is becoming the cornerstone of future strategy. By slicing, dicing and intelligently augmenting structured and unstructured information, highly contextually aware applications can enable everyone from simple web users, to demanding and skilled knowledge workers to maximize the use of the information available to them. It is used to enable a wide range of applications, some of which do not contain a standard input box: real-time alerting, market and trend analysis tools, data clustering, advanced mobile services, or automatic video and audio broadcast monitoring to name but a few.

Why is search important to you?

Search can be a truly strategic tool. Not only for organizations whose business areas relate to the Internet, but also for organizations in traditionally "offline" sectors. There are plenty of examples of search-driven improvements of business efficiency, both externally towards customers, and internally, towards employees and IT. Most importantly, good search allows fast development of new product opportunities, providing additional monetization opportunities and revenue streams.

As an example, we have seen that improved search has helped a number of large enterprise portals and eCommerce solutions achieve traffic increases, and associated revenue increases, of over 100%. Other companies have benefited from cost reductions. In one example, a company has realized a reduction from 55 high-end Sun servers to 26 commodity servers through migrating to a stare of the art search platform, reducing their TCO by well over 60%. Other companies have been able to establish leading syndication relationships requiring them to scale their search architecture to handle the significant increase in traffic, with a minimal investment in hardware. Yet another enterprise has achieved 90% savings in search deployments and maintenance by consolidating their databases, collaboration tools and several point search solutions to one single search platform, with a 2.5 times improvement in search efficiency. Others have extremely large data sets with complex analysis and retrieval requirements, with over 3000 billion discrete data points col-

Business aspects of search



lected from 200M data records including more than 5M records with live real-time feeds. Enterprise search has transformed this highly structured data into a next generation business analytics application.

" Attempt the end, and never stand to doubt; nothing's so hard but search will **find it out**. "

- Robert Herrick

Structure of the book

Deep down, every enterprise search solution is rooted in Information Retrieval (IR), the science of searching for information – in form of text, sound, images or data – in documents, relational stand-alone databases or hypertext networked databases such as the Internet or intranets. But this book does not aim to cover the matter from an abstract IR perspective, for two reasons: it is a broad interdisciplinary field best dealt with in an academic setting, and there are many excellent works on the topic; instead, we aim to write about practical and applied consequences of how IR is implemented by best in class applications in the hundreds of enterprises we have worked with – we will discuss which implementations work and which do not, and why. We aim to cover the topic at a pragmatic hands-on level, and we hope that you can start using the ideas from this book in your enterprise search solutions already today.

We have identified eleven areas of major importance for the success of an enterprise search application. For easier reading, we have divided them into two families of loosely connected topics: the business-related topics and technical topics underlying the different dynamics and make-up of great search applications.

From the business angle, we look at how organizations best leverage the wide range of built-in features of the search system to promote a better search experience and improve competitive positioning, to grow revenue or reduce costs, increase productivity and reduce risk. Features for the business-minded include relevancy, linguistics, query and result processing, content refinement, business rules and the usability of the system itself.

For the more technology minded, we cover the areas of integration, security, performance, high availability and benchmarking.

Considering the business requirements of search

Relevancy

The search engine's main job is to produce accurate and relevant results. A list of million results is of little use if the users only look at the first ten. Reversely, in a monitoring setting, incomplete result lists are of little use. Users are becoming increasingly demanding, especially in an enterprise setting. User requirements vary over time, over user groups, over application contexts, and over content categories. Thus, en-

> GET THE FACTS, OR THE FACTSWILL GET YOU. AND WHEN YOU GET THEM, GET THEM RIGHT, OR THEY WILL GET YOU WRONG. *Dr. Thomas Fuller*

terprises must have ability to tune the search engine to meet end-user expectations and business needs.

Enterprise search solutions are evolving far beyond standard Web search capabilities. Ranking models are based on multi-faceted quality measurements of the match between query and document. Relevancy is determined by concepts, and additional levels of abstraction such as context, freshness, completeness, authority, statistics, quality and geography. A ranking model is the independent tuning of each element relative to the business need, to determine whether a document is a good match to a query.

Linguistics

Linguistics deals with the structure and variation of languages to improve the user's ability to find relevant information. This improvement is applied both to user's query and to the information stored in the index. The standard linguistic tools apply not only to languages, but also to industry-specific or enterprise-specific terminology.

The linguistic optimization tools include automatic language detection, grammatical normal forms (also called lemmatization), and the use of synonyms. For Asian languages, such as Mandarin Chinese, tokenization algorithms are used. More advanced interpretation of language can be carried out with entity extraction, recognition of parts of speech, categorization, unsupervised clustering of documents, and sentiment analysis. Queries are often improved with spelling correction, recognition and grouping of idioms such as "home run" (also called phrasing), and identification of word sequences in queries that are irrelevant to the search (also called anti-phrasing).

Query and result processing

Search engines are faced with two basic information retrieval issues: first, how to help users craft better questions, and second, how to provide better results, minimizing what the user has to read through. To deliver consistently superior results, one must understand the intent of the query, know what information is available, how the different information sources inter-relate, and identify where the appropriate information is located. As the results are returned to the user, they must be formatted and presented in a way that makes the user's experience easy and rewarding at the same time.

These goals can be accomplished by using two key technologies: Natural Language Processing (NLP) and linguistic analysis. NLP interprets queries posed as questions and phrases by stripping out irrelevant terms, and identifying the query intent. Linguistic tools include capabilities such as avoiding word-sense ambiguity to distinguish between, for example, the color orange, the company Orange and the citrus fruit. There are many other means for query improvement. Search applications house these technologies in the query/result processing stage, with the two-fold goal: to analyze human language in the query to identify the searcher's context and intent, and to return the most relevant set of results.

Content refinement

The quality of the search experience depends greatly on the search engine content. But the old adage "garbage in, garbage out" does not need to haunt searchers! In an enterprise search setting, the content can be greatly improved before it is made searchable. The content refinement lifecycle includes two stages: content aggregation and processing.

Content aggregation brings together content from multiple sources; it is not uncommon to aggregate from thousands of different silos of structured and unstructured data, from databases and SAP portals to emails and multimedia content, sometimes facing more than 300 different format types. Content is made available to the search engine via the content API that acts as an information broker. It pulls content from the data source (database, CMS application, etc.) during scheduled calling requests and sends this content into the search engine. It can also work on a push basis, and performs incremental updates.

Document processing is the analysis, conversion, transformation, and enrichment of original content for the purposes of indexing and subsequent retrieval. It can be made up of one or more document processing stages – for instance, a pipeline of stages could consist of language detection, synonyms, spell checking, lemmatization, taxonomy classification, and custom plug-ins.

Business rules

Business rules are algorithms, workflows, or heuristics that are implemented and supported by a software system. For example, a business rule might be that a credit check is not necessary for returning customers, or that a platinum-level advertiser is always ranked higher than the gold-level one. Search applications allow enterprises to apply such business rules at various stages of the search, from document ingestion, ranking, query transformation, to result processing, to best align the information provided to users with the business goals of the enterprise. Business rules are also used at alerting time, where preconfigured queries push results to the user as soon as new and relevant information becomes available.

Analyzing search activity in the context of business rules enables search providers to adjust their relevancy model, and guide users to the best business-generating pages. For example, in the case of a pharmaceuticals manufacturer, the analysis of query logs will highlight the difference in information needs for users in different functions, such as R&D, clinical trials, and sales and marketing. The analysis may highlight the need for custom dictionaries (for specific industry terminology) or linguistic capabilities to reduce the number of queries or zero hits. In this scenario, business rules are effectively being used for "fault detection".

Usability

"If the user can't find it, it ain't there". Good search usability is an essential part of realizing the system's full potential. It describes how the user is guided through the system from start to finish. When establishing or revising a search-powered system, usability is ultimately judged by the users (business, consumer, administrators, etc.) themselves, so search providers should follow a user-oriented design process. Leveraging reporting and monitoring tools, such as click-through trends, page impressions and abandonment points will provide quantifiable metrics on the success of the system.

Good usability is provided by defining the search experience, aligning the system design with the definition and then letting real end users test and evaluate it. The user search experience can also impact both motivation and productivity. It is vital to make the search box itself easy to find, to provide different ways to search, and to match advanced search capabilities to users' needs and abilities.

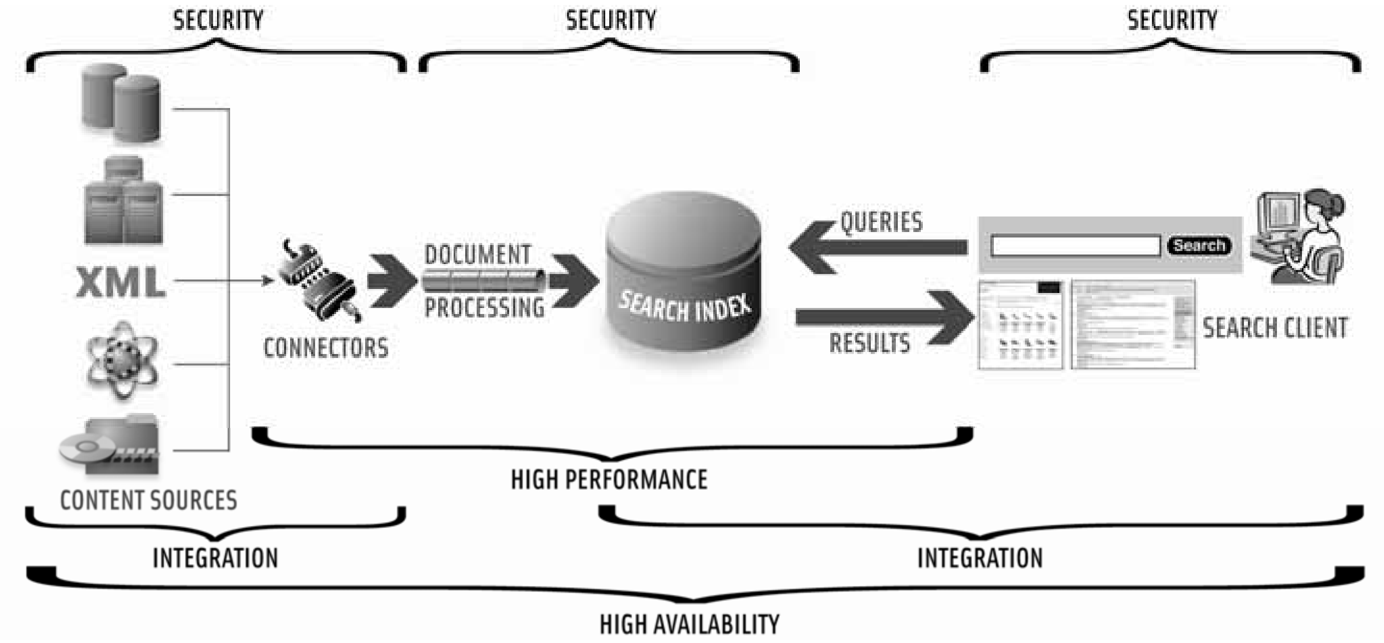
Considering the technical aspects of search

Security

Security guidelines apply to search platforms in three areas. First, they must recognize permission levels on documents stored in the search engine, and recognize end users' identity at query time. Second, enterprise search engines must validate that all query requests are issued by authorized clients and that document-gathering connectors respect every content repository's access model. Third, when executing a query, search software must always align the user's permission rights with the permission levels of the content they can see in the result list.

The most common corporate practice to ensure privacy is using folder- and document-level access control within applications and repositories. This access control logic must then be respected by other applications connecting to the content, including search engines. In this way the search engine becomes the gateway and the gatekeeper to valuable and confidential corporate data.

Technical aspects of search



Integration

Integration deals with embedding search engines in third-party software applications, such as document management systems (DMS), customer relationship management systems (CRM), or business intelligence tools.

There are two increasingly common scenarios for search integration. One is in an authoring or management application such as a DMS, where stored content has to be searchable – in a fast, flexible and easy manner. The other is in an industry-specific workflow and investigation tool (for example, for compliance in the financial-services sector) where multiple external data sources are searched and processed within the application in question. OEM integration requires substantial planning to provide a seamless assimilation of the two technologies. The component architecture has to be understood so that each connection point is identified and treated separately.

> "Knowledge in the form of an informational commodity indispensable to productive power is already, and will continue to be, a major – perhaps the major – stake in the worldwide competition for power."

There are five main areas to consider: content creation for indexing, index configuration, query logic, user interface design, and administration and configuration. The most important part of integration is deciding which configuration or query features to expose to the content consumers and managers. Top-quality enterprise search engines are flexible tools, designed to cope with many different types of applications and industries – e-commerce, knowledge management, archiving, video search, and more. So for each design and configuration decision made, the OEM must decide whether the decision applies to all clients or whether the option must be left open for systems integrators or IT administrators to fine-tune the system.

Performance

Providing a scalable, fault-tolerant and high-performing search calls for selecting the correct configurations of software and hardware. It starts with identifying the most important requirements and key metrics and mapping these to the appropriate system architecture.

Search providers across different business segments have very different needs. News and financial search require fresh data that is indexed near-real time, while litigation support services, for instance, require batched data, indexed once. The key to optimizing a system lies in understanding the user's objectives, while sizing and designing to strike the right balance between speed, size, and cost.

The performance metrics include the total number of documents to be indexed, the required ingestion rate and acceptable indexing latency, and the number of queries per second (QPS). Akin to grid computing and distributed architectures to support scalable enterprise software, high-performance search grids use replication and distribution of servers to scale along the dimensions of document volume and QPS. A third dimension is document ingestion rate, which scales with the resources allocated to content aggregation and processing. Systems using these three dimensions should be able to scale linearly, independently, and simultaneously to achieve the desired performance targets.

High availability

Critical IT systems are often described as fault-tolerant, redundant, or displaying high availability. In other words, should something go wrong – power cut-offs, hardware failure, corruption of data, say – the systems have been designed to maintain certain levels of service.

A common solution is to architect more hardware to mirror vital elements of the system. The downside is the price of the equipment and the internal cost of managing a duplicate setup. The extra expense of redundancy should be evaluated in light of the loss of revenue from the downtime and the odds of fatal errors occurring in the system. In this scenario, search providers should differentiate between availability on the content ingestion side (where the requirements may be less stringent, as servers have no problem with picking up an indexing thread seamlessly after they come up again) and on the user query side (where down-time may be much more costly for the business).

Failures that can impair the search application are divided into two areas: backbone failures, and component and service failures. Backbone failures include various hardware incidents and network outages. In most cases, there are corporate- or service-wide policies regarding power supply and data-center security, and search typically complies with these policies. But in cases where query uptime is critical, hardware redundancy should be combined with intelligent recovery operations.

Benchmarking

How good is your enterprise search? And how can you make it even better? Without adequate measurements, improvements are hard to plan and hard to prove. Reporting and benchmarking creates a structured method for measuring and validating the success of search with respect to these varied stakeholders.

So what should be measured? Search metrics can be divided into two categories: the basic search engine metrics, such as the volume of searchable data in the index and hardware performance, and search usage metrics. The first category includes the number of documents in target repositories, an audit of those documents, and information on hardware usage. Usage measurements include hard numbers such as click-counting and subjective measurements about the quality of the interface and the results ranking. The most popular queries, number of zero hits, click distance and user surveys are often used to evaluate the quality of search.

These categories are further split up according to stakeholder groups and the reports relevant to each. Identifying which group has an active interest in various metrics can be used to assign the ownership of search tracking and tuning to different groups; different groups may be responsible for identifying sources of content, for determining what good and bad results are, and for building and maintaining the platform.

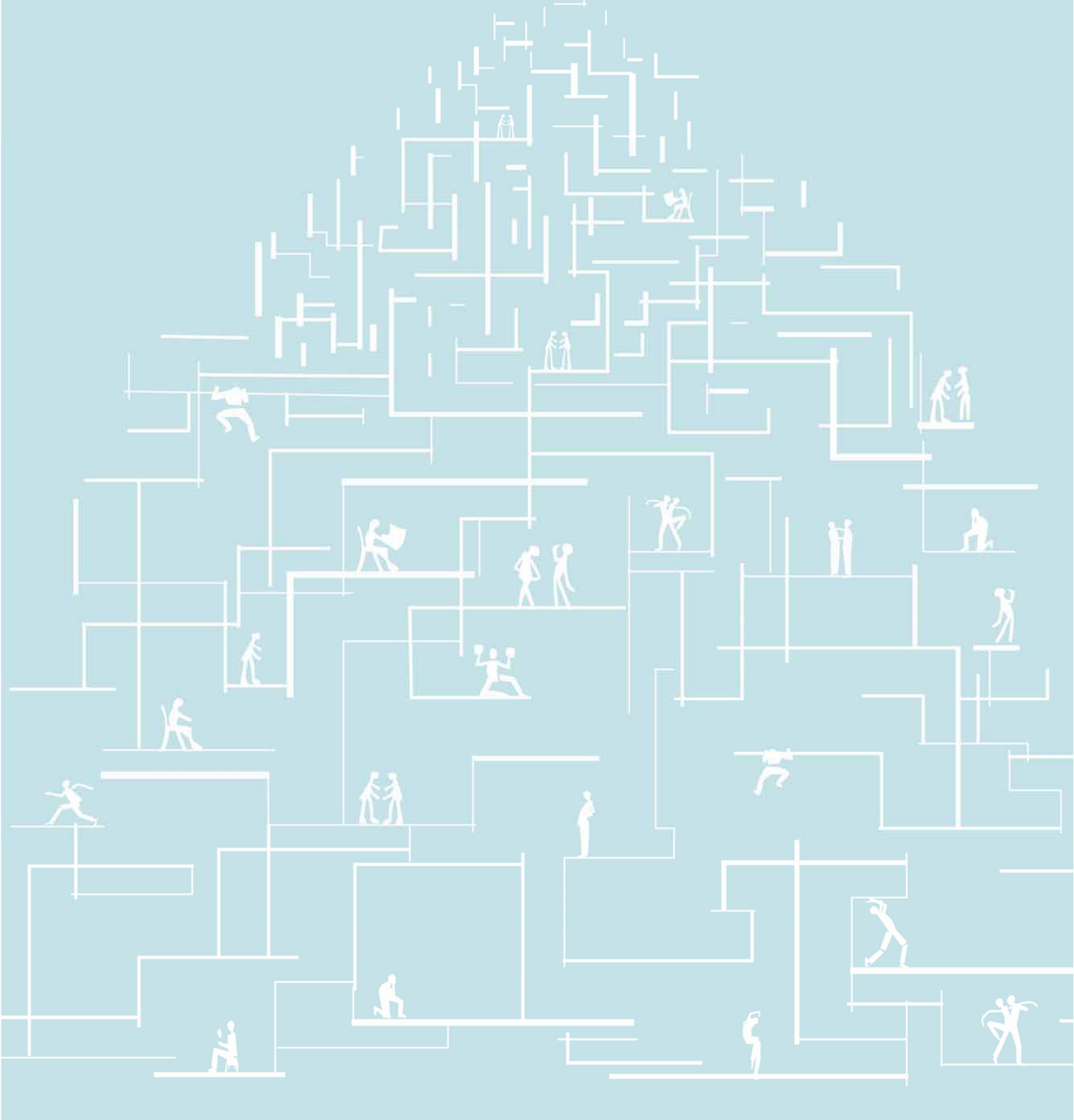
What is good search?

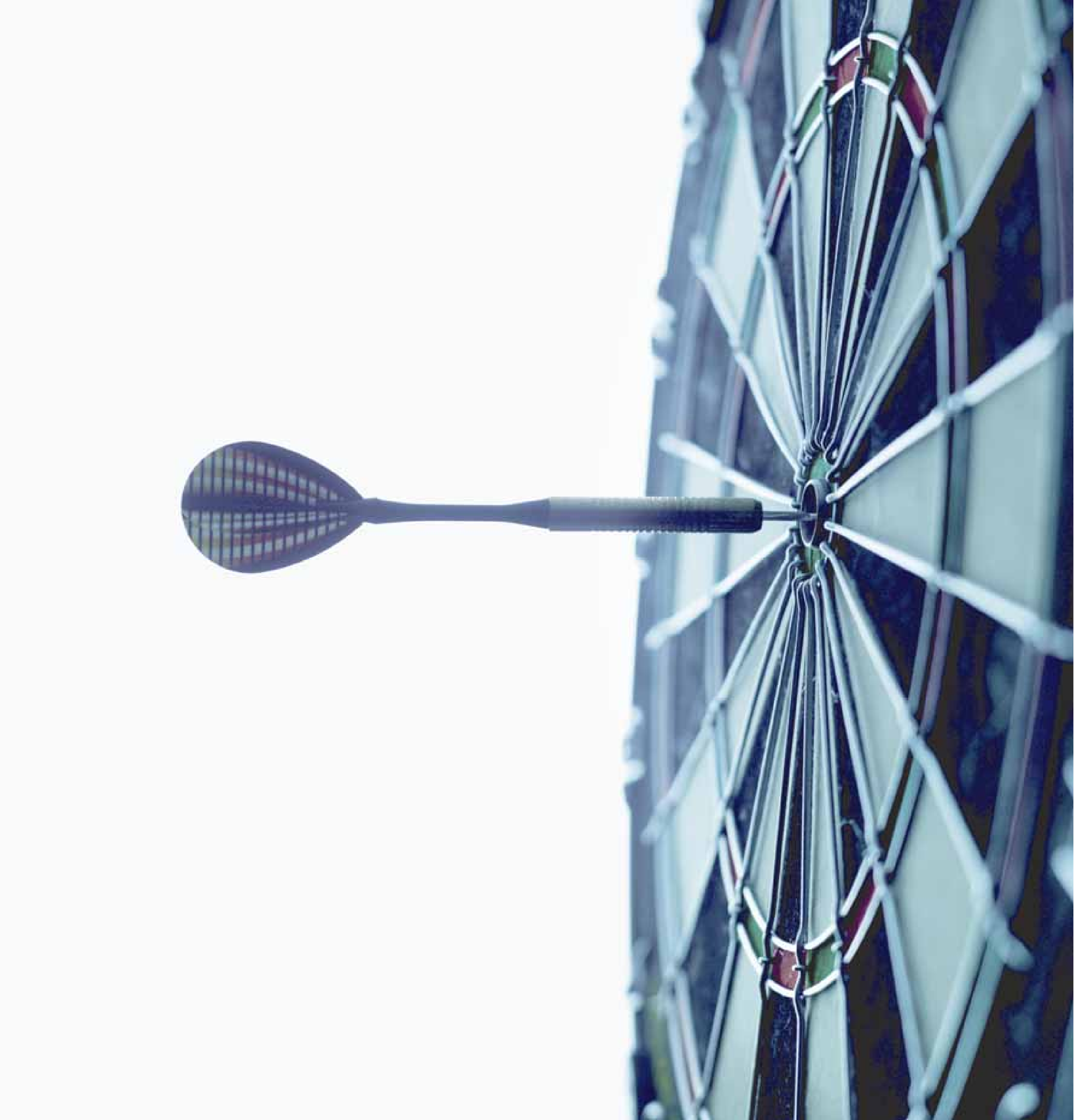
Search, with its users and its capabilities, is changing; so is its role in the enterprise world. People want answers, not references. Businesses want differentiating solutions. Extreme scalability has become a norm for businesses. Rapid deployment of new business models is necessary to fend off aggressive competitors in the aggressively converging online markets. Good search is becoming universal: it is everywhere, all the time. Technical boundaries are disappearing, for example between internet and intranet, and structured data versus unstructured data. Information is becoming source and device independent, between servers, desktops, and mobile. Search is pervasive: it permeates more and more areas of people's lives. It is contextually aware of its surroundings. It is proactive rather than reactive. It suggests and alerts and informs. Search is a necessity. Consumers demand more, governments require more, and competitors know more. With search, a higher standard of information emerges with greater expectations. So, how can you cope?

Developing an enterprise-class search that is efficient, user friendly and future proof is not difficult – it just involves lots of choices and good planning, including a clear appreciation of the available tools and necessary trade-offs.

In a world where one size does not fit all, there are many opportunities to customize each of the components from a rich suite of best platforms to suit the needs of each company and its users in their specific industries, with their specific technical requirements, usage patterns, and business goals. However, organizations are starting to realize that the effort is worthwhile; in many a major enterprise, the enterprise search platform has become every bit as important for their business operations as the ERP or CRM applications. In the past three years, we have been a proud part of the global team that has seen a transformation of enterprise search into an IT infrastructure necessity, and a key enabler for mission-critical applications – and with this book, we wish to share our experiences and enthusiasm for this powerful business tool with you.

In particular, the following eleven chapters provide an introduction to understanding the key areas that compose search, and underline the best-practices for building a state of the art system. We hope you enjoy!





Relevancy and Search

The search engine's main job is to produce accurate and relevant results. A list of million results is of little use if the users only look at the first ten.

In most companies, the role of search has evolved from basic keyword search in external Web pages to include comprehensive information retrieval solutions – static and dynamic – that cover a wide range of internal and external systems, applications, and networks. As a result, there is much more demand for sophisticated information retrieval functionality and performance.

Essentially, search engines must produce accurate and relevant results to meet the expectations of increasingly demanding users and applications – and to meet the needs of the business. That means it must now be possible to tune the relevance of the search engine.

Search technology relies on assessment tools called *ranking models* to determine how closely content matches a particular query and whether it should be included in the search results. However, the ranking models of most search engines are inaccessible, so it's difficult and sometimes impossible to alter or tune them to meet different needs. In every case, they use the same yardstick.

However, search is not limited to one uniform environment. Business users search in multiple contexts: e-commerce sites, corporate Web sites, intranets, extranets, portals, etc. Each has distinctive objectives and each user community values content differently. So it makes sense to be able to adjust the

5 things you should know about relevancy

1. Search users search in order to use information – not for the sake of searching.

2. Relevancy success depends on understanding the context and characteristics of the search user.

3. Adjusting relevancy models is like adjusting the graphic equalizer on an audio system – key components can be adjusted independently.

4. Linguistic tools can improve precision and recall.

5. Relevancy can be tested and tuned by using a "golden set" of well-known documents and queries.

yardstick used to evaluate content in order to get results that are aligned with the objectives of the search context and needs of the users.

What determines relevance? The relevance of a document is represented by a number or ranking value that helps determine how closely the document matches the characteristics implied by the query. The value is constructed from a number of factors based on the detailed analysis of all parts of the document. These include, for example, title, author, date, body, meta-tags, key concepts, classification, etc. These factors can be tuned independently, or in groups, to create ranking models for particular content – for instance, where the date of a highly relevant news article may be ranked higher than for news on the same topic.

Most search technologies employ a fixed ranking model that is designed for generalized use in a common context. A fixed model works very well within its frame of reference because it's optimized for that specific content.

However, as you move away from its design base, a fixed model rapidly loses its effectiveness. An example: a public Web search solution would use a relevancy model that is geared to rank Web pages. However, it cannot effectively deliver the goods in an e-commerce context because the evaluation mechanism is incorrectly calibrated for e-commerce searches and usually can't be adjusted.

Similarly, in a knowledge management setting, many documents come from sources other than the Web: document management systems, management systems, e-mail, file systems, etc. These documents need a different "quality comparison" mechanism.

Multiple dimensions enable full control

Enterprise search solutions are evolving far beyond standard Web search capabilities. They are now able to base ranking models on multi-faceted and tunable measurements of the quality or the match between the query and a possible result document. In this environment, relevancy is determined partly on the parameters discussed above, but also on concepts, sentiment, or additional levels of abstraction, such as freshness, completeness, authority, statistics, quality, and geography. (We will describe these levels in more detail later).

A multi-parameter model enables full control of the relative effect of each ranking component for a given query. Ideally, the search solution should provide a set of pre-defined relevancy model profiles that align with specific uses or audiences – site search, news, shopping, self-service, market intelligence, surveillance, etc. Organizations need to be able to apply these types of capabilities "out-of-the-box" or be able to easily tune relevancy models so they can be optimized for their target audiences. For example, for a Web site, page popularity is key and should be a priority, whereas for a news-monitoring application, freshness should be the primary factor, and sentiment context is important for market intelligence applications.

> "The only good is knowledge and the only evil is ignorance. "
>
> - *Socrates*

A convenient way to understand the importance of relevancy models is to visualize a graphic equalizer on an audio system, which has pre-sets for audio environments such as concert hall, car, home, classical, and rock, for example, and which also allows for individual adjustment to meet the needs of the listener. Similarly, search solutions must be capable of providing pre-set relevancy models where each of the parameters can be independently adjusted, and a change in one does not affect the others.

Many factors affect the relevancy of a document with respect to a given query. In general, it is impacted by the content of the document, the language in which the content is written, the degree to which *lemmatization* is employed, the extent to which stop words are removed, and the degree to which *synonym expansion*, thesauri, *link cardinality*, *proximity boosting*, and *dynamic ranking* are utilized. It should be noted that a document's relevancy with respect to a query is not necessarily decided on the basis of words common to both query and document, but rather by the extent to which its content satisfies the user's need for information.

The tuning parameters that define relevancy ranking models can have a profound impact on the search results themselves, and also on the associated business environment. Six key parameters stand out:

Freshness – how fresh is the document compared to the time of query? A number of new search applications require sub-second or non-stop updates to the index, so traditional batch processing is not an option. These new-generation solutions need to be able to search for news, receive stock alerts, or update the index with new products or pricing. In such cases, freshness is of paramount importance to users.

Completeness – how well does the query match superior document contexts such as a document title or URL? What matches the query? Is it the document title, the author, a mention in the body text, metadata linked to the document, root and expanded forms of words, etc.? For example, if the query is "Boston College", then results citing "Boston University" would come up less readily if completeness is a key factor to the rank profile.

Authority – is the document considered an authority for this query? Many items can be part of the analysis of documents to determine this parameter – items such as Web link cardinality, article references, page impressions, and product revenue, to name a few. For example, with link cardinality, the search application will monitor the number of links in or out of a document, with high frequencies of links indicating the authority of a document.

Statistics – how well does the content of the overall document match the query? One simple example is the number of times the query term appears in the document. Another is the proximity of the words in the document – how close they are to one another.

> "The Alchemists, in their search of gold, discovered many things of greater value."

Quality – what's the quality of the document? How important is it when viewed from the perspective of the content owner or search application? For a corporate Web site, product landing pages and press releases are typical of the types of documents that have higher "quality" ratings. For a market intelligence application, documents from news services and industry analysts may be considered more important.

Location – how important is location in relation to the query term? For an Internet Yellow Pages company, there's significant value in being able to maintain location entities at a contextual level to enable extreme precision in search and contextual navigation. The feature also offers and user stickiness – for example, if a user is searching for "BMW dealership within 15km of my workplace".

Tuning relevancy to meet the business need

Tuning the relevancy of search can lead to significant improvements not only for the search user (improved results to query terms), but also for the underlying business application (enhanced revenues, providing transparency for compliance initiatives, etc.).

I want one search platform to support my Internet, extranet, and intranet presence. How do I ensure that users receive relevant results?

If you use multiple rank profiles (with different relevancy parameters) based on the user's information needs, you'll be able to fully control the weight of each rank component for a given query. Information can be locked down so that only the users with the correct security profiles can gain access.

One caveat, however: simply maximizing each of these controls will generally not lead to improved relevancy. If they are to achieve the goal of relevancy - balancing recall and precision - organizations need to understand the search application, the underlying information, and the search user. In general, customers need to strike a balance between recall, (finding everything related to a query) and precision (finding only those documents or entities that relates to a given query). More scientifically, *recall* is the ratio of the number of relevant records retrieved to the total number of relevant records in the index. *Precision* is the ratio of the number of relevant records retrieved to the total number of irrelevant and relevant records retrieved.

When using search systems, the search application must strike a balance between recall and precision. The aim is to dramatically improve precision without sacrificing recall.

For example, *knowledge discovery* or *compliance* applications (upper left of the curve in the diagram on the next page) will rate recall as being more important than precision – in other words, customers do not want to "miss" any important documents when performing compliance-related searches. Here, customers may want to enhance the importance of the *completeness* and *statistical parameters*.

But in an *e-commerce* or *e-directory* environment, users prefer much more precise search results so that customers are not swamped with too many non-specific results. For instance, searching for "iPod" should return only results that are Apple Computer's iPod players or accessories, not the similar players made by Creative or Sony. The business objective is to convert the browsing/researching user into a paying customer in as few clicks as possible.

Furthermore, by tuning the relevancy model or rank profile, e-commerce sites can promote particular products towards the top of the result set. These may be excess inventory items that the retailer wants to sell as quickly as possible, or products that generate the highest margin.

Linguistic capabilities (lemmatization, synonym expansion, phrase detection, etc.) become important here in order to correct spelling mistakes, and to effectively turn a "bad" query into a "good" query, avoiding zero hit results. Detection of implicit phrases and proper names in the query protects them from further query transformation. By creating a list of product names, for example, the search provider can ensure that queries are directed to the desired pages that match the implicit product name phrase.

Relevancy tuning becomes an important tool for *OEM providers* that leverage search to offer a value-added information access layer across their own application frameworks, along with integration with third-party structured and unstructured data sources.

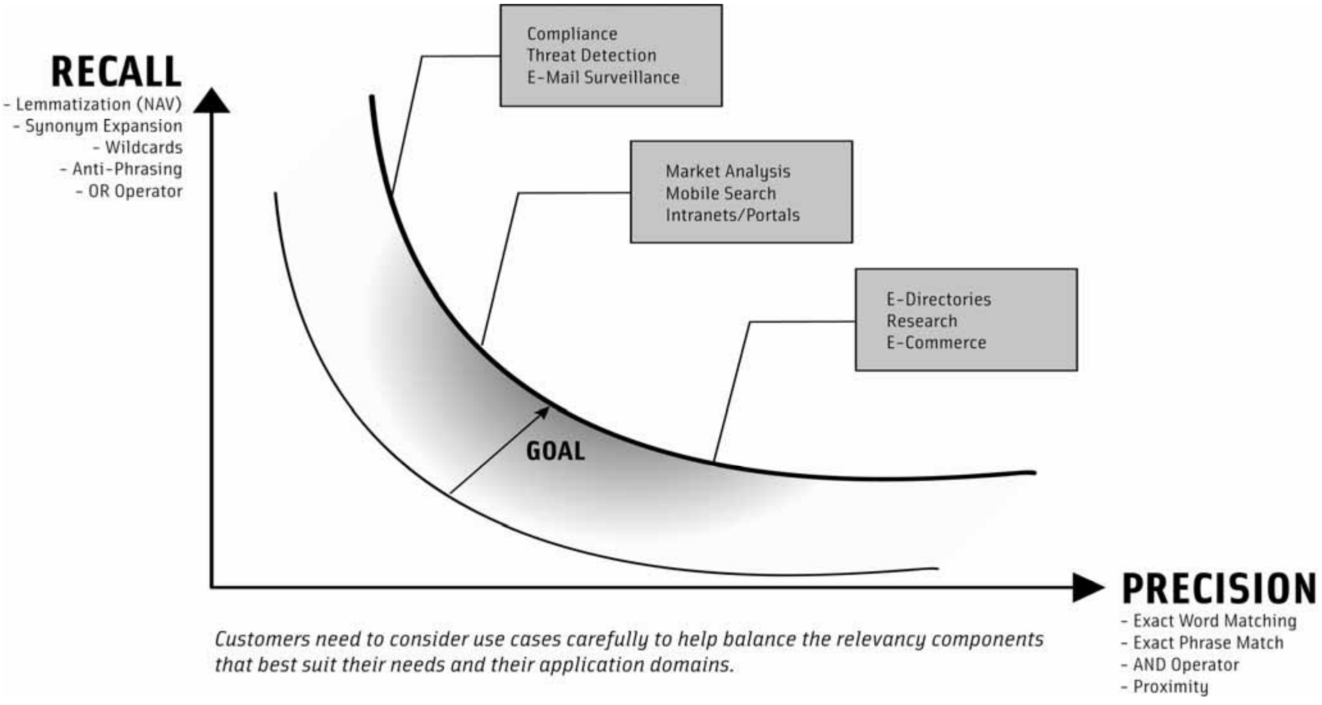
Given the wide range of implementation scenarios and user profiles, OEM providers need to be familiar with rank profile concepts so they can control the ranking and sorting of query results, linguistic tools, entity extraction, and boosting techniques to achieve the best results.

The impact of relevancy

The primary aim of applying relevancy models is to bring forward the most useful information for a particular query. In other words, relevancy enables the system to best address the user's need for information. These models provide what are sometimes referred to as a "*good-and-plenty*" result profile, which refers to the precision, accuracy, and completeness of the result set relative to a given query set. Over time, customers can tune the relevancy models to meet the needs of diverse user environments and to *shift the relevancy curve to the right*, improving both recall and precision as a result.

From the standpoint of search providers serving organiza-

Optimizing recall or precision – or both?



Customers need to consider use cases carefully to help balance the relevancy components that best suit their needs and their application domains.

tions' needs, relevancy models are often underestimated. Unfortunately, many organizations apply a one-size-fits-all methodology, which does not serve users' varied needs at all well.

In an e-commerce or e-directory setting, for instance, failure to find the right information at the right time can lead to lost sales when customers cannot locate the products and services they want. The costs are also significant for any large enterprise that depends on the skills and capabilities of its knowledge workers. In an IDC study titled "*The high cost of not finding information*", the research firm noted that poor search functionality can cost a business dearly if employees make poor business decisions based on faulty or poor information. And productivity plummets when different divisions and project teams reinvent the wheel. These tangible costs can be summarized as: 1) *the time wasted on searching*; 2) *the cost of reworking information*; and 3) *the opportunity cost to the organization*.

It is vital that content owners understand search users' profiles and their search expertise. The average user will type only one or two words into the query (and will often mis-spell at least one of them.) So content owners must combine linguistic mechanisms, advanced search, and relevancy tuning so the appropriate content can be brought forward to meet the needs of, say, a knowledge worker or an account executive.

Good enterprise search engines will place the power of relevancy in the hands of the business owner, not the IT department. The business manager or content owner defines and adapts relevancy models with respect to the user base, building in the appropriate business rules.

Customers may have to make relevancy trade-offs, such as the ingestion speed versus recall and precision, and the index size versus recall and precision.

In general, it's useful to process the content before indexing – using tools such as lemmatization – to maximize relevancy during query time. It's then helpful to store the enhanced metadata in the index that is searched. This helps reduce ingestion rates and expand the index, but it results in an unparalleled search experience – the measure against which search platforms are measured by users. One point worth noting: disk space is far less expensive than servers when considering the tradeoff between index size and ingestion rates.

Guidelines and recommendations

Since relevancy is a complex subject, it's not surprising that many mistakes are made in addressing it properly.

For example, some organizations discount relevancy as a tool to bring out the right information at the right time. Users can be quite skeptical that *machine-derived relevancy* can produce the best results for their queries. These users tend to have had experience with systems that use taxonomy and classification as a primary means of information retrieval, so they tend to fall back on what's comfortable.

At the same time, some organizations tend not to *understand the tools* available to them. Consequently, they either do not use them or they configure them improperly (many customers wrongly configure the profile of the indexed content.) And others fail to use what we'll call a *"golden set" of documents* and queries – a mechanism used as a control in relevancy tuning exercises.

In order to measure the efficacy of relevancy models, and to know how to tweak the models appropriately, search customers should understand the relationships among the body of information, the particular queries, the processing of documents, and the index and processing on the query side.

I run an e-commerce site and I have excess inventory of an LCD monitor that will soon be superseded. How can I move this inventory out quickly?

Relative query boosting allows you to promote the ranking score to ensure that a particular document is always displayed in the first N hits on the result list, provided the user searches with a specific query. For all other queries, the ranking position will not be impacted by any boost, thereby maintaining high levels of user satisfaction .

Four tips for improving relevancy

There is no escaping the need to understand *the user base* and the importance of providing different relevancy frameworks to suit different information needs and business models. Search providers cannot forget that users don't search for the fun of searching – they're looking for specific information.

The following tips are helpful for improving relevancy and search effectiveness:

Understand rank profiles – this clearly ties to understanding the user base, but it goes further. By leveraging rank profiles, organizations will be able to influence the relevancy calculations that are applied to documents. It is possible to have individual profiles configured for each of several user groups.

Influence the rank models – search providers can alter the rank calculation assigned to documents for given queries. This is particularly useful in e-commerce and paid-for-placement scenarios because they will boost the ranking of higher-paying advertisers.

Use linguistic tools – Apply lemmatization to improve precision and recall; synonym expansion to improve recall; and spell checking to prevent futile (0 hit) queries. Activate anti-phrasing to remove the "noise" from the query, such as the text of the phrase "how do I".

Test, measure and refine – use a "golden set" of well-known documents and queries to test and tune relevancy. Providers should use at least 2,000 documents and more than 50 queries.

A DOCUMENT'S RELEVANCY SHOULD BE DECIDED BY THE EXTENT TO WHICH ITS CONTENT SATISFIES THE USER'S NEED FOR INFORMATION – NOT SIMPLY ON THE BASIS OF THE WORDS COMMON TO BOTH QUERY AND DOCUMENT

By understanding relevancy, organizations gain the room to be creative with their underlying business models and to experiment in the presentation of results. They can apply *entity extraction* to unstructured data and *dynamic drill-down* to structured content, for example. Tools like these allow search users to determine what's relevant to them (price, rating, availability, etc.). They also provide a navigation experience, guiding users to the right answers in as few clicks as possible.

Of course, the final determination of relevancy is subjective, and measurements of relevancy must always include user satisfaction surveys. In the end, superlative search is possible when there are open, high performance relevancy models.

MINI CASE STUDY: Global IT company boosts sales with highly relevant search platform.

WHO: Global provider of IT hardware and services that uses its online channel for direct sales and service support	CHALLENGE: To provide customers with efficient access to the information needed to make informed purchasing decisions on over 14,000 products and solutions
SOLUTION: An intelligent and flexible search solution that provides highly relevant results by understanding the user's information needs and query patterns. The solution avoids futile queries by applying multiple rank models via multiple rank profiles and query mappings	TECHNOLOGY: Advanced index profiles and multiple rank profiles to support product or support queries. Also, advanced linguistics influence rank based on the user's spoken language and locale

FAQ

Will it be time-consuming to manage many different rank profiles for my user base?

Not necessarily. The critical step is to understand the nature of the content and the user base. Assigning different rank parameters to specific types of content will simplify this process.

How can I ensure that mis-spelled product queries are directed to the relevant page?

Custom dictionaries can be developed to align with your specific content. Query terms can be detected and corrected with alternative tokenization. For example, if the dictionary contains the term "thinkpad", a query for "think pad" will be corrected to search for "thinkpad".

When should I activate a wildcard search?

Activate wildcards only for fields that are not data-rich (that is, they lack appropriate metadata). Activating wildcards for data-rich fields can impair relevancy.

Where is the best place to use synonym expansion?

It depends on the nature of the search application. Synonym expansion can be applied at query time or when the document is processed. In order to provide flexibility and ease of management, this expansion is normally performed at query time. This also avoids the need to re-index when new synonyms are introduced.

Why does the search tool need to know what language I'm typing in? Can't it work it out by itself?

To perform synonym expansion or lemmatization at query time, it's necessary to know the language. Although automatic detection tools are used at index time, the queries are normally too short. Sometimes localization information can be used to determine this. If it does not prove useful, multiple dictionaries may be used at query time if the query is limited to a small number of languages.

What is a static rank?

It's a query-independent, fixed rank that is given to a document during document processing. For instance, the CEO's quarterly newsletter may be static-ranked as #1 in the result set for any query within a certain time frame.

How do I deal with duplicates?

A good search system will remove duplicates from the result set. Duplicates can be detected on several levels, differentiating between real duplicates and perceived duplicates. Field collapsing allows for the de-duping of results with identical value in any given result field.

Linguistics and Search

Linguistic tools, such as spellchecking of queries or grammatical normalizing of content or queries, can greatly simplify user's search experience. However, as with most other aspects of search, for optimal application of linguistics it is essential to know the users.

In the search world, linguistics is defined as the use of information about the structure and variation of languages so that users can more easily find relevant information. This is important for properly using search tools in various "natural" languages; think of the structural differences between English and Chinese. It is also important to industry-specific language usage – for instance, the English used in an American pharmaceutical company versus that used in a Hong Kong-based investment bank.

Another use of linguistic tools is to determine the intent behind keywords. For example, when a researcher enters "When was D-Day?" she is looking for information that resembles a date. Another example: someone who is searching an e-commerce site for "MP3 players" would also be interested in hits that matched "MP3 player" or "iPod". If the site shows only results for the keywords "MP3" and "players", it may easily miss out on a sale. The first example improves search precision by combining semantic analysis (understanding and interpreting the search terms) with *entity extraction* (isolating known linguistic constructs). The second provides better recall by using *lemmatization* (associating variations of word forms) and synonym expansion. However, these tools may not always improve the user experience: a stock-market trader searching for the stock quote "BAB" would just get annoyed if he was asked "Did you mean BAD?" or swamped by hits about “babies” rather than simply being given information on the British Airways share price.

Clearly, a grasp of the linguistic features of advanced search engines, coupled with knowledge about the data and users of the search service, can greatly improve precision and recall and thus yield better business results. This chapter will lay out the main options for those building a search service and bring forward the main questions that must be answered soon.

The many aspects of linguistic analysis

When designing the linguistic elements of an advanced search engine installation, the designers must consider different levels of sophistication of features in terms of their impact on the user experience, on the administration of the system, and on the complexity of the user interface.

To begin with, there are *unilingual and multilingual installations*, i.e. those that treat all documents as being either of the same language or of no language, and those that treat them separately. Within the multilingual category there are different degrees to which these languages can be treated separately, and there are systems that combine very different languages - for example, European languages with Chinese, Japanese, or Korean (CJK), all of which have special requirements (there is no symbol used for word delimitation so advanced *tokenization algorithms* must also be used).

5% of my users and content are non-English . What should I do differently for them?

You will need to think carefully about the language-specific features of your search function. If people search only for content in their own language, or there is wide variation in the language types used (English, Polish, and Chinese for example), then it will help to have users specify their language in the query interface. Where there are common linguistic roots - on an e-commerce site featuring English and Dutch content, say - it may be easier to handle everything in the most common language – in this case, English.

Crucially, given the differences between languages, for some language-specific features to work, it’s necessary to distinguish between them before each document enters the index, either manually or by using *automatic detection mechanisms*. Once that has been done, the document index can be augmented by extracting or adding information. The most common augmentation is *lemmatization* - the expansion of all known terms in the document to their inflectional forms or their base form. Lemmatization enables searches to match documents with similar meanings but different keywords.

The document index can also be augmented by using *synonyms*, which correlate words in a much broader sense and allow for industry-specific terminology and acronyms. And it can be streamlined by the removal of *stop words* (words which are frequent but have little meaning to the search), which also aids recall. For example, after stop-word removal, a query for “President of United States” would also match documents mentioning “The President of the United States”.

For languages that have diacritic characters (like Romance or Scandinavian languages) it often makes sense to consider character normalization (the mapping of diacritic characters to standard characters like {é,è,ê} à e) in order to increase recall. That said, in some search engines it is possible to preserve and search against diacritics.

More advanced interpretation of language can be done through entity extraction (the spotting through the combined use of dictionaries and *syntactical patterns* of certain entities such as people, places, product codes, prices, etc.), parts of speech detection, and *sentiment analysis* (the evaluation of the text’s sentiment - typically positive or negative - based on the usage of language). Categorization (the identification of documents as being part of an ontology or taxonomy from rule-based or conceptual category definitions) and the *unsupervised clustering* of documents (grouping related documents on the basis of their content without referring to a taxonomy) also leverage document semantic and conceptual interpretation.

It’s important to note that some of the techniques described above can be applied at index time or at query time by modifying the user’s search. The choice of where best to integrate

My software features a search engine and my customers worry about the installation footprint. What can I do with linguistics to minimize my hardware usage?

Disk usage will vary with the amount of indexing-time expansion (entity extraction, synonyms, etc.) but it may reduce the QPS (queries per second) or functionality available to end users. There are two classic trade-offs to consider: ingestion rate vs. query speed and recall vs. index size.

lemmatization or synonym expansion, for example, usually comes down to considerations of performance, practicality, and flexibility.

Linguistic features can also be focused on the query side, modifying the search and turning a “bad” search into a “good” one.

An obvious example: *spelling correction* (either automatic or “did you mean…?”). Other features aim to improve queries too. *Phrasing* (the recognition and grouping of an idiom such as “home run”) and anti-phrasing (identifying word sequences in queries that are irrelevant to the search) are good examples. And the tuning of the relevancy model can be used to increase or decrease the importance of statistical linguistic analysis (i.e. *tf-idf*, which uses the relative frequency of words in a corpus of data to determine their importance).

Phonetic search can be applied to structured searches such as name searches. In this case, it isn’t the morphological variants or synonyms that are identified; it’s the words that are pronounced similarly. For example, phonetic search will detect all possible variants of “Muamar Gadaffi” (Muammar Al Ghaddafi; Muammar Al Qaddafi; Muammar Al Qaddafi; Muammar El Qaddafi; Muammar Gadaffi; Muammar Gadafy, etc.)

Linguistic methods are also used when performing *speech-to-text.* By analysing the phonetic models of speech, along with knowledge of the nature and distribution of words within common languages, modern transcription software is able to create an accurate dictation of the audio track of multimedia

5 things you should know about linguistics

1. Better use of linguistics will improve precision and recall
2. Industry and user knowledge are needed to optimize multilingual systems
3. Linguistic choices can impact hardware and performance
4. Some sites should favor language independence
5. Bad queries can be turned into good queries with the proper linguistic tools

files. This in turn enables the audio and video files to be searchable alongside other more typical formats such as pdfs or Word documents.

Different industries, different solutions

With such an extensive list of linguistic options, the key to the best user experience in terms of precision and recall is to work out what permutation of them best suits the business needs of the search application.

For instance, in the case of an *e-commerce* application, the business driver is to make sure the user finds the product he is willing to purchase. It's important to include a dedicated thesaurus specific to the types of products being sold (i.e. "MP3 player" -> "iPod"). Automatic spelling correction is another invaluable feature to ward off user frustration: a user who accidentally types "floghts to London" should still be taken to responses to a search for "flights to London" rather than having to click on a "did you mean…?" dialog box (whether the application automatically corrects or prompts is an interface design decision). And lemmatization and spell-checking must be used with great care. Someone looking for "Golf GTIs" should not be steered towards "golfing gifts". Ideally, for e-commerce, custom spelling dictionaries should be maintained and few or no lemmatization and stop-words should be used.

At the opposite end of the spectrum, extensive lemmatization and stop-wording are essential when performing *knowledge discovery* over verbose unstructured data where meaning is more important than simple keyword matching. Entity extraction will also play a much bigger part to ease navigation through larger amounts of data.

Language-specific functions such as entity extraction and lemmatization require knowing the language of documents and queries to be performed optimally. Therefore OEM integrations of a search application require caution about configurations that assume this knowledge, since the embedding application will be used in a variety of contexts without the opportunity for tuning.

What's the effect of linguistics?

Linguistics aims to leverage the meaning of documents or words outside of the keyword paradigm. It transforms queries – a valuable feature since users type only one or two (often misspelt) words on average.

To determine the success of a search application from a linguistics point of view, the most important metric is *the number of empty result sets returned*. If the number is too high, features such as lemmatization, synonyms, and spelling dictionaries should be investigated or refined. However, this requires that application manager take a proactive interest in the user's search experience and work closely with industry experts and librarians to leverage their knowledge of the content. From a search user's point of view, the usefulness of such search assistants will depend on their level of expertise in the field as well as their familiarity with search.

A *novice user* or one in an unfamiliar domain will favour an interface with the least number of options to allow natural

I run a mass market classifieds Website. How can you help me make my interface simpler?

Let's say you want people to search for "Two bed flat in midtown Manhattan with two bathrooms" and they find "Midtown - 2 bed apt. 1 en-suite". They key is to get the index to do the work instead of your users having to fill out forms. Bespoke thesauri, entity extraction etc. Also you're running a consumer service so don't forget to design the system with high query volumes and do as much processing index-side as possible.

language searching. A *librarian or expert* will want to be able to select for themselves whether to use the synonym dictionary, and they will use the date field if searching for a date rather than relying on the natural language processing algorithm to translate "When was…".

This comes back to the key point: *an understanding of users and their needs is essential from the content owner's point of view.* This will determine what types of entities should be marked up (the "When was D-Day" example relies of the system having marked up all date type content in situ so dates near the word "D-Day" can be identified) and whether to do lemmatization by expansion or reduction.

Lemmatization by expansion will give the best query performance and is required when the language of the searcher is not known (for example, when there is one corporate search service in a multinational enterprise). However, this increases the index size, which is costly for certain languages such as Finnish or Hungarian.

On the other hand, *lemmatization by reduction* or stemming modifies both the index and query terms so the language of the search must be known to avoid spurious and ambiguous results. This approach is recommended though if storage space is an issue.

Synonym expansion at query time will affect query performance, but it will allow the search system's administrators to modify their thesauri when they want without the need to re-index – a big difference from doing the expansion before indexing.

Guidelines and recommendations

Many *common mistakes* are made when configuring the linguistics of a search engine.

Firstly, there is the failure to use the linguistics features of good search tools to their full potential, especially when the system has a broad and varied user base.

Another common pitfall is attempting to use these features without correctly identifying the language of the documents, and more importantly, the user's first language. It's also necessary to gauge what type of service you are trying to provide: a multilingual search, but "one size fits all", most likely optimized for the most common language; or a targeted system where content is segregated and tweaked according to its different sources.

Yet another area to watch out for is the *mismatch of interacting components*: for example, when using different processing on the index and query side, or when there are conflicting synonym and lemmatization or stemming methods. It is also important to understand the trade-offs between hardware (disk, processor), index and query performance, and functionality.

The fundamental steps to improve search

From a language perspective, a poor search installation implies a failure to understand the nature of the user base or the importance of localizing to the industry or country when providing search over unstructured content. To help head off the consequences of such failures, content owners and search providers must ask themselves some simple questions:

» What types of queries do my users perform (natural language, keyword, field-based)?

» What languages are my documents in? What is their nature (structured, unstructured)?

» What languages do my users speak? And do I have enough multilingual users to justify optimizing for them separately?

» Is hardware a primary consideration over functionality?

» Can I leverage my industry-specific knowledge to improve synonym and spelling and entity dictionaries?

Answering such questions and mapping the appropriate linguistic tools to the subsequent platform profile will help drive the success of search applications in fields such as e-commerce and knowledge discovery. Users will feel that the search engine is working for them; they will feel they've been "listened to" and feel confident that more relevant information is delivered to them. Content providers will see *improved visibility for and consumption* of their information, giving them greater productivity and increased e-business sales.

MINI CASE STUDY: Newspaper network taps advanced linguistic search features to drive growth in classified ads.

WHO: Major American newspaper network with growing Web presence.	CHALLENGE: Make Web site more profitable by generating revenue from classifieds.
SOLUTION: create a smart interface to allow highly structured searches and alerts for users who know what they're looking for. At the same time, allow natural language search over text with highly specialised acronyms and abbreviations.	TECHNOLOGY: Entity extraction and customized synonyms to power navigators for advanced search, and language processing for query-side novice buyers.

FAQ

What is an ontology?

A specific set of knowledge related to a given domain – for example, pharmaceuticals.

Do you need to worry about linguistics with structured data?

Usually, most linguistics features can be turned off. However, if your users want to do natural language queries, you will need some query-side processing.

What is meant by language morphology?

The rules and semantics of the formation and structure of permissible words in a given language.

My data is in different encodings. How will multilingual search work?

Everything should be normalised to UTF-8 but with an important caveat. When importing dictionaries, it's valuable to normalize accented characters for simplicity.

Why does the search tool need to know what language I'm typing in? Can't it work it out by itself?

To perform synonym expansion or lemmatization at query time, it's necessary to know the language. Although automatic detection tools are used at index time, the queries are normally too short. Sometimes localization information can be used to determine this. If it does not prove useful, multiple dictionaries may be used at query time if the query is limited to a small number of languages.

What is a bad query?

A query that is too short and ambiguous for simple keyword matching to bring back relevant results.

Query and Result Processing

Good search engines do most of the hard work behind the scenes, and thus simplify the user's search experience. For example, vague or misspelled queries can be refined in query processing, and results can be filtered, merged and post-processed for intuitive navigation in result processing.

Searching is becoming increasingly complex. Queries now include single words, phrases and questions, and whole passages and documents. In some cases, the right result can be a single document or answer. In most cases, the correct result is an array of relevant information, strengthened by precise navigation to related information and topics that can help the searcher discover other insightful results or get a more complete answer.

To deliver consistently superior results, you must understand the exact intent of the query. You must also know what information is available, how it relates to the query, and where it is located. Accomplishing these goals requires a mixture of technologies, each with complementary strengths.

In search, true success comes from understanding what the user is asking from their query. Some user queries are simply stated, while others are stated in a Boolean format (“apples AND oranges OR bananas”), or presented as whole paragraphs, passages, or documents with a request to “find similar” information. So the search platform must have a range of tools in order to accurately understand what is being asked.

Two applicable technologies are Natural Language Processing (NLP) and linguistic analysis. NLP interprets queries posed as questions, phrases, etc., in part by identifying and stripping out terms that don’t contribute to the relevance of the results. Linguistic tools include capabilities that circumvent word-sense ambiguity – for example, distinguishing between the color orange and the citrus fruit. Search applications use these technologies to analyze human language, identify a searcher's intent, and return the most relevant results.

Enhancing search with QR processing

The challenge with information retrieval revolves around two basic problems: 1) *getting a good query from search users* with the aim of helping them craft better questions, and 2) *presenting "easy-to-judge" results* to minimize what the user has to read through. For example, are a title and the first few sentences of an article a satisfactory result?

Basic search cannot always figure out which words are most important. In the query "How do we replace our Social Security cards?" is "social" more important than "security?"

My company has an integrated search platform that connects multiple content sources from files systems, ERP data, and corporate applications. How can I ensure that employees can access only the information they're permitted to access?

To begin with, you should look to use the underlying security principles or models for each employee, role, application, etc. and make this data available to the search application. You can also configure multiple rank profiles and relevancy models that will surface only the permitted content for the respective employee roles or groups.

Phrases are not always obvious; do "social" and "security" form a phrase? Boolean formatted queries are not always clear; is it "social" AND "security" or "social" OR "security?" And the query could have an "unstated" question – the user may just want everything about "social security."

Turning every search request into a well-understood query requires analysis of ambiguous types of queries as well as alternate complementary analysis capabilities. Enterprise search engines should analyze such queries along these four dimensions:

» *Orthographic* -- checking for typos, official variants (e.g., German spelling), etc.

» *Morphologic* -- including all forms of a given word via linguistic normalization (lemmatization).

» *Syntactic* -- entity or phrase extraction, anti-phrasing, removing word-sense ambiguity (orange color vs. fruit), etc.

» *Semantic* -- applying a combination of general and specific thesauri and ontologies, automatic phrasing, etc., to understand the intention of the query.

In order to effectively analyze the search query and deliver appropriate results, search applications rely on a key component known as the '"query and result processing" engine (see diagram later). Fundamentally, QR processing is the application of algorithms to the original query or to the raw results returned by the search engine. In general, queries from the user come into the query processing and transformation subsystem. This framework takes the original query, analyzes it, transforms it with, for example, corrections of spelling mistakes (“Nisan Macra” will be corrected to “Nissan Micra”), and then sends the query to the search engine.

The node in the search matrix that receives the query performs its retrieval operation and returns its results to the results-processing subsystem. The raw results are passed to the results-processing subsystem (which performs duplicate removal), results merging (from different search nodes), sorting, rank ordering, etc. All results are then sent to the search user. In general, QR processing is measured in terms of *QPS* (queries per second) and by the *perceived relevancy* of the results. These measurements are affected mostly by the following: the hardware used in the search matrix, the limits imposed by licensing (e.g., how many search nodes), the linguistics features available in the index, the query’s complexity, and the query-specific features invoked.

Use of the QR processing stage depends on the complexity of your content, your business model, and your search goals. The depth and quality of processing that the search platform performs on content directly affects the speed and quality of the query results.

When leveraging QR processing, search customers will typically start with the functionality that comes standard with the search application. Over time, they will introduce more complex processing capabilities to deal with the mix of different data sources. Linguistic tools such as spell checking, lemmatization, and query re-writes can then help to improve the relevancy of the results.

Hidden within the search application is the relevancy model that can be tuned to meet the search needs of particular users or aligned to the business environment. (This is discussed in more depth in the “Relevancy” chapter.) In this environment, simple and advanced queries would be supported, ranging from simple keyword searches to Boolean operators.

Depending on the profile of the user, organizations may wish to use the appropriate security and filtering features to ensure

5 things you should know about query processing

1. QR processing, or query and result processing, is the application of algorithms to the original query and/or to the raw results returned by the search engine.

2. The goal is to analyze human language to identify the context of the searcher’s intent in order to return the most relevant set of results.

3. Loading the system beyond its query rate capacity will create a backlog of queries.

4. The query rate capacity is limited mainly by the number of search rows, with multiple search rows providing a linear scaling of QPS.

5. OEMs should leverage the query API of the search system rather than relying on HTTP.

that the user gets access only to the information that he or she is permitted to view. For example, if employees in research and development search the company intranet for "remuneration packages," they would likely get back only documents that describe HR policies in general, and not be able to access information on employees' salaries.

Advanced use of QR processing capabilities would see organizations employing custom query transformations, which would automatically modify the query in order to improve recall or precision. An example would be geo-encoded searches, where the user is looking for results within a preferred distance of a particular location. Another example: automatic acronym transformation, where "IBM" would also return results for "International Business Machines" and I.B.M.

My CMS application features a search engine, but it currently displays lists of results. How can I improve the usability without degrading performance during query/result processing?

Consider using the analytical capabilities of the search solution – entity extraction on unstructured content and/or dynamic drill-down for structured data. This will allow you to provide navigational search. To protect performance, ensure that the search platform carries out as much of the processing as possible, and try to minimize post-processing of results. Increasing hardware (if appropriate) can offset the additional load on the system.

It is also possible to request that a query term string is transformed in case there are no returned hits from the original query. In this case, the modified query term string is returned so that a result page may inform the user of the performed transformation.

On the results-processing side, advanced scenarios would try to improve the usability of the search interface, especially if thousands of results are returned. In this case, an organization may opt to cluster results or analyze or sort them. For example, the query "BMW dealership" on an Internet Yellow Pages site might be set up to return the categories and number of hits – for instance, "BMW dealers (12), BMW garages (42), BMW repair shops (17)," etc.

In an enterprise environment, organizations may blend results from multiple systems (this is known as a federated search) so that users can get results from corporate file systems, or from the intranet, extranet, or Internet. In this case, the user interface becomes critical to simplifying the viewing and consumption of information.

Alerting functionality is becoming popular with seasoned search users who prefer to have information pushed to them rather than searching for it each time. Here, multiple filter conditions (or triggers) are matched to an incoming stream of fresh data in the form of events such as news articles, stock quotes or other documents processed by the search application. Alerts or filtered content streams are provided in real-time and converted to appropriate applications or end-user devices via XML.

Different industries, different solutions

It is apparent that many facets of an enterprise search system need reasonable consideration before they are used in business-critical environments. So what does query and results processing mean for your business?

An investment bank will have vastly different *knowledge discovery* or search needs compared to an oil and gas manufacturer. A bank needs tightly integrated security models to ensure that employees on either side of "Chinese Wall" (equity research teams and corporate finance teams, for instance) are granted access to only the information they have permission to receive. Oil and gas manufacturers also need to consider security, but not on the same level as the bank.

In an *e-commerce* environment, the business model focuses on sales volumes, so e-commerce providers have to pay particular attention to the processing and presentation of results. If a search was for "PDA", for example, linguistic processing would be performed ("PDA" is translated as "Personal Digital Assistant") before the search application returns results. The results would be categorized by PDA brand, category, price, type, color, availability, etc., to allow simple navigation. Information/entity extraction would be used to ensure that consumers can complete a purchase in as few clicks as possible. High QPS and performance will have to be considered in this environment. If consumers have to wait more than five to 10 seconds for a results page, they might defect to a competitor.

> **Linguistic optimization tools can be applied both on the index side and at query time. The choice of where to use these tools comes down to performance, practicality, and flexibility considerations.**

OEM integrations of search need to consider how much linguistic processing to perform on queries, and the level of results processing necessary for different user scenarios. OEM applications should utilize the standard functionality of the search system and interface directly with it, to leverage the administration and reporting tools that will allow configuration and tuning as needed.

Understanding the impact of QR processing

Different search contexts call for different response profiles, and different enterprise objectives dictate different response parameters. However, most search engines use a fixed-ranking relevancy model, which is acceptable only when the search solution is used in the context it was designed for. It's far better – consistently superior – to integrate linguistic and result-processing capabilities into a holistic and adaptive approach.

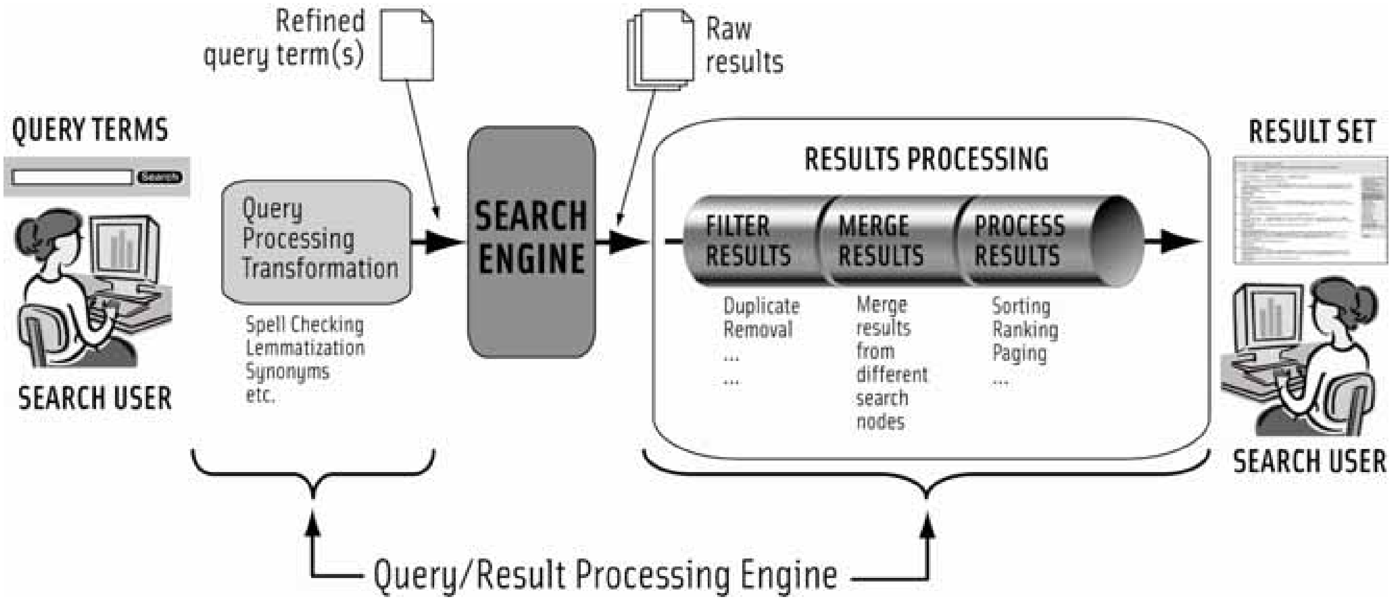
The holistic aspect means applying linguistic analysis across the board – documents, queries, results, and navigation – to maximize the contribution that such technology makes. The adaptive aspect means using the right components, leveraging industry terminologies, and tuning the ranking model to match the type of search application – from broad uses like a general information portal to specific solutions like shopping sites.

As noted on the previous page, the main objectives of QR processing are to turn a potentially bad query into a good query, and to present the best results in the form that's most helpful to the user. The search provider can develop modules for specific query analysis and results processing in order to offer a personalized user experience. The front-end search

can be configured to select particular rank profiles and present the results in certain ways if the user group is known. For example, a medical application might process and display results differently for doctors compared to the results offered to nurses or to medical students.

most efficient to offload as much of the processing to the search platform as possible rather than trying to post-process all of the results. This becomes particularly important if you are federating results from other content sources or applications.

Elements of query and results processing



It's interesting and worthwhile to note that the search consumer may not always be a person; it may be another IT application. In that case, the search application owner will manage and maintain the query and the results processing stages, with specific frameworks for post-processing that allow the results to be presented in formats that are consumable by other IT systems.

It is rare to get a perfect search environment from the outset, so business managers will have to monitor the functional specifications, and over time refine the QR processing stage (using control and administration tools) to meet the evolving needs of the user base.

Trade-offs associated with QR processing include the points at which processing is performed – in the core of the search engine, in the QR stage, or on the client side. Typically, it is

The filtering of results, such as removing duplicates or limiting the results, can be carried out per the terms of the query (using the search application's query language and constraints), or it can be done by the client-side application. For example, searching a sports category may limit only the documents written in English or Spanish.

To achieve the best results, in terms of relevancy and speed, it is worth letting the search application do the work to return as few highly relevant results as possible.

Guidelines and recommendations

Generally, the metrics used to evaluate QR processing include QPS (queries per second), the number of results returned per query, and the relevancy of these results as judged by real users. The query load performance is measured as the maximum QPS number that the system can process with acceptable response times.

The QR processing overhead and query response time need to be quantified – both perceived and actual volumes. The algorithms applied at query and response time need to be measured for speed and efficiency, especially relative to the time that the core engine spends in search.

Among the mistakes most commonly made when organizations develop search applications are the failure to understand the query processing stage, and the attempt to perform too many calculations across a large result set during result processing. Loading the system beyond its query rate capacity will generate a backlog of queries, which creates higher query latencies and possibly disruption of service. Depending on where the system bottleneck is, the backlog may cause timeouts and retransmits. This in turn generates even higher loads on the search service, markedly degrading service.

The query rate capacity is limited mainly by the number of search rows, with multiple search rows providing a near-linear scaling of QPS within the search engine. It is also constrained by the number of servers used for query and results processing, which becomes especially important when using result-processing features. It is recommended that you plan for increasing query loads and upgrade the system accordingly.

It is crucial to understand the performance of the query API on the client side. In some situations, the query API can perform a substantial amount of parsing and processing – for example, it can include result clustering and navigators with multiple buckets. Search users should understand this and design the client API appropriately.

Finally, the feature set has a strong impact on the effective QPS. Result-side features such as dynamic duplication removal, clustering of results, and result-side navigators will add

I need very high QPS on my e-commerce site. How can I achieve this?

Query rate capacity is limited mainly by the number of search rows – multiple search rows will provide linear scaling of QPS. The number of query and result servers is important when using result processing features. You should limit the result-side features (dynamic duplicate removal, result clustering, etc.) to optimize for the number of QR servers needed.

substantially to the load on the query/result servers. Increasing the level of hardware can offset the load on the system. Other features like deep navigation and full wildcard support will also add load to the search nodes.

The fundamental steps for improving search

It's best to take a phased approach when developing your search application. This will allow you to identify what works well and to isolate areas for improvement. It is useful to leverage the standard query processing options such as synonym expansion and automatic query rewriting.

With QR processing, it is advisable to do as much as you can in the core of the search application. Increasing the number of search rows allows you to enhance the speed of processing and boost scalability in a near-linear fashion. Ideally, search providers should offload as much of the processing to the search platform as possible rather than trying to post-process all of the results, which can create latency issues. Here, the system should use deep navigators and avoid result-side (shallow) navigators; that way, the search tools will be less likely to return more results than is necessary per query.

When returning results, it is useful to understand the result volume, since most search applications can return the top N results or the entire result set. This is a trade-off between speed and satisfying the need for all information. The more data returned, the more time it takes to stream it back to the search client.

It is vital to understand the impact of relevancy. Most users and applications will need only a small subset of the entire result set if the relevancy model is adapted to their needs. Organizations are urged to understand the impact and cost of features that operate on an extended result set – features such as results clustering.

In an ideal environment, the best performance and consistency will come from having all content indexed by the search application. This isn't always possible, however, so users should be aware of the impacts of federated or blended searches such as mixed relevancy or throughput.

From an OEM perspective, it is advisable to leverage the query API of the search system rather than relying on HTTP. Generally, the query APIs and connectors will provide a rich and robust wrapper around the underlying HTTP interface to the search engine. This makes it easier to work with the search application and provides additional capabilities such as error checking and an administration interface for reporting.

MINI CASE STUDY: Enterprise storage vendor uses search as a competitive differentiator.

WHO: Worldwide provider of enterprise storage solutions.	CHALLENGE: To provide value-added features and functions to storage solutions using advanced search capabilities.
SOLUTION: Provision of unique "charge-back" model for storage managers in large organizations. Ability to handle large content volumes (50 million-plus documents) and provide high-availability configuration, integrated administration, and security.	TECHNOLOGY: Advanced query/results processing features, advanced administrative APIs, and integration with third-party data and custom code.

FAQ

What's a rank profile?

The relevancy of a document with respect to a query is represented by a ranking value. A rank profile concept enables full control of the relative weight of each component for a given query (e.g., How important is the title relative to the body of the article?). This enables individual relevance tuning of different query applications.

Can I return information that resides outside of the search application?

Federating or blending results is possible, but you have to consider the consistency of the result relevance and throughput. These issues can be solved with additional hardware and by tuning of the relevancy model and the associated rank profiles.

What is query transformation?

It refers to the analysis and subsequent rewriting of a query – typically linguistic transformations such as lemmatization and spell checking. If need be, you can also plug in custom query transformation stages.

What is results transformation?

This is the algorithmic processing of search results. It includes result-set reordering (e.g., duplicate removal), adding navigation information (e.g., clustering/drill-down), and result content conversion or reformatting.

Can I pass results from the search system to a third-party application?

Search systems should be able to return results in the format that you require – text and/or XML. You have to understand the downstream consumer's needs so that your application returns information in a suitable form.

Is it possible to customize the QR processing stage?

It is possible to augment and enrich the frameworks associated with the QR processing stage for your application needs. Typically, this will require custom modifications. It's advisable to seek expert advice to design and build the right solution in the shortest time.

Content Refinement

Content is the external data, from many different structured and unstructured sources, that is fed into a search engine. Before being stored, the content is refined for optimal retrieval.

Today's public search applications have taught users to find information in the shortest time possible. They have specific information requirements that must be fulfilled. However, they aren't perfect. Users often have to launch multiple queries before they find the information they want.

One reason for this is that queries are typically one or two terms in length and fairly generic.

Another reason is substandard content quality – *low quality content and poor queries will result in bad hits*. If the content quality improves, or the content and the queries improve, users will get better results. Generally, search applications have little or no control of the information being fed into the search system, so query results can be poor.

Content owners and business managers can control the quality of content before it is pushed to the search application - although many aren't yet aware that this is possible.

This chapter highlights the business need and impact of *content preparation*, the interconnected topics of *content aggregation* (getting information into the search system), and *document processing* (analysis, transformation, and enrichment of original content for the purpose of indexing).

5 things you should know about content refinement

1. Clean and normalize content to achieve the best possible relevancy during query time.

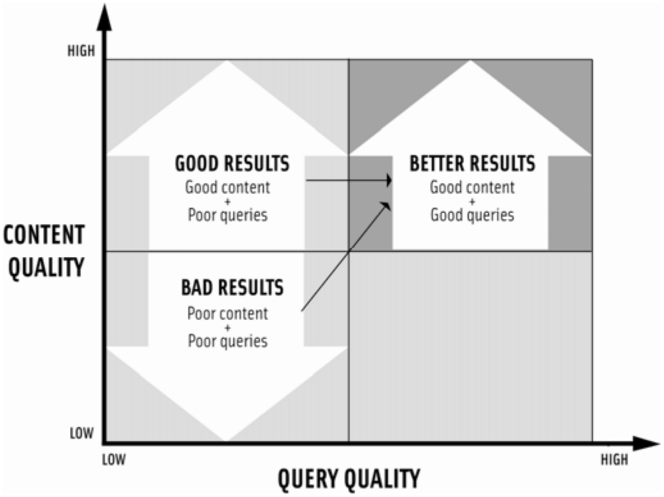
2. Normalize content – ideally data (especially structured data) should be consistent and without duplication.

3. Appreciate that ingestion of content will be affected by the amount and number of different types of data, in addition to the latency of the source systems.

4. Optimize document processing – remove all unnecessary document processing components and choose the right processor for the content type and task at hand.

5. Marry content with the appropriate document processing – language detection, synonyms, spell checking, lemmatization, taxonomy classification, custom plug-ins, etc.

For good results, first improve content



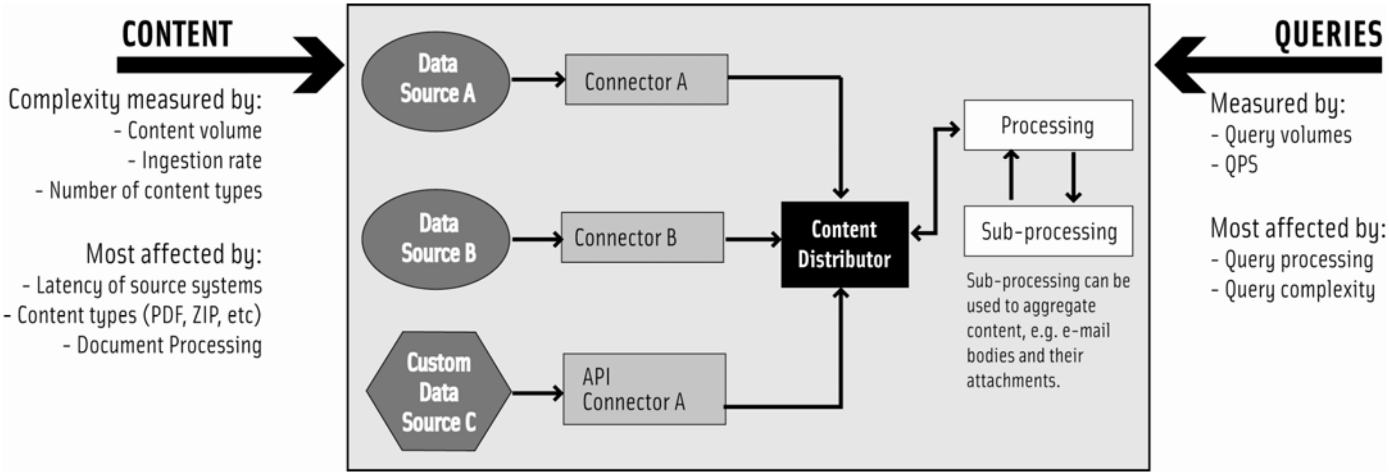
Content preparation

The old adage "*garbage in, garbage out*" springs to mind when considering content quality. Content preparation means selecting the right content, appropriately transforming and tagging it, cleansing or normalizing (regular and consistent, appropriate spelling or style) the content, and reducing the complexity of disparate data types. Information sources may include: *Web/intranet* (HTML, XML, multimedia); *file system* and *content management systems* (Doc, XLS, PDF, text, XML, CAD, etc.); *e-mail* (e-mail text, attachments); and *databases* (structured records).

This data will comprise free and semi-structured text, structured data (*XML*), binary files, and highly structured text and numeric data in the case of databases. Content preparation on each can vary from simply adding meta tags to deep cleansing of content.

Typical search application use will have limited content preparation. It will include multiple data sources and types, Web crawls (with no special content preparation), intranet crawls, file system content, CMS data with standard metadata applied to all documents, and pushed or pulled access to database content that has been cleansed.

Content Aggregation and Processing Diagram



Organizations may leverage some automated tools to assist content preparation as a first step.

Advanced organizations will look to do it all, but will have a *consistent metadata model* applied to all data, including Web content. They will also add specific keywords or industry terms to documents to improve relevancy and search. Content preparation will be performed by editors (actual staff) or by automated workbenches to correctly tag up information.

Organizations must understand the benefits and tradeoffs of using prepared content versus the raw form, and the benefits of using people over automated tools. People will be more accurate in the initial stages, but over time will be less consistent than an automated system, and will take much longer. There will be a cost overhead to factor into search estimations. Organizations will delegate a certain amount of the tagging to the content owners to enhance the "correctness" of content.

As part of the preparation, organizations must *close the loop of content* and monitor the effectiveness of search and navigation, and manipulate it to align with overall business objectives. Understanding the *ratio of content preparation to information* usage will help determine where to focus efforts. Over time, content preparation can enhance the value of future searches dramatically.

Best practices in content preparation include:

» Planning ahead - deciding which content needs to be prepared, by whom and at what quality level. You need to factor staff-driven operations into your resources, work and time estimates.

» Aiming to increase relevancy – people use a search platform to find the information they need, when they need it. Focus efforts on increasing the relevancy of the results returned.

» Normalizing content – ideally data (especially struc-

Our IT group developed document processing-type rules and code from a previous search solution. Now we have changed providers. Can we reuse the solution?

If a document processing code already exists – in Java or C++, for example – you can use it by writing a stage in the document processor to call out to it. Don't waste time rewriting code. Although document processing stages need to be written in a scripting language, you can leverage this existing IP as scripting language with the ability to invoke other languages.

We want to integrate and expand our current e-directory offering to include multiple document types from multiple sources. What should I be aware of?

Aggregating content at ingestion time requires carefully correlating the source documents. Aggregating results from multiple sources requires relevancy tuning, benchmarking, and index reconstruction, all of which are time-consuming. It's best to take a phased approach and understand the impact that additional sources will have.

tured) should be consistent and without duplication.

» Logically partitioning multi-lingual and localized content – isolating documents on a per-site or language basis.

» Striving to normalize acronyms – they can be easily expanded in the search system (i.e. IBM → I.B.M → International Business Machines).

» Considering directed search by preparing content to provide multiple navigation points – product line, model family and price are all complimentary navigation facets against the same data.

» Automating where possible, since information is produced and consumed at incredible rates. Use automated preparation tools to save time and reduce error rates.

Common mistakes include expecting good results from bad data, failing to consider how to process the content for highest utilization, and ignoring the potential for error introduction by automatic data preparation. Not all systems are perfect 100% of the time.

Organizations are urged to take a *step-by-step approach*. Do not try to prepare all content types at once; learn what works and then introduce no more than one or two content types per review cycle.

The remainder of this chapter will discuss content aggregation and document processing.

Enhancing search with content aggregation

Content aggregation is the *bringing together of content from multiple sources* for the purposes of later retrieval. It is used to consolidate search results into a comprehensive whole.

Federation of search is also important within content aggregation. Benefits of federated search include the potential for more complete result sets, and no need for an increased index size or for the associated hardware. Organizations must balance the increased flexibility with the tradeoffs of such an approach. Over-simplifying the search experience does not add value.

Content is available to the search and filter/alert engine via the content API which acts as a broker of information. It performs the tasks of pulling content from the data source (database, CMS application, etc.) during scheduled calling requests and also handles the pushing the content into the search engine.

The *content pull* approach leverages data connectors to retrieve the information via standard APIs or interfaces. This is the core technology of most search solutions, and it includes retrieval of Internet-based information, database information, or file server-based documents. The data connectors do not require integration programming towards the target data repositories although in some cases they may not provide the required real-time performance. In these cases API integration may be preferred.

The *content push* approach requires that data repositories, applications or messaging middleware send the data directly to the search application via the content APIs. This omits the latency of crawling but it requires a closer relationship between the content application and search engine.

> "I was born not knowing and have had only a little time to change that here and there."

A traditional search approach typically implies long latency from the time the data is modified until the modification is reflected in the searchable index. This means that the search engine does not handle dynamic data and may not be sufficient for processing real-time information.

Some enterprise search solutions remove this limitation by scheduling frequent updates to ensure that the *information is made searchable in short timeframes*. The system takes this functionality further by integrating the real-time filter engine that matches information against pre-defined queries as it becomes available.

Typically, organizations will deal with multiple content sources, their structured and unstructured data will be *semantically related* through the appending of correlation IDs for grouping, and they will need to ingest and sub-process data as a single unit (e-mails and attachments). An advanced scenario will potentially display results based on *grouping of content* – as in the case of an e-directory displaying a mixture of Web content (description, name, etc.) and database content (for example, opening hours).

Organizations need to appreciate that ingestion of content will be impacted by the *amount and number of different types of data* in addition to the latency of the source systems. Different types of content (doc, pdf, zip, etc.) will process within different timeframes. The *complexity of the document processing* (the numbers of individual stages and their roles) will impact the speed at which content can be ingested. External factors such as network performance, repository speed and crawling/spidering windows will all have an impact on ingestion speeds.

Enhancing search with document processing

Document processing is the *analysis*, *conversion*, *transformation,* and *enrichment* of original content for indexing and subsequent retrieval.

The document processing component of a search application is shown below. Content flows in from the left of this schematic, with content ingestion rate measured by documents per second per server and by the total ingestion volume (number of documents handled). Document ingestion rates are affected by hardware capacity (the number of nodes, the RAM on each node, disk capacity and latency, CPU usage, I/O wait time, etc.), the amount of work the performed, and external lookups, where a particular stage may make database lookups or calls out to a Web service.

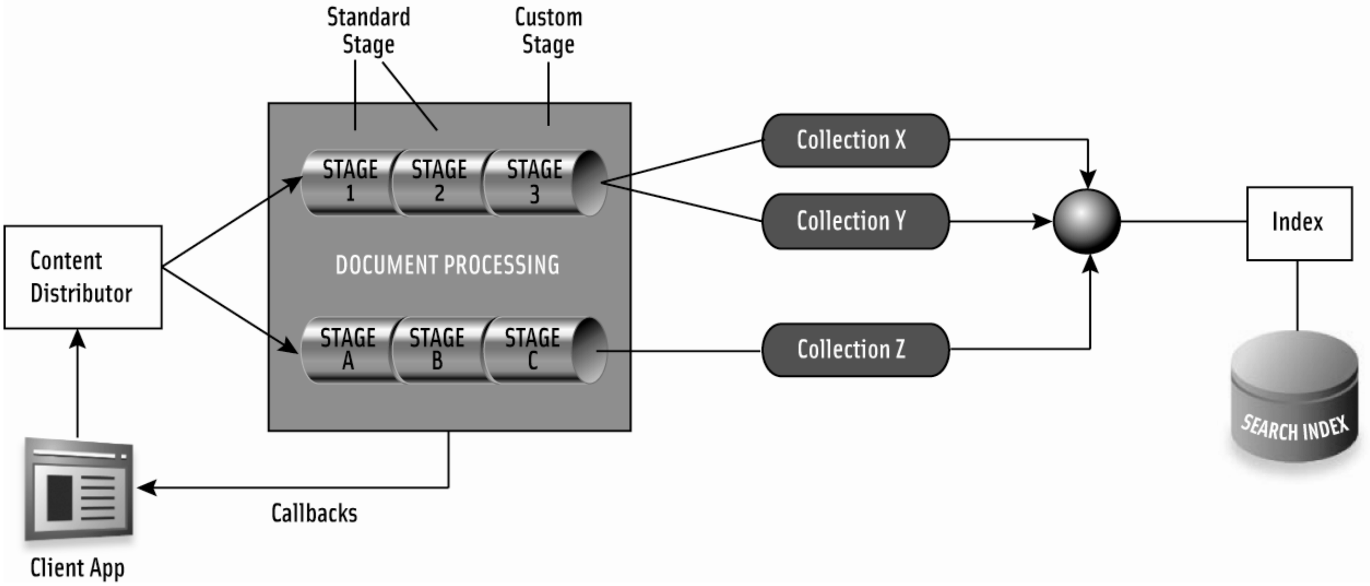
Content flows from outside the system into the content distributor. The content distributor dispatches content to the proper component. A pipeline processes content (serially) on behalf of one or more collections. The content is pushed out in post-processing as XML and is indexed with respect to the configuration of the index.

Document processing can comprise one or more document processing stages *(language detection, synonyms, spell checking, lemmatization, taxonomy classification, custom plug-ins*, etc.). These stages analyze the content and add or remove or transform data accordingly. Using this type of linguistic processing is vital to improving the search experience. Content can be normalized using language-specific document processing, language/industry specific synonyms (by defining dictionaries), etc. It is also possible to remove unnecessary content that doesn't need to be indexed, such as menus, frames, etc., before it hits the searchable index. (See the Linguistics and Search chapter for detailed information on this topic.)

Enterprise search applications usually ship various pre-built stages into different processing components. They are designed to accept certain types of data and operating processing – Web data, XML information, news, etc. Organizations can use the standard stages as is, or create new stages based on existing ones, or augment existing pipelines with stages that they write themselves, combining these techniques as they see fit. Combining processing stages provides search customers with document processing platforms that meet their specific industry needs.

Queries are submitted against the *index* (right hand side of the diagram on the following page), with the critical metric here being *QPS* (queries per second). This is affected mostly by query volumes, query size, and complexity, and by query transformations such as spell checking and query re-writes.

Modular Processing Stages for Flexible Content Refinement



Taking a closer look at normal document processing, the raw content is normalized and turned into structured information – specifically, *name-value pairs*. These name value-pairs are then sent to waiting document processors.

Once the document processors have completed their task, the *output of one document processor serves as the input to its downstream neighbor*. At the end of this lifecycle, the document's name-value pairs are mapped to a field in the index. The index acts in much the same way as a database schema in the RDBMS world, defining the overall structure and constraints of the data it stores. The advantage here is that the scope fields will offer a more flexible search experience compared to the database.

We have a corporate taxonomy. How can we use it to help us with searching for documents?

During document processing, you can assign sets of taxonomic information to documents, prior to their being indexed. Subsequent searches can use this information as a filter and a UI drill-down mechanism.

In a typical scenario, the customer will have deployed a multinode solution with *distributed document processing*. The document processing nodes would be separate and distinct from the indexing-and-search nodes. In this case, the customer would employ multiple collections, use a combination of standard and custom document processing stages, and possibly use a combination of content connectors and the content API for submitting information to the system.

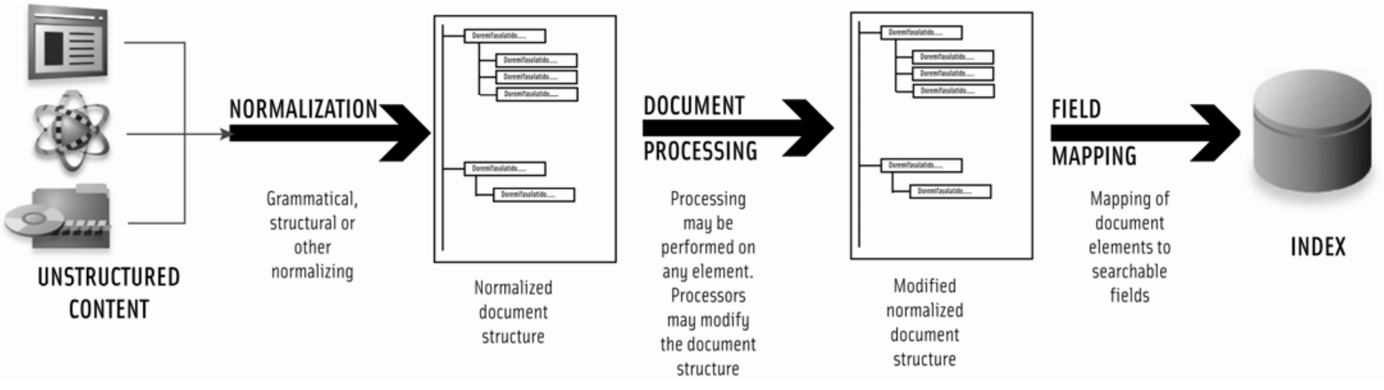
In an *advanced scenario*, the customer would have a setup similar to the typical scenario. However, there would be more nodes involved as part of the installation, and the level of customization of the processing components would be higher. In addition, an application submitting content would use the callback features of the API and would contain logic for automatic error detection and recovery.

Different industries, different solutions

Different business drivers apply to different industry domains, and the supporting structure and quality of content is vastly different from organization to organization. A flexible and tunable search application is needed to *pull disparate data sources together*, to *cleanse and process* it before making it available for consumption via the index.

Let's consider an *e-directory*. Today, e-directories are facing intense competition from sites such as Google and Yahoo! Many want to protect their positions in the market by leveraging search applications, and by moving their traditional print models online. Some are enhancing and differentiating their content offering by crawling and integrating local Web content, providing mapping capabilities, federating to user re-

Some Examples of Content Refinement Stages


UNSTRUCTURED
CONTENT
NORMALIZATION
Grammatical, structural or other normalizing
Normalized document structure
DOCUMENT
PROCESSING
Processing may be performed on any element. Processors may modify the document structure
Modified normalized document structure
FIELD
MAPPING
Mapping of document elements to searchable fields
INDEX

views (restaurants, hotels, etc.), and including other directory content such as White Pages and city guides.

E-directories must overcome content quality (internal database and Web) as well as aggregation and processing problems in order to succeed. Improving the traditional Yellow Pages interface is vital to provide simple digestion of additional content. Using *navigators* and *sentiment analysis* (for reviews) and exposing database content will provide improved usability. This further highlights the ability to process and correlate structured IYP (Internet Yellow Pages) and Web content. Enhancing the search experience and the increased exposure of advertisers will potentially draw more eyeballs to the site and will enable the e-directory to offer alternative pricing models.

> "Knowledge is of two kinds. We know a subject ourselves, or we know where we can find information on it."

In a *knowledge discovery* environment, such as an oil and gas provider's intranet, there may be more human intervention in the preparation of content based on the complexity of content and use of specific industry terms. This can be combined with entity extraction tools to enhance the usability of the content and search experience.

Support for multiple types of documents is a must with anticipated aggregation from file systems, CMS applications, and databases. Custom applications can be included in this environment by using content APIs. Multiple document processing stages will be necessary to correctly categorize unstructured and structured data; the use of *synonym expansion* and *concept extraction* can assist scientific researchers.

OEM integrations with the document processing approach will leverage the search application's built-in connector – in this case, the file traverser for loading content from the OEM's file system. Using the out-of-the-box connectors provides the ability to rapidly support many repository types.

My OEM application is suffering from submission errors. How can I fix this?

When using the content API, use call-backs to detect and respond to error conditions. This enables your program to monitor progress, report problems and attempt to automatically recover from submission errors.

Custom-built applications use the search application's indexing API to push data to the search engine. This requires all original data connections to be performed using the calling application, which controls scheduling, interfacing protocols, and data structures.

The API supports error-logging callbacks, where actions may be triggered when documents fail to be processed or do not make it into the index. It should also support updates and deletions if the system has dynamic content.

Understanding the impact of ...

...Content aggregation

Public search applications have created demanding search users who expect highly relevant search experiences with *comprehensive access to all content*, no matter where the source data originates. As a result, there is a pressing need for flexible content aggregation tools.

So the search provider must bring disparate content together for augmentation by the document processing component of the search application. OEM providers or integrators would develop various document processing stages that support specific application, content and business needs, and allows the manipulation of results.

Key decisions must be made related to *how and when to aggregate* – ingestion time, query time or both (federation or merging of indexed content, for example). Performance and hardware costs need evaluation. Other tradeoffs to consider are the aggregation of content versus raw ingestion (and correlating via IDs), and aggregating at ingestion time versus at result-processing time. It's best to perform content aggregation before the content is processed and passed to the index. This improves speed and accuracy of results and increases user satisfaction.

...Document processing

The primary objectives of document processing are to *apply business logic* to the original content, making it easily searchable, and to *augment content with business semantics* to add value when it is retrieved.

Organizations need to consider relevancy, advanced linguistics and other capabilities, including custom stages. Adding additional stages to document processing will impact the efficiency of indexing content, but is dependent on the complexity of each stage that is added.

The impact of document processing is highly dependent on the quality of the content processed. An important factor for

CONTENT PREPARATION is often integral component of the overall **workflow** that supports business processes. Use subject matter knowledge to best prepare the data for later retrieval.

customers to consider is the cost required to clean and prepare the content outside of the document processing versus the degraded performance ingestion rates.

Customers have *unique content needs* that will require specific custom processing to ensure that it aligns with their business goals. Organizations need to make such decisions on a case-by-case basis, based on their preferences, skill sets, and available time.

Guidelines and recommendations

The perceived relevancy of content and of search success is fairly subjective. User surveys can help an organization understand the needs and concerns of its customers. The next section of this article highlights the metrics needed to measure the success of content aggregation and document processing as well as the potential mistakes to avoid.

Content aggregation

The metrics for measuring content aggregation are straightforward: total content volume, format, and speed. One key metric to consider is the time it takes to reproduce, re-aggregate, and re-index if the data was lost due to a hardware failure, for example. An e-commerce site would need to prepare for the profound business impact of bad or missing data in the index. In this case, raw content should be stored prior to indexing.

Document processing

When measuring the effectiveness of document processing, customers should consider the number of documents per second per server for each node and across all nodes. This will help customers determine the efficiency of each system. In addition, customers should monitor error rates; high error rates indicate problems with the source content, the document processing stages, or the configuration of the index profile.

Common mistakes include failure to optimize the document processing stages, not understanding the processing data model, not appreciating certain index configurations, and not understanding which processing stages add value. Many customers waste valuable time duplicating code and functionality that comes standard with search applications.

> **The impact of document processing is highly dependent on the quality of the content processed.**

Best-practice guidelines for improving search

Content quality has a profound impact on the usefulness of search applications. Not only should organizations incorporate the cleansing and preparation of content into search initiatives, but they should look to leverage powerful content aggregation tools. Sophisticated document processing capabilities must be leveraged to their full effect to provide a best-in-class search application. Here are the best-practice guidelines:

Content aggregation

» Plan the aggregation strategy. Understand the data, and consider ingestion time versus query time aggregation.

» Consider correlation IDs. In some cases, it can be enough to use an ID field to relate content (for example, mapping zip codes or post codes from Web searches with IYP database content).

» Store content locally. The search solution should automatically store a local copy of content before submitting it for processing. This is not necessary if the content can be quickly and easily regenerated from the source, or is obtained by crawling. This allows content to be reprocessed in future based on changing business needs.

» Automate the process. Content acquisition and aggregation can be largely automated via scripts and applications. The less human involvement, the better!

Document processing

» Prepare! Spend time evaluating document processing needs and augment content as needed. Consider your goals carefully and be deliberate about what and how you process. The index might need to be updated to accommodate your processing.

» Size accordingly. Use the appropriate number of hosts for feeding and document processing. Consider the impact of ingestion rates, linguistics, document size, etc.

» Use existing document processing stages. Use prefabricated stages where possible.

» Optimize document processing. Remove all unnecessary document processing stages. Choose the right processor for the content type and task at hand. Remove all unnecessary stages that can degrade performance.

» Monitor the statistics of CPU usage and I/O wait time per stage. Add processing servers where necessary to tune and overlap CPU usage and I/O time.

» Submit documents in batches. When using the content API, submit document in batches of between 10 and 200 — depending on the average size of the individual documents. Tune appropriately to minimize processing overhead and increase overall throughput.

» Update the index profile. Ensure that the document processor, index, and front-end are compatible. Some document processing changes will necessitate index changes. Align document processing with the content of the index.

- Build a library of processing stages. When building custom stages, try to solve the given problem as generically as possible. You can reuse code in subsequent projects.

MINI CASE STUDY: Global publisher uses content aggregation and document processing tools to provide comprehensive scientific search site.

WHO: Global scientific publishing provider.	CHALLENGE: To provide the most intuitive science reference site and to allow scientists to process and retrieve content to support their needs. The publisher has a wide variety of content sources and large data volumes (90 million Web pages, 15 million journals) and provision of entity extraction and classification for simple navigated search.
SOLUTION: Single indexing of unstructured and structured content at high content volumes and ingestion rates. Advanced document processing and linguistic capabilities enabled results to be blended from a variety of target systems.	TECHNOLOGY: Advanced document processing stages with advanced linguistics and entity extraction/classification for navigation. Document processors can also call out to Java and C++ custom processors. Content aggregation via Web crawling, files, database connectivity, and tuning of result processing framework.

FAQ

I don't believe that the automation of content cleansing will work for me. What do you suggest?

The best practice for search is to leverage automatic tools where possible. This will shorten the time it takes to take content in its raw form and publish it in the index. If you want to use staff editors for your specific vertical needs, you have to consider that normalizing relevancy from multiple results sets is time-consuming and imposes a management overhead.

Is there a limit to the number of document processors and processing stages that I can use?

In theory there isn't, but this needs to be balanced with the latency tradeoffs that may be experienced with too many processing stages. Using more document processors will ultimately speed up the ingestion and processing of content to the index. You should map each processor to the type of content – Web, news, XML, etc.

What is sub-processing?

A set of processing stages that work to process distinct documents from the main pipeline. Sub-processing aggregates e-mails and attachments, for instance.

How should I appropriately size the document processing part of the search application?

The sizing of the document processing component depends on a variety of factors: Use of linguistics; required throughput; content characteristics; failover requirements; etc. As a simplified answer: If possible, use at least one extra host for feeding or document processing. If you feed more than 10 documents a second, use two hosts. If you use linguistics heavily, use two or three hosts. If you deal with large documents, use two hosts and lots of RAM (3 GB). If you do all of these things, use three to four hosts.

Why should I integrate my search application with the structured data in my database? Isn't database mining sufficient?

Search applications provide an index architecture that is well suited to both structured and unstructured information. Integrating with a relational database is performed for two reasons: 1) relational databases are not very efficient for handling large query volumes, and 2) integrating a large number of different data sources into one index and one search bar provides a more convenient search experience. For example, an e-directory would look to publish both Web and database content, such as company description or offerings (Web) and opening hours or price catalogs (database).

EXIT

Applying Business Rules to Search Relevance

Relevance models determine the ordering of search results, but such models need not only be based on mathematical IR (information retrieval) methods - they can also incorporate the business logic at hand, to organize the answers in a way that is commercially or organizationally optimal.

Organizations today are governed by business rules and workflow. The aim of the game is to tune the appropriate business rules to meet the needs of the business – and of its markets. But how do business rules apply to search technologies?

By definition, *business rules are units of logic that govern how a system should behave or act*. For example, a business rule might state that no credit check is to be performed on return customers who are applying for insurance policies. Another rule might flag a "Level 3 alert" if an abandoned car has been sitting in a "threat hot zone" for more than 2 weeks. The search application should be an open platform, allowing customers and search providers to deploy business rules at various stages of the search system - at ingestion, ranking, query transformation, or at alerting time.

Business rule configuration needn't be overly complicated. The most direct approach is to leverage the *inbuilt management and monitoring tools* of the search application. Here, business managers can adjust the relevancy and ranking models that are assigned to a particular subset of content, so, for example, a letter from the CEO pops to the top of each query, or a particular product leads the list of search results during a marketing campaign.

5 things you should know about business rules

1. Well-constructed content is vital. Business rules should augment good content, not hide weaknesses.

2. Understand the time it takes to formulate and apply business rules rather than relying on the standard relevancy model.

3. Strive to eliminate futile queries, abandoned sessions, and multiple similar queries by the same user in the same session.

4. Measure, measure, measure, and refine. Incorporate a closed-loop system of measurement, reporting, and refinement.

5. Business rule management allows OEMs to better understand the impact of search on their applications.

On the other hand, developers or IT managers can dig deeper, augmenting and *analyzing the performance of the document and query processing stages* to mine the query results themselves. That way, they can determine what queries and results do and don't work well in the search system, and then configure the system accordingly.

> Monitor user query behavior to refine the search system over time.

A deeper approach still is to construct a business rule environment, providing a *syntactic or declarative model* for creating, applying, and managing rules and the actions they trigger. In this scenario the rules are applied to new content as it appears, performing actions as directed by the rule.

Users can be more certain that they've got a flexible, best-in-class search solution when they understand the impact of business rules management and combine that understanding with the functionality of built-in search features such as document processing, index profile, relevancy/rank profiles, linguistics, and analytics tools.

This white paper will look closely at the impact of business rule management on search technologies and will review the options available to managers eager to leverage the capabilities available in today's search applications.

Enhancing search with business rules

Search application goals will vary significantly from one organization to another – hospitals will have different needs than an online careers Web site, a producer of consumer packaged goods, or a federal intelligence agency. The one common thread is the goal of displaying the right information to the right person at the right time – and in the right order without extraneous noise. Sometimes the information is simply the results of the search (as in investigating archival data); other times it is content set aside for further action or an alert to do something (as in monitoring streamed data).

It is important to note the "right person" aspect of this goal. Within a knowledge discovery environment, user-specific business rules should differentiate content from one role to another. This is where many standard web search engines fall down; but the requirement is important because business rules always have scope, and that scope is generally defined by the activity of the person (or role).

I run an e-directory. How can I evolve my business model by leveraging business rule functionality?

Offer Gold, Silver and Bronze advertising packages on top of the paid inclusion model. For each package, you can apply different levels of ranking via the management interface and tie them to queries and keywords.

Enterprise search engines do have this capability courtesy of the management tools shipped with the software. The tools have typically been designed to allow business and IT people to configure and improve the search experience so they can minimize *time-to-relevant-information*.

It's possible to configure for a particular user by monitoring that user's query behavior; the search system can be refined over time. Such management tools usually consist of three main elements: *rank tuning*, *query reporting*, and *user management*. Let's look at each in turn.

Rank tuning

Rank modifications based on business rules enable the search provider to influence or override the automatic ranking of documents – for example, by directing users to business-generating pages. There are several techniques to configure the ranking of documents:

Absolute query boosting: Suppose you want a document to be displayed consistently at a given position in the result set - for example, in the top position - when a user searches with a specific query. In this situation, you can specify a document-query combination that assigns a fixed absolute ranking position to a particular document. This ensures that the specified document surfaces within the result list whenever a user is searching with the specified (matching) query. You can also prevent individual documents from being displayed during such searches by giving it a ranking value of zero, for example.

Relative query boosting: This feature is valuable if you want to ensure that a particular document is always displayed among, say, the first 20 documents in the result list, provided a user searches with a specific query. For all other queries, the ranking position of the document will not be impacted by any boost. Here the business manager specifies a document-query combination and assigns a sum of ranking points (that is, enhancing the document's relevancy score) with which the document's overall ranking value is to be increased whenever a user is searching with the specified query.

Relative document boosting: This feature is useful when it's necessary that a particular document is always displayed within the first 20 documents in the result list, no matter which query a user has submitted. At the same time, the user does not want to assign a fixed result list position to the document. Applying this feature, it's possible to specify that the overall ranking value of the particular document has to rank higher or must be increased by a certain number of points.

It is also possible to manipulate the relevancy score (rank) related to categorization using taxonomies present in the user's environment. This allows a rank boost to be applied to all documents within a category against all or specified queries. For example, the user may want to boost the medical category for queries specific to a particular area of medicine. Boosting for specified queries may be useful when tied to understanding and mining the query logs from the search system. Here, the most frequent queries on the site may be considered for boosting against specific categories.

Use other features, e.g. navigation, to support and augment the effectiveness of business rules.

In addition to applying the *static rank tuning mechanisms* discussed above, users can also adjust the *dynamic relevancy of documents* based on the rank profile concept, meaning that documents can be given higher ranking values based on freshness,

completeness, authority, statistics, quality, or location. Using the freshness parameter (how recent is the document?), recent news articles or press releases can be boosted so that they appear at or near the top of result sets, for example. This feature is discussed in depth in the SBP Relevancy white paper.

Query reporting

Query reporting or mining of the raw query logs is a task that is overlooked by many organizations when they're running search applications. They incorrectly assume that once the search application is up and running, it will look after itself. Although the search engine is designed to continually return 'relevant' results, the needs of the search user and the business environment are constantly evolving so it's crucial to understand the wider and more role-specific needs of the search base. This can be achieved by *monitoring query trends*, *volumes*, *successes*, *zero hits, click streams*, etc. and it provides the business owner the tools to adapt where necessary. We explore this area in more depth in our Benchmarking Search white paper.

For example, an e-directory provider may want to know the total volume of queries, or the top queries processed in a given time period. By doing so, the company can review its advertising revenue business model and potentially charge more for particular query terms in near real-time. Conversely, the directory provider could also analyze the top 'x' queries that yield no results and configure the system so that search users are at least given an alternative path to follow, such as 'find similar…?'.

Another example: mining the query logs of a mobile operator discovers that when their customers query for a song, they are most likely to then download its related ring tone. So, why not make that the follow-up question "automatically"?

Ideally the management interface should convey these results in a simple graphical manner to shorten the time-to-action. Analyzing query logs is a task that organizations should take seriously if they value the investment in search technology. The frequency of analysis depends on the business need – every day, once per week, per month, etc. – and on the resources available.

User management

Many of the elements discussed above work best when applied to specific search users or to groups of users with similar profiles. The administration tools inherent in search applications should enable simple and *effective user management*. Ideally, they should provide configuration (creation, deletion, modification, etc.) for users, groups, collections, or roles, so that results are targeted differently towards doctors, nurses, or medical researchers, for example. In an e-commerce environment, it can be difficult to manage users manually with the administration tools. So it's a good idea to integrate data from, for example, billing systems to automatically profile the user, not unlike the collaborative filtering functionality used

> "I don't search, I find."
>
> *- Pablo Picasso*

by Amazon.com to suggest new products to the user based on his or her previous spending patterns and demographic profile.

In a typical deployment, organizations can enrich the document processing to control the static rank. They may also have a customized index with multiple rank profiles, based on the dynamic relevancy parameters discussed above, and would perform the standard query processing. The management functionality would likely be used for externally controlling the boosting of documents and for reporting tasks.

My CRM system features a search application and I need to modify the ranking for thousands of individual documents. Is there another way to do this, rather than using the management GUI?

It's possible to bulk-load rank tuning tasks. Similar functionality is provided but XML files are used as the input. The XML file will contain a specification of the rank modifications to be performed. This approach is preferable if you're able to extract the rank boost information from other data or applications.

In more advanced deployment scenarios, search providers would use the features described above and then apply document augmentation to enrich the query processing for tuning queries and to map them to the desired rank profiles and business rules. Boosting of documents (top 10, relative/ absolute boosting) will occur where queries are tied to specific boosting models. Search providers can implement the ability to restrict or block out documents from particular queries in addition to using advanced alerting tools. Reporting and query mining and analysis would be pivotal to such deployments.

Finally, for scenarios where the content is streamed and the user model is centered on alerting rather than ask-and-answer querying, the role of the business rule changes from affecting results ranking to discovering patterns and initiating actions based on them. A wide corpus of content is compared against a control set and if a candidate for "success" is found, it is stored in a separate index for further investigation. The business rules determine what constitutes a possible success. Ex-

..there is a REAL BUSINESS need
to run business rules
and analyze query logs
and offer reports...

ample scenarios are anti-money laundering, where business rules are used to find patterns in deposits and withdrawals; copyright infringement, where business rules determine whether or not something is a copy; and online black market trading, where business rules find patterns on sites that identify candidates.

We are a large organization and our most popular query is 'lunch menu'. How can I ensure that the searcher receives today's menu?

Within the search application, business rules can be configured to be time- and date-aware. For example, if a user searches for 'menu', he or she will be shown the menus relevant for that date. Also, rules can be implemented to promote breakfast, lunch, or dinner menus, depending on the time of day that the search is performed.

Alert-based business rules are generally more independent of the search engine, being managed through a separate – and often custom from user to user – management portal. They may be as simple as a selection of choices, or a complete logical language with the capability to group together rules into composite rules complete with full Boolean logic.

Different industries, different solutions

Different business rules apply to different industry domains and need to be configured to suit each business environment. Let's look at the example of an *e-commerce* site. It can use the business rules inherent in the search solution to control which products and services are delivered to the customer based on the particular query. By tuning the system in this manner, the e-commerce site can promote products that deliver higher margins, say, ahead of other products that also match the particular query term – in effect, boosting them to the top of the results list. The impact on the top line can be profound: for example, such capabilities can help offload end-of-season stock which in turn can reduce inventory, reduce warehousing costs, and also give the e-commerce retailer the capacity to stock new and 'hot' products.

From a *knowledge discovery* perspective – think of a pharmaceuticals manufacturer, for example – there is a real business need to run business rules, analyze query logs and offer reports in order to tailor and improve the search activities for operations such as R&D, clinical trials, sales and marketing, etc. Here, the analysis may pinpoint the need for custom dictionaries or linguistic capabilities (spell check, lemmatization, synonyms, etc.) to reduce the mean number of queries or 0 hits. In this scenario, business rules are effectively being used for '*fault detection*'.

For *monitoring and alerting* environments, business rules are used to trigger activity, or isolate outliers or candidates for further investigation. Their value is in the dramatic reduction of investigative content. In other words, they act as a filter that removes all the unnecessary pieces and leaves only those that are candidates for deeper investigation.

Business rule management allows *OEM*s to better understand the impact of search on their applications, and to tweak as necessary to improve the value-add that search provides. This is particularly important for CMS applications where documents are published and are subsequently available in the search index. An alerting functionality can be applied here to notify users of new content so they don't have to search for it specifically. These types of tools eliminate the need for custom coding of search rules in an ad-hoc manner, and bypass the need to generate custom reports. If necessary, custom reporting can be achieved by exporting the reporting logs/data from the search application to a third-party reporting tool.

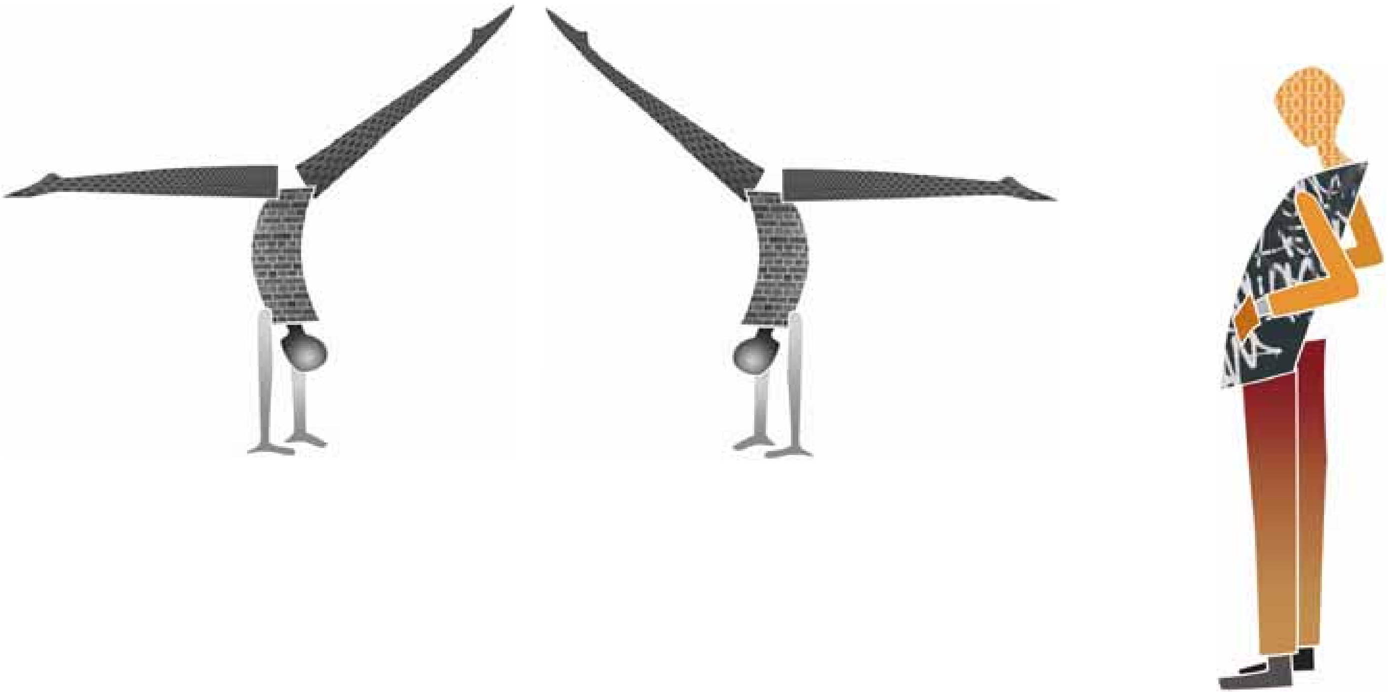
Understanding the impact of business rules

The search provider's foremost objective is to provide a flexible and configurable platform that displays what the business owners believe is the most relevant information for different contextual environments. The search experience needs to be altered easily – without needing to write custom modules – so that the search application reduces the time-to-information and helps users avoid having to mine for what they need. By monitoring the end-user query behavior to refine the search system over time, the search experience steadily improves, eliminating futile queries (0 hits) and enhancing user satisfaction rates.

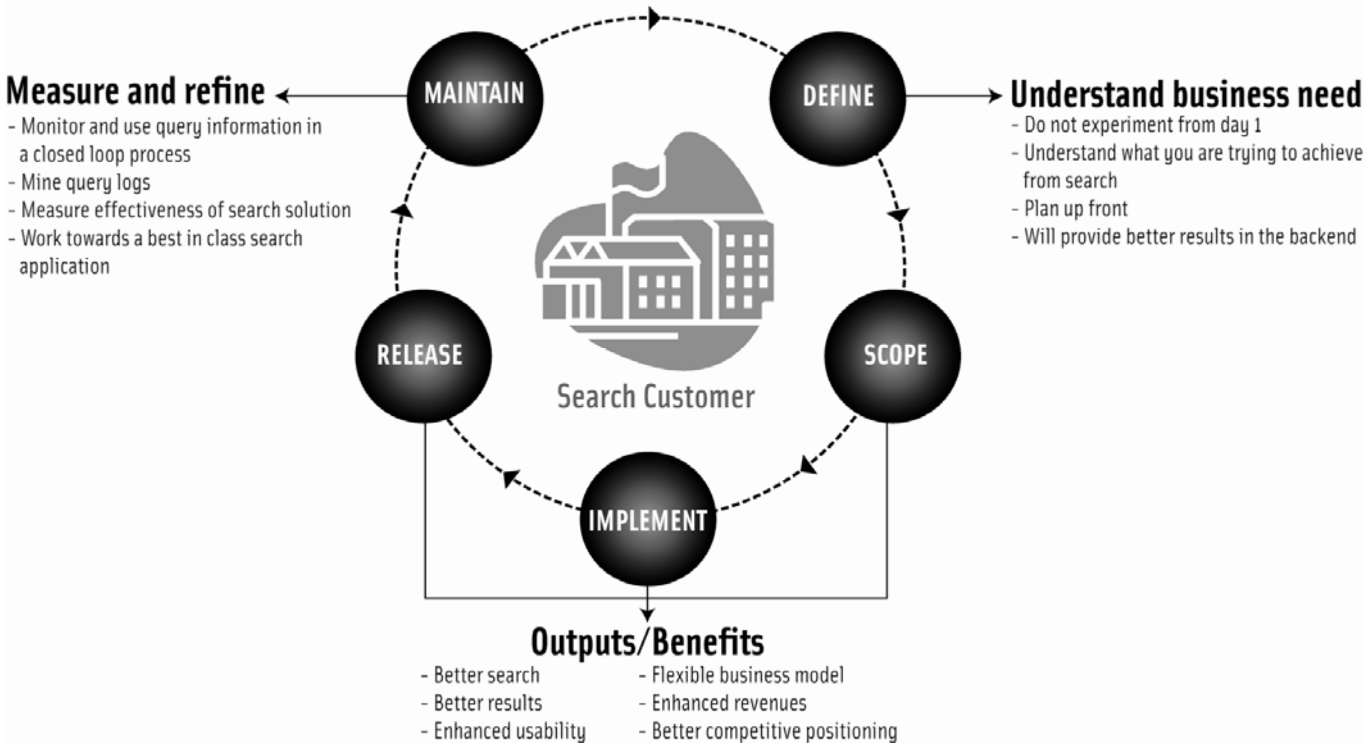
By integrating management tools with the index and rank profiles, managers can ensure that the search framework aligns and supports the organization's strategic and tactical goals. That means that in the case of a marketing campaign, say, more important or more relevant information or products can be surfaced relative to a query. Managers can also quickly test (and deploy) different assumptions and models based on changes in business direction or environment.

From the IT support perspective, management and monitoring tools ideally *reduce the burden of custom programming* and integrations. They can also obviate the need to produce custom analysis and reports, returning more control of content to the business owners. However, experience indicates that business rules often end up being customized UI "hacks" rather than structured and well thought-out configurations. IT managers must be aware of the resulting maintenance issues.

It's also critical to balance the need for control with the tradeoffs associated with management and monitoring tools. Managers first need to ask themselves whether the standard reporting functionality – the functionality embedded in the search application – is sufficient for their business requirements. If it is not sufficient, they need to understand that it takes time to formulate and apply business rules rather than relying on the standard quick-configuration relevancy model. It also helps if managers appreciate the differences between the static business rules model and the standard dynamic relevancy model with its multiple rank profiles. Ultimately, logic on the query side will have to be defined so that the right rank profiles are selected. This will take time.



Iterative operational steps for good search applications


Measure and refine
- Monitor and use query information in a closed loop process
- Mine query logs
- Measure effectiveness of search solution
- Work towards a best in class search application
MAINTAIN
DEFINE
Understand business need
- Do not experiment from day 1
- Understand what you are trying to achieve from search
- Plan up front
- Will provide better results in the backend
Search Customer
RELEASE
SCOPE
IMPLEMENT
Outputs/Benefits
- Better search
- Better results
- Enhanced usability
- Flexible business model
- Enhanced revenues
- Better competitive positioning

Another area for consideration is which *reporting tool* to use to mine the query logs and which *web tracking tool* to use for monitoring click streams. Although it makes sense to keep the reporting of search within the application itself, this is not always appropriate for every organization. In this scenario, organizations should look for search solutions that allow outputs to be integrated into third-party solutions, or into home-grown tools. Typically the output from the reporting tool will be in a standard ASCII text/XML format.

Finally, there's value in appreciating the additional management and maintenance time and constraints of building and deploying business rules. This must be coupled with a thorough understanding of the business needs and a realistic view of what's achievable. It is advisable to continually analyze the usage trends long after deployment.

Guidelines and recommendations

The true measures of any search application are its ability to return relevant results, and the time it takes for the user to find the information needed. It's no longer acceptable to return futile queries, so the search system must be able to provide an *'alternative path' to the right information* – 'similar to…' or 'people like you also liked …'

By measuring the average and mean number of queries per session, you can understand how long it takes to locate the right information, or how many queries are needed. At the same time, monitoring the number of similar queries shows where there is room for improvement if the user has to rephrase the query several times. It is also important to measure and monitor session abandonment rates, where users may try two or more queries before giving up a search.

In all cases, business managers can tune the search interface or match queries with particular content to improve the search experience.

As noted earlier, many organizations fail to fully understand what they are trying to achieve before deploying search applications. Furthermore, they are likely to neglect the frequent *monitoring and tuning* aspects of search. Planning upfront will provide rewards later on, such as better search and results processing.

> Business rules should augment good content, not cover up its weaknesses

Other common oversights include the failure to fully understand boosting models and their application, and failure to mine the query logs for useful information. Unfortunately, organizations too often map queries and boosting of documents without fully considering why they are boosting particular content. *Without mining the query logs, it is like the 'blind leading the blind' when it comes to refining the user experience.* A crucial success factor is a familiarity with the usage patterns and the search intent.

Additionally, organizations must acknowledge the need for continuous and iterative measurement, reporting, and refinement. Advanced users of search applications usually assign this task to a particular employee or business owner. That way, it becomes that assignee's responsibility to continually monitor the search application, allowing for *quick detection of trends and fast solutions to search trouble spots.* Many e-commerce companies have reported increases in traffic and advertising-based revenue by using effective search tuning and configuration.

Five fundamental steps for improving search

In theory it is not difficult to put control back in the hands of the people who understand the dynamics of the business, but it can be tough to achieve in practice – especially from the standpoints of resources, experience, and company politics. That equation needs to be balanced carefully. There are five fundamental steps for improving search effectiveness. Here's a summary:

1. *Understand your business rule needs* – management and reporting tools enable flexible rank tuning, boosting, and blocking of documents with or without respect to a given query. These tools also produce reports to help business managers to track the efficacy of the system.

2. *Determine the best method for rule deployment* – depending on the scenario and an appreciation of the trade-offs discussed above, it's possible to apply other types of rules using the core of the search application – the document processing and query processing stages, the results processing stage, and configuration of the index itself.

3. *Leverage the experience and knowledge of the search application* – business rule tools have been configured to integrate and operate seamlessly with the search framework.

4. *Measure, analyze and refine* – use reporting tools to analyze query logs and measure the effectiveness of your search solu-

tion. Get close to the reports to continually refine the search experience; use boosting or hard-wired queries to draw out important information as quickly and cleanly as possible.

5. *Develop a 'search analyst' role* – this new position will be responsible for the closed-loop cycle of managing periodic updates and refinements, and will make sure that the results of the analysis are fed back into the search experience.

MINI CASE STUDY: Global e-commerce provider streamlines its sales channels by using monitoring and management tools to know its customers better.	
WHO: Worldwide IT e-commerce site.	CHALLENGE: To enhance the sales throughput of its e-commerce site by better understanding its user base and appropriately targeting highly relevant information.
SOLUTION: Knowing that well-constructed content is vital, the company made sure that its business rules enhanced its existing content rather than hiding weaknesses. The project team put anchor text and links into clean content so that the site became self-organizing. Navigated search techniques were built in to support and enhance the effectiveness of the business rules.	TECHNOLOGY: Advanced business management and monitoring tools, rank tuning (and bulk loading of rank models) and URL boosting.

FAQ

What is meant by boosting?

Boosting is the act of moving a document higher up the result set. It is achieved by adding points to a document's rank value. By default, documents with the highest rank values are received by the user sooner than documents of lower rank values.

What is a query log and how can I mine the data?

The search application will track the statistics of issued queries and information about them, e.g. volume, times, etc. This information is stored in ASCII text files called query logs. The search system's reporting tool will allow you to graphically mine the logs; otherwise you can push the data to an alternative preferred reporting tool.

What is a boost point?

A boost point is a value that is added to a document to increase its rank score (relevancy) relative to other documents returned in a set of search results. Boost points may only be added for certain user queries, or across the board for all queries.

How can I delegate management/reporting tasks to multiple business owners?

The business reporting tool should permit multiple user 'roles' that align with admin (full control), rank tuning, reporting, and user management. Each role has its own respective rights.

How can I alter the rank tuning in the management tool?

There are several methods of doing so, such as creating saved queries, managing saved queries, creating and editing document boosts, and/or direct input. Creating saved queries allows you to increase, decrease, or block the result set of a collection for the query that is being searched against. Managing saved queries allows you to delete or edit the rank of a saved query. Creating and editing document boosts allows a user to boost a document regardless of what query is entered. Direct input allows a user to create saved queries and add boost to documents that are not necessarily in the searchable index.

The Usability Factor

The old adage "If the user can't find it, it ain't there" is very much true for search applications - for both the query entry page, and the results page. Designers of search application sometimes forget that their users tend to be less technical than themselves.

New computer systems are brought online daily – either as upgraded versions of existing installations or as completely new systems – and they all bring with them the potential for increased efficiency, lower costs, and increased revenues. They are optimized for maximum performance, interoperability, flexibility, and ease of maintenance.

What is your new system really worth if prospective customers can't find what they are looking for, or if the system is simply too slow and they abandon your site as a result?

Usability is a key component of a system's total delivered value; it should be obvious to the end user how to navigate the system. For example, an editor should be able to manipulate text in an intuitive way, so it must be easy to publish and retrieve information from a content management system. Good usability also applies to search-powered systems and is essential for realizing a system's full potential – studies highlight that e-commerce sites lose approximately one third of prospective customers with every unnecessary click.

Search usability describes how the user is guided through his or her interaction with the system – from start to finish. Ultimately, a system's usability is judged by its users, so it is important to follow a user-oriented design process when developing or revising a search-powered system. Understanding the user, whether they are end users, business users, administrators or developers, etc. is crucial to success. Different degrees of functionality should be provided to cater to the user's search experience and information need.

To ensure a good result, it is very useful to combine user testing with other forms of usability evaluation in order to collect feedback during the design process, following these steps:

» Define the search experience.

» Align system design with the definition.

» Let real end users test and evaluate the system.

Enhancing search through usability

You can focus on usability to enhance search by defining a good user experience for someone searching on your site. Consider these alternatives:

» Should your search be very simple, or powerful and flexible? Giving the user more search choices will make the user interface more complex.

» Will your users want to compare results, or will they want to find a single relevant result? Do you want to provide speed of search to keep users at your site longer?

Your answers to these questions should evolve from these major design forces:

» *Know your business*. What are the business reasons for having search at your site, and what is the nature of your content?

» *Know your users*. Who are your users and what are their goals?

Know your business

Consider what *role* search plays at your site. In other words, what is the basic business case for the search you will be designing? In what way does search contribute, and how important is that contribution?

For example, in an e-commerce search environment, search has a direct impact on the browser-to-buyer conversion rate. In a knowledge worker environment, search would impact both motivation and productivity.

How do I measure usability?

The metrics should be aligned with how you define the user experience of your system. Often, the metrics are related to users' performance on a set of test tasks: success rate (can the user complete the task at all?), time needed to complete a task, error rate and users' subjective satisfaction.

It is important to understand what makes your content unique, since it will influence who your users are and what tasks they will perform. Conversely, you may want to base any new content or process content on the types of users you'd like to attract.

Know your user

A useful technique for understanding your users is to define a set of *personas*. Each persona represents a fictional user. Each can be named and assigned a background, along with a job description, skills, hobbies, and personality.

Depending on the complexity of your offering and the breadth of your potential user population, you should aim for three to ten personas. Your personas should be treated as external users. However, you may also want to include one internal persona such as a "Webmaster" or "search maintenance expert."

As you work with the search design, consider each persona individually. Would this person understand and like this page? Why or why not? Working in this way also helps you to recognize that all users are not alike.

User intentions (sometimes called use case scenarios) comprise a technique that is used to decide what functionality and content to include at your search site. You can develop a list of possible thoughts that each of your personas might have when visiting or searching your site.

Set usability goals and prioritize them based on your business goals for search, your content and your users. It is suggested that you select a maximum of three to four goals and put them in order of importance. Prioritizing your goals may be difficult, but it will help stakeholders develop a consistent vision.

Align system design with the usability goals

Now it's time to move to implementation. The general guidelines discussed later in this chapter can be used as a starting point for your design. In summary, they involve starting design implementation, creating a prototype, testing, refining, and testing some more.

5 things you should know about usability

1. Good usability is necessary in order to realize the full potential of your system.

2. End users are the ultimate judges of your system's usability.

3. When designing your system's user interaction, test it often with real end users

4. Not all users are the same; different categories of users represent different requirements for your system's user interaction.

5. In user interface design, less is more.

Keep it simple

Using pictures, interactive design answers two user questions, "What can this do for me?" and "How do I make it work?"

A key concept is *less is more:*

- » The less text you put in a picture, the more it is likely to be read.
- » The fewer fields you put in a form, the more are likely to be used correctly.
- » The fewer graphics you include, the more likely it is that the user will perceive the interface as easy to use.

As search becomes a more powerful tool, good interactive design is necessary to ensure that your users can take advantage of the features you choose to offer.

Different industries, different solutions

Search plays different roles in different industries, so, you must *define the search experience* to guide implementation of the user interaction.

In an *e-commerce* setting, site visitors purchase products, services, or content. Search and search-enabled navigation help visitors locate what they want. The user experience can impact conversion rate, customer loyalty, and the frequency with which a customer visits the site.

Looking at a *knowledge discovery* situation, the users are employees, students, or clients of the site owner. The site owner has a vested interest in helping these visitors be more effective in their work or daily tasks. Search enables workers to quickly and easily locate job-related content, and it may help with basic information analysis. User experience can impact both motivation and productivity.

OEMs integrating search technology should align the search experience with the behavior of the rest of the system to offer consistent user interaction.

Understanding the impact of usability

When considering the usability of Web sites, every visitor should be able to easily understand the page or site. Search input, results, and result navigation must also be effortless for users. When evaluating a page, you have to consider whether visitors without special knowledge will be able to understand it.

Discussing visual design: CROCodile

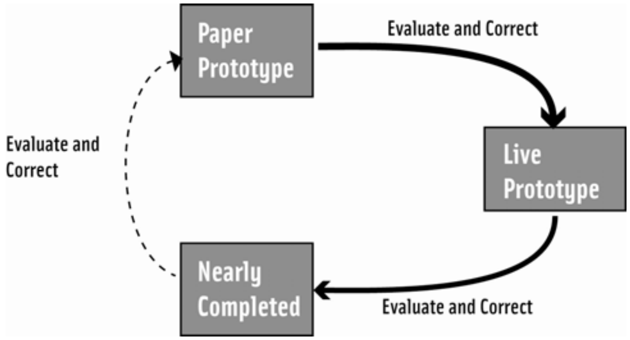
CROCodile is a simple acronym checklist technique you can use when talking about screen pictures. It can help you to focus on how the pictures communicate:

- » Contrast: What stands out?
- » Readability: Do you want to read it? Is it easy to read?
- » Organization: Is it easy to see parts, groups and order?
- » Clickability: Is it easy to see what is clickable and what is not?

Contrast: Squint at the page or screen. What jumps out at you? These are the elements with highest visual contrast, and the items that the user will usually notice first. If these high-contrast items are also the most important items, then all is well. If not, then a visual redesign may be necessary. For a usable search, it is essential that users can easily spot the search function.

Readability: Readability issues are especially important for results pages. Ask to see typical results pages with real data in order to judge readability. When you glance at a page, ask yourself, "Do I want to read this?" If your answer is no, there may be too much unbroken text on the page. Shorter line lengths, short paragraphs, subtitles, and call-outs can make the same text appear more inviting.

Iterative Steps in Design Prototyping



Organization: Is the picture divided into easily perceived parts or groups? Organizing a lot of information into a few chunks makes the picture easier to understand. Users prefer well-organized pictures, and understand them more quickly, too. Ideally, groups of information are arranged from top left to lower right. Groups do not have to be separated with lines and boxes. The less you put on the page, the easier it looks to work with.

Clickability: It should be immediately clear to your users what is clickable. Usability tests repeatedly demonstrate the following points:

» Many users don't realize that a logo in the upper left corner is usually a clickable link to the home page.

» Users do not think it is fun to wave the mouse around to see what is clickable.

» Clickable text links are easier for users to notice than clickable pictures or graphics without text.

» How clickable a link looks depends on what's around it. (Again, contrast is important!)

In general, underlined text and buttons are assumed to be clickable.

WE CAN HAVE FACTS
WITHOUT THINKING, BUT WE CANNOT
HAVE THINKING WITHOUT FACTS

- *John Dewey*

Does the size of the search input field matter?

Yes, in this case size matters. Make the input field long enough to encourage users to type at least two or three words. Longer queries are more likely to produce the results the user wants.

Guidelines and recommendations

The best way to ensure a good search user experience is to incorporate user-centered evaluation activities into your development process from the start, applying a process of controlled iteration:

- » Do a first draft design on paper. Evaluate and correct the design before proceeding.
- » Do a second draft of the screen pictures. Evaluate and correct the design before continuing.
- » Begin the actual implementation. During the quality testing and de-bugging phase, run live usability testing to identify remaining problems. Make corrections in the final days or weeks of development.
- » As part of the design, consider how you will measure usability once the system is in production: through logging, online surveys, feedback mechanisms, or continuous usability testing. Once the system is in production, you need to track usability and attempt to identify required changes for the next version.

Test early and often

Clearly, the earlier a problem is detected, the less it costs to fix. Consider involving designers, developers, and any hands-on stakeholders in evaluation activities such as focus groups or user tests, rather than outsourcing the work entirely. As first-hand observers, they will more easily reach consensus about which problems remain, and also about the relative importance of various problems, and focus their efforts accordingly.

Internal walkthroughs

Internal usability reviews are inexpensive and fast, and should be done before usability testing. You can have a team member assume a persona and walk through the tasks that are appropriate for that persona. Internal walkthroughs work best when you are familiar with your users and their tasks.

Expert reviews

Usability experts perform usability tests and can predict potential user problems. In an expert review, you simply have the expert look through your system and list potential problems. But there are some disadvantages to expert reviews: some potential problems may not be identified, for example. Also, experts typically identify problems that won't affect users.

Focus groups

Focus groups are good for testing site names, content ideas and presentation, general layout, and visual design. They can elicit user intentions ("Why would I go to this site?") as well as user opinions about competing sites. Because they take place in a group setting, focus group results may be invalid if one or two opinionated people have been permitted to dominate a group, or if the facilitator asks leading questions or otherwise indicates which answers are most welcome. So it's essential to use an experienced and objective facilitator.

User tests or usability tests

User tests are conducted with a test user and a facilitator. Any number of observers may also be present. In more formal usability testing, observers are concealed behind one-way glass or observe via closed-circuit television. Prior to the test, a script must be written, logistics worked out, and observers chosen. After the last user has competed testing, problems must be compiled categorized, prioritized, and possible solutions discussed.

My site is an Internet-facing movie database. What would be typical user intentions for my system?

would depend on whether you target a niche audience or the broader public. For a general movie database you might have fan, ent, and actor as typical users. The user intentions may be mapped according to specific functionality in the following way:

User	User intention	Ideas for functionality
Fan	"Find a list of movies that Viggo Mortensen has been in"	Names searchable
Agent	"Ensure that all the celebrities I represent are getting adequate press coverage"	Create links on "My Page" to the sites I visit regularly, and track when information is updated
Actor	"Research all movies involving a certain producer/director"	Name search, contextual navigators on person entity. Extract person entities based on role: Director, producer, etc.

By including designers, developers and stakeholders as observers, you can use usability testing to ensure a shared understanding of which problems should be tackled and what the potential solutions might be.

Usability tests can be performed on prototypes or early versions of the site. It can also be very helpful to make comparative tests of competitors' sites. It's important to schedule usability tests early so that time and resources are available to make the necessary changes. A test scheduled when you aren't motivated to make changes is called a user acceptance test. This is usually a good test of the development organization's sales ability, but it's not a test of usability.

Analysis of Web statistics or logs

This form of evaluation is useful for a site that's already up and running. Put in place a logging and reporting system that, at a minimum, regularly monitors:

- » What visitors are typing into your search field, and the desired results
- » Which queries yield no results.

Your log will not capture the user's intentions – just what he or she does. It's best to use logs and statistics together with other techniques – such as analyzing click through trends, page impressions, points of abandonment, etc. for maximum benefit.

Online surveys and online feedback

Online surveys, or departure surveys, are triggered by a user after completing a process or action. Their biggest advantage is that you can capture user intentions and user satisfaction. The drawback is that respondents are self-selecting; users who are articulate, technology-oriented, and dissatisfied often answer.

However, if the surveys are relevant and well-designed, they will help you to quickly identify problem areas. Repeating survey questions over time will allow you to compare user response to other key performance indicators (such as conversion rate or number of results viewed). By correlating online survey data with changes made to the site and changes in key performance indicators, you will eventually be able to form a clear, factual picture of what is important and what is not.

Eight steps to improve search and usability

The discussion below highlights topics to keep in mind as you design your users' search experience. The ultimate goal is enhancing the search experience through improved usability. It is recommended to follow accepted UI (User Interface) standards – while new and novel UIs may seem better for particular tasks, if it is vastly different than the prevailing standard it will most likely be poorly received.

1. Minimize waiting

Slow system response almost always frustrates users. But, what is "slow?" The answer is relative. If you provide meaningful visual feedback such as a progress bar, it tends to raise the thresholds; users wait a bit longer before getting irritated or thinking about something else.

Users Patience Limit

1/10 second	A response within 1/10 of a second is perceived by the user as instantaneous.
1 second	From 1/10 second up to one second, a lag is noticeable but not problematic. After one second, users will begin to be irritated. The degree of irritation will vary with the individual's temperament - and grow as the lag lengthens.
10 seconds	Loss of attention. The user will stop waiting and start thinking about something else.

2. Make search easy to find

When search is a supporting function on your site rather than the main reason that people visit, you need to ensure that users can locate the search function:

- Use an input field and a search button.
- Put search near the top of the page – usually at the top right or at the top of a left-hand menu area.
- Make sure the search tool has good visual contrast to its surroundings.
- Keep search in the same place throughout the site.

If search is the main reason for your site, you should of course always make the search tool a very prominent element on the home page.

3. Provide different ways to search

Keep in mind that your site should support the user's search activity in many different ways. If you provide more than one way to search the same data, many users will move seamlessly from one form to the other to reach their goals.

You can offer access to search in these ways:

- Provide an input field and a search button. Consider setting the focus so the user's cursor is automatically placed in the search box when the page is loaded.
- Search by browsing and clicking text links. Make sure links look clickable to your users and that the text is concrete and descriptive.
- Text links to search and advanced search.

4. Match advanced search to users' needs and abilities

If you need to have an advanced search function, put a text link to advanced search near the search button. Even though it's called advanced search, you should not expose the Boolean operators "And" and "Or" in your advanced search interface unless your target user group are experts in search. Also avoid the term "string", as in "search string" or "exact string." Instead, use terms such as "Any", "All", and "Exact".

The concept of "progressive disclosure" should also be adopted as the search application matures. This is the unveiling of progressively more complex functionality and capabilities to meet the needs of users as they advance from beginners to expert searchers. The aim is to avoid penalizing users no matter their experience.

5. Optimize results lists for scanning or shopping

Consider your users' tasks and the nature of your content. At a site with a shopping search, for example, items in result lists should be easy to compare, and illustrations are extremely useful. It makes sense to use a tabular format where each result is the same height, with similar information consistently positioned.

At a pay-to-search site or a site that enables knowledge workers, result lists should be easy to scan. The user will be looking for information that characterizes the document. In these situations, it is helpful to:

» Highlight document titles and, if necessary, source the site – but not the whole URL.

» Highlight searched terms.

» Present sufficient content in the teaser to characterize the document.

» Show cues for the smart searcher, such as when the item was last updated.

6. Preserve result credibility

Users expect results in order of relevance. To maintain credibility, you should separate and clearly identify sponsored or featured results – even if you present them first. And ensure that the starting point for "natural" results will be normally visible "above the fold" – that is, without the user needing to scroll down to find this point.

7. Supply a few good result navigators

Navigators help the user maneuver and refine the result set of the search, and need to be present in the individual documents in your searchable index. That means that you must decide which types of navigators to offer before you feed the content into the index.

There are many different ways to allow your users to navigate once they are working within a result set. One thing that's guaranteed: if you show too many navigation and refinement options, you "hide" your functionality from the users. So you'll need to consider carefully which types of navigators are best for your users and your content.

8. Suggest solutions instead of pointing out errors

Give users what they asked for, even if they misspelled their search terms. If you can catch the misspelling, you can offer them a link to the results for the correct spelling. If there are "0 results" for what users asked for, but a spelling correction or synonym gives results, discreetly tell them what happened while giving them the results. If you changed the user's query completely for some business reason of your own, politely say what you did and give the results you want. For example, "We're sorry, we don't have Nike, but you might be interested in these Adidas products." If there is no way to tell what users want, it's helpful to repeat the query to make it easy for them to re-evaluate or edit. You can say what happened without blaming the user, and give them some links to browse with.

The most frequent source of "0 results" is a search that's too sophisticated. Many users submit too many criteria, and some find it difficult to understand that they need to be less specific – not more detailed.

MINI CASE STUDY: Better usability drives revenue growth for mobile phone operator.

WHO: One of the world's largest mobile phone operators.

CHALLENGE: To tailor the user experience for multiple user groups and multiple search clients (i.e. mobile devices with varying screen resolution).

SOLUTION: The user interface conforms automatically to the target device's screen resolution, color depth, and navigation capabilities. Advanced content and query processing is used for extreme precision of search results. The solution cut the number of "clicks to target" from four to two; content browsing was reduced by 50% while content search increased by 100%. This drove a 20% increase in ring-tone revenue – within four weeks of launch.

FAQ

What's a taxonomy?

A categorization or classification of entities based on a predetermined system, with the resulting catalog used to provide a conceptual framework for discussion or analysis. For example, a car manufacturer may have a taxonomy based on the type of car (e.g. convertible, SUV, wagon, etc.).

What's a navigator?

A navigator is a construct that enables filtering and grouping of search results. For an international site, you may have a navigator that enables you to display only results with content in a given language (e.g., "Display English results only").

What's the difference between useful and usable?

A useful system can solve all relevant user tasks, although the solution procedure may not be easy to use. A usable system is easy to use, although it might not solve all relevant user tasks.

How many representative users should I schedule for a usability test?

This is a trade-off between the costs of adding "one more user" versus the benefits of identifying additional pain points with the tested system. Scheduling three to eight users is appropriate in most situations as you will most likely identify more than enough pain points to address.

Integration

Search applications are often integrated into large software applications or complex information systems. These environments are often complex, and there are multiple integration points; careful planning and consideration of user's needs is a good investment to ensure long-term success of search integration.

These days, users take it for granted that every Web site or computer application contains a search function. It may simply be a character string matching feature, such as that commonly included in text editors, or it might be a database-driven field lookup – for example, finding contact addresses from exact names in a Customer Relationship Management (CRM) application.

Simple string matching is no longer satisfactory for most users, given the rise in their familiarity with search and the increase in variable data types (such as Microsoft Office documents, pdfs, Web pages, and e-mail). They expect the full power of search, including fast query response times, dynamic and highlighted teasers, drill-down navigators, advanced linguistic optimizations, searching across many fields at once, and Boolean operators.

That helps explain why many new software products now include search features, and why existing products are upgraded to include advanced retrieval capabilities. The decision for application publishers is whether to use time and resources to internally develop search functionality, or integrate with existing search engines.

Two areas where integration is typical are an authoring or management application such as a *Document Management System*

5 things you should know about integration

1. Integration is the act of embedding third-party software into an application.

2. In-house development of a search engine is not a trivial task!

3. Search engine integration is done with a modular approach, using flexible APIs for content indexing, querying, and administration.

4. Content push provides flexibility and the ability to integrate advanced error handling .

5. A well-integrated search engine can power a lot more than just search.

(DMS), where stored content has to be searchable, and an *industry-specific workflow and investigation tool* (e.g., for law firms or for compliance in the financial services sector) where many external data sources are processed.

If the preferred option is to license software, it is essential to make sure that the integration is carried out effectively. A successful integration will translate to a seamlessly improved search experience within a familiar and well-liked product without compromising the product's security, scalability, or other key features.

To build or to buy

Most companies with in-house development teams will sooner or later face the buy-versus-build decision. They have to take into account the programming language of the existing application, how deep or how comprehensive the integration needs to be, and how much the existing technology will have to be customized.

The advantages of developing a proprietary in-house solution include the complete control of development cycles and commercial flexibility of pricing.

The assessment will also weigh the financial consideration of

My organization is running a sophisticated wiki application. Can a search engine support a suitable document model?

The assumption is that your wiki application contains many entries, each with many comments and updates. These would all need to be indexed as nested entries with their own metadata (such as author and edit date) to allow searching within the scope of a single edit or the whole wiki entry. It's possible to use an XML schema that would support this and a search engine with advanced XML indexing and searching capabilities would allow querying across it.

My application is intended to run on mobile phones. Should I use a thin or thick client?

With mobile phone applications, where bandwidth is at a premium and the possibilities with WML (Wireless Markup Language) are limited, a Java-based client installed on the phone may enable a more intuitive and fast experience. Advanced XML indexing and searching capabilities would allow querying across it.

the time required to build and maintain an internal solution versus the license and royalties specified in an *OEM (original equipment manufacturer)* agreement. So it is important to bear in mind the cost of developing the core engine, as well as the cost of the administrative and configuration utilities, scaling and failover mechanics, documentation, etc.

Since the functionality requirements of search are increasingly costly, the investment required to create an offering that can compete with off-the-shelf solutions has never been higher.

The primary advantages of an OEM search solution are its ability to provide world-class search while minimizing the financial outlay and the associated execution risks and reducing the time-to-market by leveraging tried and tested modules. In general, software vendors will select an OEM solution so they can eliminate development costs and avoid the potential difficulties if users are offering substandard application-centric search capabilities.

Integration points

Despite being off-the-shelf, OEM integration will require substantial planning to provide a seamless assimilation of the two technologies. The component architecture of search engines needs to be understood so that each connection point is identified and treated separately.

There are five main areas to be considered: content aggregation, index configuration, query logic and result processing, user interface design, and administration and configuration. Below is a quick review of each:

Content aggregation

This refers to the process of feeding data to the search engine's ingestion process to create the index.

The first way to feed content into the ingestion process is to use an off-the-shelf connector. Simple connectors are a file system traverser, which monitors directories for new, modified, and deleted documents, a Web crawler which does the same for Web pages, or a database connector which will use Structured Query Language (SQL) to extract structured data and embedded documents. Integrators can also leverage connectors dedicated to repositories, such as Lotus Notes or Documentum. These may be CRM applications, email systems or legacy data stores. Best practice is for the connectors to be preconfigured by the OEM, based on prior knowledge of repository installation types found at customer sites. Alter-

In my CMS application, the data can follow different templates. For instance, if used for a news site, there will be Headline, Byline, Summary, Date, Text, and Picture URL. The same CMS used for a restaurant review Web site will have Name, Chef's Name, Address, Phone Number, Average Price, and Reviews. Do I need to allow my administrators to configure the search index themselves to support this?

One solution which gives flexibility to end users but does not increase complexity of the installation is for the search engine to be preconfigured for a certain number of index, navigator, and numeric fields. Of course, there will be a limit to the number of each type that the end client can use; and the content feeding program will be required to map "Headline" to "IndexField1" for example. On the other hand, it will allow every single customer of the OEM to have the same index configuration, which will greatly simplify management and support.

natively, a full installation of the connector will need to be performed by the end client, the OEM's professional services team, or by an implementation partner, although this can increase the installation and configuration complexity of the application.

Using off-the-shelf connectors allows technology partners to rapidly support many repository types. The other option is for connectors to be developed using the search engine's indexing API (described below). This gives the flexibility to support data sources that may be particular to an industry, or where no standard connectors are available.

A closely coupled content-side integration leverages a *content indexing API* to push data to the search engine. When using such an approach, all connections to the original data store are made by the calling application, which then has complete control over scheduling, interfacing protocols, and data structures. The API also supports error-logging *callbacks*.

Taking advantage of these callbacks, actions may be triggered when documents fail to be processed or do not make it into the index. For example, for auditing purposes a compliance or storage application will keep a report of all documents that did not get indexed.

The indexing integration should include provision for updates and deletions if the system has *dynamic content*. This will be handled by an off-the-shelf connector if used, but must be built in separately when an API document push method is chosen.

Whatever the method chosen, the data is then pushed to the document-processing stage which accepts either text-based (HTML, XML) or binary formats (e.g., Microsoft Office, pdf, or multimedia files), along with the metadata associated with these documents.

Index configuration

Index configuration refers to the configuration and tuning of the search engine. The best search engines provide a *complex and highly tunable search experience*. For example, they offer the ability to weight different fields within a document when searching. Different fields can also be configured for other purposes: querying, sorting, navigators, and range restrictions. Nonetheless, while a search engine is designed to be very flexible (in terms of metadata schemas etc.), a search-enabled application will have a common (fixed) implementation structure. Therefore index configuration decisions should be made upfront by the application designer rather than by allowing end users to choose.

Another key element when designing an index is security. This includes document and attribute level security. Either the application performs a standard search and then filters out the documents that the search user does not have permission to see, or security Access Control Lists (ACLs) are indexed as metadata to allow filtering by the core search engine. The chapter on “Security” details this, as well as all the other aspects of a secure search implementation such as communications encryption and user authentication.

The final crucial aspect of configuration is the need to understand *benchmarking* and the *installation footprint*.

The OEM must consider what type of application profile is being supported – high-volume, high QPS (queries per second), or both – and also understand the key constraints. Is search a significant part of the solution, and will the end clients be prepared to dedicate the appropriate hardware resources to the search application? Or is search just a “nice to have” feature that should have a minimal effect on the application’s installation and hardware footprint?

Query logic and results processing

Query logic and results processing: this addresses the processing of queries and post-processing of results.

Typical decisions involve whether or not to embed Search within an existing page or screen, how to apply navigators for browsing in the result set, and whether to include advanced search options, such as metadata drill-downs or filter boxes.

What if my customers can define their own metadata – how do I set up the index?

For OEMs that build horizontal and highly configurable products, the integration should use dynamic or scoped search capabilities. This way, new fields can be added by the client interface without having to reconfigure the index structure.

Query logic concerns the *query syntax* that is exposed to the end user or that is leveraged by the application behind the scenes, and any transformations between the two. It also relates to *relevancy configuration*, such as whether to weight the freshness of a document in the ranking of hits and any business-specific logic such as synonyms or query disambiguation.

Regarding the processing of results, some OEMs choose to push into the index only the text that will be searched or filtered, and not the metadata that is used for display purposes. One example: the quantity of items in stock for an e-commerce site.

When display information stored in the native application is required, the results must be post-processed, with additional API calls to the source performed. In cases where everything required in the results set is stored in the search engine, the search hit list can be populated directly, with the appropriate formatting applied.

It is important to understand that if one particular field has a higher modification frequency when compared to the rest of the document, it may not be advisable to store the information in the search engine.

User Interface design

User Interface design: this covers how both the query interface and the results presentation are built into the application.

From a UI viewpoint, the planning of a search OEM is very similar to classic search design. Typical decisions involve whether or not to embed search within an existing page or screen, how to apply navigators for browsing in the result set, and whether to include advanced search options, such as metadata drill-downs or filter boxes.

When designing a UI a decision must be made about the nature of the client application. One option is to use a minimal *thin Web-based client* which displays data directly from the search engine. In this scenario, the results are manipulated only via templating. More sophisticated thin clients will parse the hit list returned from the search engine in XML, to then process and format the results. This is typically done in a JSP or a similar server-side technology. A *thick client* is the third option, where a program running on the client computer (for example, as part of the OEM's existing application) processes the results.

Administration and configuration

Administration and configuration: these are the administrative and configuration UIs and APIs (the extent of which will vary from one technology to another).

A key goal is deciding how much of the search engine's configuration and administration is left for the end clients of the application, versus how much is pre-defined when the product is developed.

For features deemed necessary for the client to tweak, the tools to do so must be built into an existing administrative UI, using the search engine's administrative API. In general, though, giving direct access to the search engine's administrative UI is bad practice because it will expose too much to the client, creating confusion, and permitting more modifications than may be appropriate. It will also make each installation more varied and therefore trickier to support.

Complex integrations and best practices

The decision to buy versus build will be based on the need to leverage tried and tested technology, and to keep the company's focus on its core competencies. Therefore, the first key recommendation when embedding search within another application is to investigate the capabilities and APIs provided, to identify which parts are reusable. Here is a quick look at three of those facets:

Content creation

There are two principal types of search OEM. One has applications that deal with data storage, such as a document management, records management, or archiving solution. The other provides a bridge across one or many sources, such as a compliance application (spanning e-mail and a DMS) or a litigation support tool (for example, indexing everything on confiscated hard drives in order to find valuable evidence for use in trials.)

When there is only one repository of data to be considered, content should be aggregated using a file system share or database, or by feeding documents to the content API.

In that example, integration requirements should be minimal. For file systems, a shared folder to which both applications have access is used as a landing area for new documents. The search engine's file traverser monitors and indexes any new documents within that folder. Or, if the native application uses a file system for the internal storage of content, the traverser can be given direct access to the appropriate directories. Mechanisms to add metadata with this simple method are available. Similarly for the database approach: a temporary table can be created with the relevant information (required metadata, path to the binary file, etc.), the connector can be run against the original tables, or a view thereof.

In the second option, the content API has the advantage of error callbacks to track the document throughout its indexing lifecycle. Content pushing allows the OEM to control the indexing flow and lag between document creation and indexing much more tightly. This is critical when each document must be accounted for, or when an action such as a re-try, or an alert is triggered for failed documents.

Using the API approach, both binary data and metadata are pushed together, They are handled by the document processing part of the search engine for conversion into plain text and manipulation. Ready to index text can be submitted in XML format, for example. XML fields can contain paths to documents, enabling hierarchical models with binary data to be indexed. A sophisticated OEM integration may also include complex data structures, such as hierarchical document models, requiring the use of XML for indexing.

Enriching and modifying data can increase the effectiveness of queries. For example, if the application is targeted at law firms and indexes attorney memos and othe legal documents, a custom entity extraction module could be used to tag all legal citations for cross-referencing and navigation. When field manipulation is required, the data can either be pre-treated by the source application before ingestion, or the framework and tools from the search engine's document processing stage can be used.

Data processing is also used to reduce an index's size. If the application is an archiving solution, it may be sufficient to index key metadata, or extracted top terms, rather than the whole text of each document. The removal of duplicate terms, or lemmatization by reduction (each word is reduced to its base form) will also reduce the size of the index.

Index configuration

Search engines are used for a range of applications, with a variety of data and query requirements. In a backup and archiving application, for example, large volumes of data will need to be indexed, and disk and memory usage should be kept to a minimum. An online application searching a large repository, such as a set of scientific or legal documents, will be concerned with scaling in number of documents, in addition to supporting enhanced functionality and a higher rate of QPS. On the other hand, a niche content owner, such as a provider of mobile-phone ringtones, will have a small volume of data but will need to maintain a high QPS. In this scenario, index latency will be unacceptable because it will almost certainly lead to missed sales.

> "GET YOUR FACTS FIRST, AND THEN YOU CAN DISTORT THEM, AS MUCH AS YOU PLEASE."
>
> - *Mark Twain*

This is one of the main areas where the OEM can reduce the configuration complexity of search that is exposed to its customers: by configuring memory settings (capping which parts of the search engine use the most memory), or tuning options that increase or decrease the disk footprint (different field processing and types, such as integer fields, parametric navigators, etc.). Based on an understanding of typical usage, OEMs can configure an application once so it satisfies the needs of most customers.

Query and results processing

One of the most common query transformations required for OEMs is when the application has an existing search feature – for example, home-grown or from another search vendor – and the users are already familiar with the syntax.

More often than not, the new search engine will be able to support the same types of logic as the previous one (e.g., Boolean, nested, proximity queries, etc.) but the rules and syntax will be different. Therefore, care must be taken when writing the translation element.

Another common OEM request stems from the provider having its own source of content over and above the customer's managed content. An example is a news publishing CMS: the technology provider may have a source of news feeds which the client can access as part of the license, or as an upsell option. The recommended solution is to centrally index the common data (the news feeds), using the same search engine technology, albeit configured to support greater query loads. When performing a search, the application installed at each client will fire off two queries. One is against the local search engine, which has local content indexed and the second searches the remote installation in the data center, merging the results locally within the search broker layer. This will minimize the content and installation that is managed at the customer's data center.

Operational integration alignment

Commercial software packages will have best practices of their own, including high-availability strategies, usage patterns, and back-up policies.

A successfully embedded search engine will be installed and configured to be in sync with these application-specific policies. Only under these circumstances will the combined package of the original application and search be able to scale seamlessly and ensure a uniform level of service. For example, a DMS that scales to one billion documents will need a search engine that can do the same; if the search tool cannot, it will be obliged to offer an incomplete solution. Ideally, the coupling will have be executed in such a way that when scaling or backing-up the core data modules, search will automatically and seamlessly follow.

Another area of alignment is operating system and language support. For whatever the application already supports, the search engine must naturally be configured to suit.

Maximizing the search ROI

Good search tools include other features and functions on top of their basic search capabilities. Alerting, browsing by metadata, data storage, "latest news" pages, expert locators, and collaborative filtering are among many possible features that can be added to traditional search.

None of those features uses search in the traditional sense of a box where a user enters a keyword. However, once data is indexed into the search engine, these are quick-win additions that can be used to augment a product. For example:

Alerting: the integration effort has already developed a mechanism pushing each published document through a conversion filter to extract the text. Most search engines provide a module to match this text against predefined rules to trigger alerts to users.

Browsing: if the search engine has indexed drill-down navigator information, this is used to refine hit lists. These drill-downs can be used for a document browsing feature (e.g.; browsing the content in a DMS by author, folder, or file type) which would be powered by parametric searches.

Data storage: if size of application is a concern, one solution is to add search without affecting the installation footprint. This is achieved by replacing the existing text data storage with the search engine itself. That is, whereas previously the text data was stored in a file system or a database and the user interface displayed contents by sending requests to that store, now all data display actions will be queries to the search engine, which holds a unique version of each document.

More features can be used by the OEM to increase the product's appeal, or sold as functional add-ons to grow licensing potential without substantial development work.

OEMs can further increase revenue generation with traditional search *SI (Systems Integrator)* work. As discussed, a key objective is to hide the complexity of search from the customer. Nonetheless, the tuning of relevancy models, the adding of new connectors and data sources, the development of industry- and customer-specific taxonomies or entity dictionaries, and GUI design can all be positive offerings for customers looking for a more complete search experience.

Fundamental steps to seamless integration

The most important part of integration is *deciding which configuration or query features to expose* to the consumers and managers. Quality enterprise search engines are very flexible tools. They are designed to cope with many different types of applications and industries — e-commerce, knowledge management, archiving, video search, etc.

An OEM is typically targeted at a subset of the areas where search engines must compete – for example, a niche applica-

tion within one industry. Therefore, for each decision about design and configuration made by someone installing a search engine (what relevancy model to use, which fields to make searchable, how to tune for optimal indexing and querying speed), *the OEM must decide whether the decision can be generalized to all its clients* or whether the option must be left open for systems integrators or IT administrators to fine-tune the system.

This is a crucial point; after all, the first line of support when something fails is the integrator, not the search engine vendor. By reducing the number of installation permutations, the search engine will become easier to trouble-shoot when customers flag problems.

In addition to the above, common best-practice recommendations for OEMs are to:

- Use a content push method to enable flexibility and error-checking within the indexing side.
- Modify advanced index settings to fine-tune the footprint of the index dependent on the impact that search has on the application.
- Evaluate the pros and cons of using the search engine as a possible store for the data as well as an index.
- Favor a loose integration to begin with if turn-around times are short.
- Consider a deeper assimilation of the technologies later to increase the benefits the partnership.

OEMs should be aware of the potential *revenue opportunities* arising from search - for instance, quick-win add-ons and potential SI-style customization work. These financial gains on top of the increased competitiveness of any product that can perform world-class search, justify the thoughtful planning that must go into ensuring seamless integration of search.

MINI CASE STUDY: Storage provider adds search and chargeback to product line.

WHO: A leading worldwide provider of enterprise storage solutions.

CHALLENGE: To index large data volumes (more than 50 million documents per server) with high- availability configurations, integrated administration, and security.

SOLUTION: Stored data is tagged with hierarchical XML in varying schemas containing document metadata. This XML is pushed into the search engine using the content API. The charging and reporting tool gives administrators a unique birds-eye view of the usage of storage within the enterprise, such as data size usage per department, using analytics on navigator fields.

FAQ

What is integration?

Integration is the joining of two or more software systems at the logical or physical levels.

What is an API?

An Application Programming Interface (API) is a software interface that enables developers to access the features and functions of a hardware or software platform.

What is the difference between a content API and a query API in the context of search engines?

A content- or document-importing API is designed to feed documents and metadata into a search engine. A query API is used to execute queries and obtain a hit list in a usable format.

Security

Search engines are built to provide easy access to available information - but in an enterprise setting, it is essential that this information is delivered in a secure way. Good solutions manage to combine and balance security requirements with search performance and scalability.

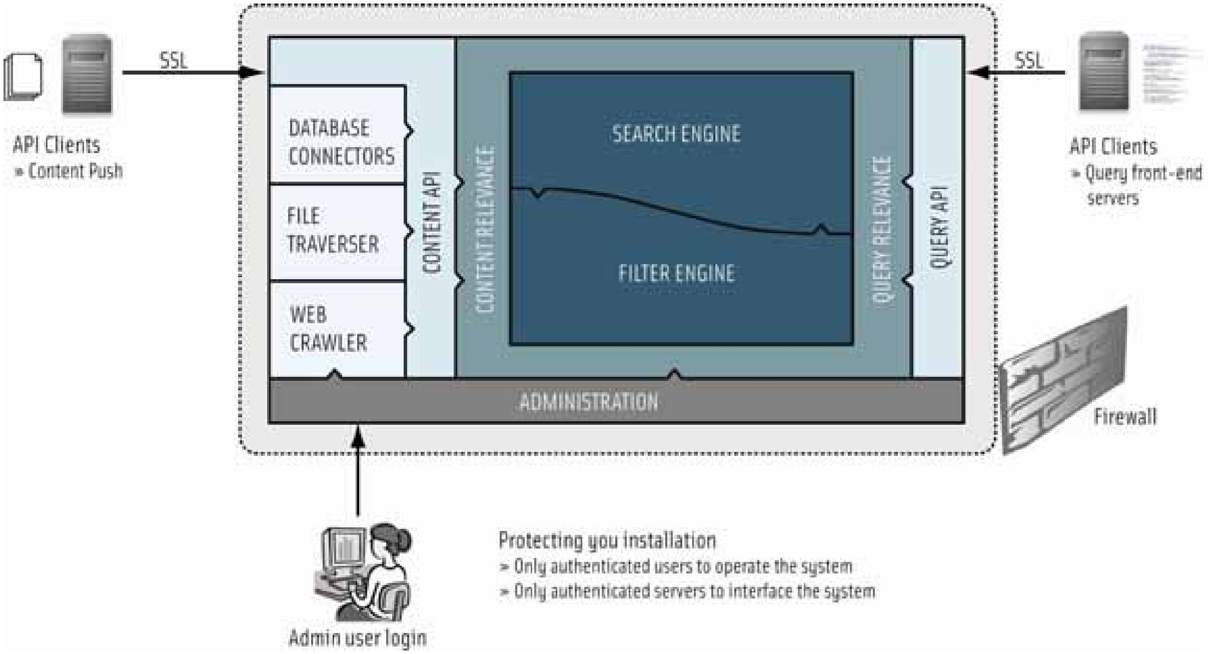
The goal of a search engine is to provide easier and better access to information, whenever and wherever that information may be important. Yet much of that data may be confidential. Although a search engine is a gateway to sensitive corporate data, it also acts as a gatekeeper for such data. So the search engine must be thought of as a *trusted computing base*.

Security is applied to three areas. First, in terms of managing end users, it is used to verify their identities and the levels of content that they're entitled to access. Second, from an application perspective, security validates that all Application Programming Interface (API) calls are issued by authorized clients and that connectors are respecting each repository's correct access model. Third, security is used to manage the authorization control of the administrators who modify the search system itself.

All of these elements have links to search; the most critical is that the permission levels for documents when executing a query are enforced by search software. For example, within an OEM environment, it's crucial to ensure that access via the search interface provides the same level of security as the actual application.

The same challenge occurs when connecting to a third-party application such as a Document Management System (DMS) where the access control model must be reverse-engineered.

Some Elements of Search Security



Protecting you installation
» Only authenticated users to operate the system
» Only authenticated servers to interface the system

The context for document-level security

It is essential to the maintenance and scalability of an organization's IT systems that different employees all use the same IT infrastructure and servers. This may seem obvious, but good security relies on all the systems in use having the ability to segregate and protect data, and giving document access only to those who are entitled to it.

This is particularly sensitive with respect to "Chinese Walls" (barriers between employees to avoid conflicts of interest) and to other confidential data such as employee salaries. Using folder and document-level access control within applications is the most common corporate practice to ensure appropriate privacy. This access control logic must then be respected by other applications connecting to the content, including search engines.

One solution is to use index-based access filtering, where the search engine indexes Access Control List (ACL) information along with each document, and resolves permissions at query time, acting as a sophisticated metadata filter. Users only see documents within a hit list that they have sufficient credentials to view. This is the most scalable solution, but only some search engines provide this functionality.

To begin with, the connector must be able to extract ACL information from the document repository. This information will describe the users and groups that have different access levels – who has read and write access, who has read-only, and who is denied access. In the case of an NT file system, this access information is contained within the operating system metadata, whereas within a DMS, an API call normally allows the data to be extracted for each document. The ACL is then processed and normalized before being added as special metadata within the search index.

The query-side application determines which groups a user belongs to, and group resolution will be performed across multiple repositories if necessary. The search solution must either maintain a cache or have access to an up-to-date mapping of this information. The other solution is to resolve the user-to-group mapping at index time. This is not recommended as the ACLs will potentially grow very large, and each time a group is modified the ACL must be re-indexed. This will cause a large numbers of document updates.

After the above steps (extracting ACLs and resolving group membership) in an index-based strategy, the search engine is responsible for resolving and enforcing the actual access control.

Instead of embedding the access permission information in the search index, an alternative way to handle document-level security is to check entitlement in real-time against the source. For each hit, the search application will check the user's access rights against the repository.

By and large, this solution does not scale since the front end will have to retrieve a potentially huge number of results for each query to find enough authorized documents. This also adds load to the document repository.

> A search solution must be integrated into the security fabric of the organization or of the host IT shop.

An intermediate solution is to combine the two methods. The search core can filter the results, perform a final real-time check to verify that a user's permission has not been downgraded, and filter out documents for which the user has lost privileges. This will increase security in between index updates at the expense of query performance. (If the feeding process and index freshness are correctly configured, this should not be required.)

Location sensitivity is built into some systems. This is modifying security at point of login where the access model for content is modified or overridden based on where and how a user connects to the search environment. For example changing access depending on connection type (public, DMZ, behind firewall, wireless, wired) and device type (workstation, laptop,

5 things you should know about security

1. A system is only as secure as its weakest link. Don't spend time and money on traffic encryption and yet allow staff to leave print-outs of confidential data lying around!

2. Document-level security means that individual documents cannot be accessed by other authorized users of the system.

3. System-wide security means locking down against unauthorized access using encryption, IP filtering, and OS-level security.

4. Index-based ACL mapping is faster and more scalable than post-query filtering.

5. Search engines typically rely on being installed within a secure environment to maintain complete data integrity.

PDA, cell phone). These complex rules must be reflected in the search engine, and may require a hybrid approach.
Collection-level security may also be used. Here, the application tier will assign different authorization levels to various collections within the search index. End users will then have access to the set of collections that map to their authorization levels.

Searching within a secure environment

Document-level security is the most important element of deploying a search engine in a secure installation. Other areas to consider when searching within a secure environment are protecting data access and transfer to and from the search engine.

Assuming that the search engine is returning protected information, the connectors must have access to all secure data and must authenticate to the applications sourcing it as highly privileged users. A wide range of security protocols are in use among all potential sources, although using connectors hides many of the application-specific details such as native APIs. The other option to get content into a search index is for the repository to push data, rather than having connectors pull data. In principle, this does not change the requirements for authentication between the application and connector, although the specific details change.

Server-level security is required in order to protect the integrity of the trusted computing base itself; it can be accomplished using firewalls, and by ensuring that all traffic, within and through the firewall (query, content feed, and administrative access) uses appropriate encryption and security protocols.

In addition, user authentication must be enforced to ensure that only authorized individuals are granted access. Typically, the search interface is embedded within an application or portal that performs this authentication. One set of authorizations govern search user access to collections and documents. Another set of authorizations governs administrative user access to various administrative functions, although the issues of user authentication and authorization are much the same. It's recommended that even if query access is not restricted (such as in an e-commerce or Web search scenario), authentication always applies to the administration console.

Search security must work in conjunction with other IT security mechanisms and policies.

Designing a secure system

Security is a complex topic. When designing a secure system, the first rule is to plan ahead, to understand what the users really need and what the corporate IT infrastructure can realistically accomplish. For example, a single sign-on (SSO) solution is generally recommended for knowledge management projects, or where the search index spans multiple authorization-controlled repositories. However, if the merging and unifying of authentication and usernames across all repositories is not already in place, the search roll-out may be delayed if it is waiting for an SSO implementation.

Integrating the search system with the corporate central security directory (i.e. Microsoft ADS, LDAP or Netegrity) is also recommended for seamless and secure document access.

Collection-level security can be used when there is a division

I am integrating search into my secure application. The push mechanism for indexing is using 128-bit encryption to secure the data. Is that safe enough?

N-bit (such as 128-bit) encryption means that the key required to decipher the encoded data has a length of N. For brute-forces attacks, N is a measure of how difficult an encryption algorithm is to crack, since it determines how many permutations must be tried in order to find the correct key. For example, even if a computer could test one trillion keys a second, it would take two million years to decipher a 128-bit key. It is assumed that for at least the next 10 years, 128-bit encryption is virtually unbreakable.

of data without much granularity. It is useful for departmental separation of information in a company, or when the underlying repository has no security. For instance, there may be two servers for shared documents, one for marketing and the other for finance, where the separation is enforced by common usage rather than Active Directory. Collection-based security could be chosen to enforce this practice.

I have Exchange and Documentum with NT usernames and Lotus Notes which use their own conventions. Can I still do secure search in one pass across all three sources?

Assuming an index-based security solution, each document contains an ACL which will contain users and groups in the repositories' own formats. The tricky part is for the front-end application to determine the Domino and NT names for a user once he or she has been authenticated. This information will typically be stored within an LDAP-compliant directory or a SSO product. Once that information has been obtained, the front end can find out what groups the user belongs to in each domain and then send all that information to the end user, together with the query text.

Choosing the correct document security model

Once the security model of the application has been determined, the focus becomes the correct replication of all underlying access control mechanisms.

When reverse-engineering of access control models is not possible or is undesirable because of the system's complexity, or an index-based solution is unacceptable due to the security latency, the best recommendation is to perform post-filtering of the search results against the source to remove unauthorized documents. For instance, dynamic access managers (using the time of day, the client's IP address, or the strength of the authentication mechanism to govern access to data) require degrees of sophistication that may be prohibitive to ACL mapping.

However, with this approach, when a user requests N hits, the filter will often need to request much more to account for those that will be removed. In particular, as the number of documents in the index grows and the ratio of documents that each person can see diminishes, query performance will drop. Additionally, even for small indices, performance is often worse because of the time needed to submit requests to the source. Therefore, the index-based approach to document security is more scalable.

If there is no way a connector can extract access control information on a per-document basis (for example, if it is not supported by the underlying API) it will be necessary to fall back to a query-time filtering approach. Caching is then used to increase the speed of searching. Generally, this is worth the extra engineering effort because a user will often search for related themes where the same documents appear in hit lists or repeat the search to review the results. The benefits of caching user-to-document matches will therefore be apparent.

If the preferred approach of index-based authorization is used, there should be no issues for the integrator or search development team in the case of a connector purchased with built-in document-level security. On the other hand, for custom connectors and push mechanisms, where the application developer is responsible for designing the security, it's important to scrutinize the ACL model. There will inevitably be caveats and individualities from one source to another which must be studied for both the query logic and the ACL creation at document processing time.

Results granularity need also be considered, where the document is not the atomic unit of security. A proper security model should support property-level security, i.e. which fields can be viewed.

The final element to the index-based method is building the user-to-group mapping. This can be cached within the application session to avoid slowing down queries. If a further update latency is acceptable in exchange for faster login times, an external cache can be updated on a schedule.

The downside of index-based mapping is the lag between an ACL being updated and the search index being notified. This is usually acceptable, as the lag period will actually be very short. In addition, the false positives returned will be documents that the searcher was able to view during the last update cycle.

However, some system managers feel that this represents a security hole. It may be unacceptable for users whose document-access privileges have been revoked to see a document's title and teaser within a hit list. In such circumstances, a mixed approach is recommended, with a last-minute check performed in real-time. This approach benefits from some of the scaling advantages of the mapped ACL solution along with the real-time validation of a post-query filter.

The fundamental steps for improving search

There are two key takeaways regarding search and security. First, index-based ACL resolution is faster and more scalable than post-query results filtering. Second, search engines are only as secure as the firewall and the user authorization infrastructure they reside behind.

Developers sometimes attempt to break a search engine's security mechanisms, since the time taken to crack a system is seen as a measure of its integrity.

In fact, an IT manager's main concern will be the security surrounding the search engine. For the engine to function properly, it requires only a very limited amount of access from third-party applications. Therefore, the way to secure a search engine is to deny access to it, which entails:

- Putting all search engine components on the same network behind the same security infrastructure.
- Securing the hosted systems by IP address and port.
- Encrypting communication between the calling application and the query server.

In this way, the search engine simply becomes another server that needs to be protected, but will not open any security holes in the company's IT infrastructure.

When designing secure search, the principal goal is that correct document level security must be ensured, but it must be imperceptible from a performance and scalability perspective. It needs to meet these criteria:

- The search provider must feel confident that no information will leak.
- Users need to be assured that no unauthorized people will see their data.
- IT managers must be able to support search for multiple user groups within the same IT infrastructure.
- Business managers need assurance that the search technology used will enforce the corporate security policies.

With a well-designed and well-planned index-based security model, all of these criteria are attainable without compromise to search speed or scalability. That way, every stakeholder can feel confident about the integrity of the search application.

MINI CASE STUDY: Entertainment portal rolls out scalable secure search.

WHO: Italy's largest information and entertainment portal.

CHALLENGE: To allow secure search across a wealth of data from many sources, including 4 million intranet documents and 6 million crawled documents with over 20,000 user sessions per hour.

SOLUTION: Multiple connectors (Lotus Notes, File Traverser) all indexing ACL information from the source, with connector surveillance to monitor for ACL updates that can be quickly pushed to the search index. In the front end, a Single Sign-On is used for document retrieval. The application also links to both Active Directory and Lotus Notes for user-to-group resolution.

FAQ

What is AD?

Active Directory (AD) is a Microsoft service that identifies all resources on a network and makes them accessible to users and applications.

What is LDAP?

The Lightweight Directory Access Protocol (LDAP) is a set of protocols for accessing information directories. AD is an example of an LDAP-compliant directory

What are an ACL and a DACL?

An Access Control List (ACL) is a set of data that tells an operating system or application what access rights each user has for an object. A DACL (Discretionary ACL) is a user-controlled ACL.

What does encryption do?

An encryption algorithm modifies data so that it's unreadable to applications except those for which it is intended. Decoding the information requires knowledge of the algorithm and either one or two (public and private) keys.

Can an index-based ACL solution show users documents that they don't have permission to see?

No. In certain circumstances users may see the title and teaser of a document they previously had permission to see (their permission may since have been removed.) They will not be able to see the actual document, though, since the underlying application will perform its own authorization check when requesting the original.

What are Chinese Walls?

Chinese Walls are the internal policies put in place, typically in financial organizations, to restrict communication between different teams within the same company. The goal is to segregate knowledge transfer where a conflict of interest is possible, for example, between teams working on different sides of a same deal.

High Performance Search

Time really is money! Any lengthy wait for search results to appear on the screen is time that could be spent doing something more productive. Out-dated information can also be costly, even if it's only a matter of a few seconds' delay: just ask any stockbroker or emergency room doctor.

Users who have enjoyed millisecond response times when searching large bodies of data no longer tolerate slow queries or large indexing latency. However, providing a scalable, high-performance search offering is predictably not simply a matter of choosing adequate software. User satisfaction comes from having the correct software and hardware configuration based on the most important performance and fault tolerance requirements.

A search service that meets those requirements typically relies on identifying key metrics and on specifying the hardware and software appropriately. From a performance point of view, the metrics include the total number of documents to be indexed, the required ingestion rate, the acceptable indexing latency (the time between a document addition or change and the index being updated), the number of queries per second (QPS) that can be handled, and the target end-to-end average response time. A Web search engine will primarily be concerned with handling a high QPS rate; an archiving solution, which may be indexing billions of documents, is most concerned with minimizing the number of servers required to contain the index and maximizing the ingestion rates when bulk loads of documents are added. In this chapter, we will examine the characteristics of high-performance search in more detail.

The three dimensions of scaling

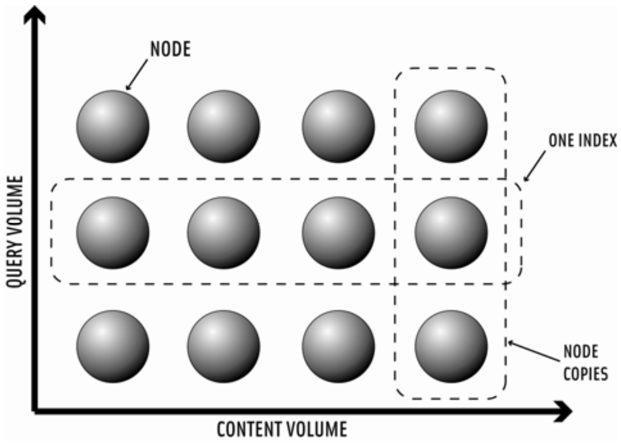
High-performance search grids can scale along three dimensions: document volume, QPS, and indexing speed and latency. Scaling by volume is achieved through distribution and QPS through replication; the third dimension, document indexing, scales with the resources allocated to content aggregation and processing. In a well-designed system using leading-edge technology, all three dimensions should be able to scale linearly, independently and simultaneously. That way, the desired performance targets can be achieved along each of the three axes within the most cost-efficient architecture. Below is a review on how those three areas are considered when designing a high performance system.

Document Volume

In a typical large-scale system with one logical index, the actual index may be shared across multiple machines, where each segment of the content is contained within one column (see diagram).

Imagine a situation in which one terabyte of raw data translates to 300 million records of varying size. Based on the required functionality and the search technology chosen, it may be decided that up to 40 million documents would be indexed into each node. So the full index will be spread across eight columns, with each column containing 37.5 million documents. This decision is driven by the scaling of the search engine – that is, the point at which the search engine's ability to perform queries within the desired average response time range drops with the increase in number of documents. This tipping point varies with the search engine technology, the average document size, desired freshness, and the available hardware (especially memory and disk space). It will also be a factor of the index features chosen, such as lemmatization, navigators, and additional metadata fields.

Multi Node Search Installation



Once a strategy has been decided, it's necessary to deploy software components for the division of data and the merging of results so that one logical index can be broken into many nodes. Distribution components allow the documents being ingested to be evenly spread across the nodes of the matrix. Typically, the algorithm for distribution will be a simple randomizing function, statistically ensuring an even distribution across the nodes. It's preferable to consult with search experts if another distribution model is used (based on creation date or source, for example) or if the number of documents grows substantially beyond original expectations.

A query dispatching module will broker each query received to all the nodes of a row, and merge the results set to provide one single hit list to the end user application, regardless of how many underlying nodes were present. Therefore neither the user interface (UI) designer nor the end user needs to see the complexity of the underlying matrix.

Query Volume

Scaling to high query volumes is guaranteed by scaling the number of rows. That is, if all the documents in a search index are contained within M instances of the search engine, these M servers are said to represent one row. To increase either query performance or fault tolerance, more rows are added as duplicates of the primary row. Commercial load-balancers are normally used to balance queries between the various rows, since no logic specific to the search engine is required here.

Assuming that the hardware is identical among all nodes, the scaling of QPS is near-linear with respect to the number of rows since there are no interactions or co-dependencies between them. If N rows are needed for a target QPS number, $N+1$ will give a fault tolerance, allowing one row to fail while maintaining the target QPS. $2xN$ will support double the query volumes with no drop in search performance.

The above example has explained how to scale QPS and query response times from the search engine's standpoint. In fact, there is a critical difference between the perceived query time and the search engine's time (i.e., "Results 1-10 of about 10,000 ($0.XX$ seconds)" from a Web search). The end user measures the total query throughput – the elapsed time from when he or she enters the query to when the results appear. This period obviously includes the time spent in the search engine, which in turn is affected by the size and complexity of queries, any query transformations, the number and complexity of navigators used, document-level security, the type and levels of sorting, the linguistic features applied, and of course the core technology chosen.

Total query response times, on the other hand, are also affected by pre- and post-processing of the query external to the search engine, by federated searches, and by network latency. Network latency is more significant when the UI displays images such as results thumbnails for an image or video search, and when large quantities of hits are requested. The performance of the thin-client Web server and the application server that hosts the UI also have to be optimized if bottlenecks are to be avoided.

5 things you should know about performance

1. Search engine performance is measured in terms of document ingestion rates, QPS (queries per second), average response time and index freshness.

2. User perceived speed is not just the core search engine query time but by the time it takes for results to be displayed after the user hits "Search."

3. The main point of the sizing exercise is to determine the optimal number of documents per search node for a given set of performance criteria.

4. Performance design is often a trade-off between hardware costs and the end user experience.

5. By identifying bottlenecks, cost-efficient incremental improvements can be made.

It's important to note that in terms of scaling with query load and document volume, the calling UI will traditionally only be aware of the one logical search index no matter how the underlying server matrix is composed. This facilitates application design independent of index-scaling complexity.

Ingestion rates

From a content ingestion perspective, performance is measured by document volume (total size), ingestion rate (documents per second), and latency (total time taken from the entry point of the document in the system to it becoming searchable). These values are mostly affected by the average size of a document, by the format's complexity (for example, large pdf files are more processor-intensive than simple HTML files), and by the degree to which linguistics and other document transformations are applied during document processing.

Document importing is normally a sequential process. So higher ingestion rates are typically achieved by increasing the

WHAT COMPONENTS have to be balanced when considering performance? Consider your **needs** in terms of document volumes, ingestion rates, QPS averages and peaks, and weigh those needs against hardware number of document processing pipelines which simultaneously parse documents. This is also dependent on the availability of CPU and memory, which tend to be the limiting factors within the content processing sub-system. Once a server is at full capacity, the search provider can deliver faster throughputs by adding processing stages to additional servers. This will only bring improvements if document processing is the limiting factor within the content chain, not the actual indexing or the access to the source repository. Therefore it's also valuable to consider the connector's configuration and the throughput from external sources.

It's possible to tune an index's refresh interval. This is the time between a document finishing its processing stage and its being added to the searchable index, often called index freshness. Resource usage increases as more documents are added to the index, impacting query performance (notably by causing more disk activity). Therefore, a balance must be achieved between the target freshness required and search performance. For instance, a news search will desire freshness near real-time, whereas a corporate knowledge management search may be satisfied with something slower. In larger installations, to avoid this co-dependence, index and search nodes should be run on separate servers.

Generally, search applications are configured so that there's enough processing capacity in the document processing stage. But sometimes the data is not being fed in fast enough. This can be due to the scheduling of the connector, where it might not be aggressive enough, or because requests to the repository itself are slow. For example, in some large organizations the IT infrastructure is distributed so that content sources reside is multiple distinct locations. In order to create a centralized search index, the connectors must access the data across the WAN (Wide Area Network) where download speeds may be slow.

Designing a high performance system

Search providers do not all share the same needs in terms of document, query, and freshness metrics. For instance, providers of news and financial search place a premium on freshness, whereas litigation support services, receiving data in a batch and only indexing once, are more concerned with ingestion rates. The key to optimizing a system is to be very clear about its key business objectives.

The cost of hardware will to a large extent determine whether the objectives are realistic. By choosing a search technology that supports a high QPS rate across a large number of documents per node, it's easier to reduce the total cost of ownership (*TCO*) of the search function. The search numbers will vary from less than half a million documents to more than 50 million documents per node, and from less than one to over 100 queries per second, depending on which search engine technology and configuration are used. There will always be tradeoffs between cost, ingestion, and query performance.
The first step to minimizing hardware costs is to be realistic about document volumes and expected queries. IT managers are often guilty of overestimating the numbers of search users and the volume of documents to be indexed.

We are implementing search over our document management system for the first time. How do we estimate the system's expected query load?

If there is no empirical data to go on, user behaviour will have to be estimated. The first question is how many employees have access to the document management system (DMS)? That estimate must then be divided into different subsets depending on their DMS usage: for investigative purposes, for regular information purposes; for information purposes, but on an irregular basis, or to store personal documents, for instance.

The next step is to estimate an average number of queries per day for each user subset, from maybe one per day for irregular users to 20 per day for those with aggressive information needs. It can then be assumed that 60% of the queries will occur during the peak hour, say. This calculation will give a peak QPS figure which should be used along with the estimated document volume when sizing the system.

If a search solution is already in place, query usage logs from that system will be a good indication of expected future usage. QPS peaks are the key here, since searches are never uniform over time and the most positive user experiences will come from searches that perform consistently even when usage is heavy.

If there's no search system in use, the search patterns have to be estimated, beginning with questions such as: How many users are expected? How many searches will be performed on average per session? Are users geographically dispersed so that system loads can be spread throughout the day? When sizing the system, it's also important to think about all the possible actions that can trigger a query. If drill-downs and navigator browsing are populated and driven by the search engine, for example, or if stored searches are executed by certain pages, they all count as queries to the server so they all add load.

When evaluating the data volumes that the system should be able to handle, the key metric to consider is the amount of raw text that is extracted. For example, one terabyte of data on disk can represent different amounts of text depending on the format used. If all the documents are pdfs or high-resolution images and videos, where only their metadata is being indexed, the quantity of text generated will be much smaller than if the terabyte of disk space was taken up with e-mails.

Average binary-to-text ratios are available for common formats. In the case of a knowledge management solution, for instance, the administrator can determine the proportion of content in each format from the solution itself and make an approximate calculation of raw text accordingly. In other situations, empirical data is normally used to estimate the index size.

Knowing the predicted QPS and text data volumes, the sizing exercise can then calculate the optimal number of servers. The search engine experts will usually do this, although the integration development team can also benchmark the query performance of the engine for a given hardware specification, tracking various QPS and data measures to determine the "sweet spot" for the number of documents per server.

My e-commerce site currently has one search row capable of handling 100 QPS. We're planning a large marketing drive that is expected to increase traffic significantly. What steps should I take to make the system more powerful and more robust?

A second row would allow the system to handle 200 QPS with no redundancy, or 100 QPS with failover. Three rows will handle 200 QPS with failover: if one row fails, there will still be two live rows capable of handling 200 QPS. The decision is whether, should a failure occur during peak times, the average response time is allowed to degrade or not. It's a trade-off between hardware cost and desired service levels.

It is important to remember that the optimal number of documents per server will depend on the search application itself. If longer response times can be tolerated to save on hardware costs, then more documents can be indexed per server. For example in an archive search which is infrequently used, it may be acceptable to have a query latency of seconds or minutes (rather than milliseconds) to save hardware costs.

When assessing the required hardware specifications, another factor to be taken into account is the nature of document turnover. How fresh must the data be? Is the data dynamic, with updates and deletions occurring regularly? Or is it static - never modified until expiry? Let's take the example of a static data set - a compliance or an archiving solution, say, where new documents are added but existing ones are never modified or deleted. In this scenario, the engine can be tuned to allow for a one-off ingestion followed by a static phase where only queries are received. This scenario enables optimizations to maintain QPS levels equivalent to those for a standard index, but with a less costly hardware specification.

> The optimal number of documents per server will depend on the search application.

To support this model along with a low indexing latency constraint, a dynamic index must be implemented for new documents. For example, in the case of a mail search where e-mails must be searchable as soon as possible after they arrive, the best solution is a hybrid of a dynamic index with static indices for the old e-mails.

Additionally, there is the notion of dormant indices, which can have a massive number of documents – hundreds of millions – but where queries times are longer. This is useful for an application where the query load is small, and longer query response times are acceptable; it allows for a very large index with minimal hardware costs.

Optimizing search performance

Search engines are designed to be flexible applications, to be used in many different scenarios and configurations. Therefore, given that each installation is tackling a specific use case, steps can be taken to optimize the search function.

Streamlined processing can optimize document parsing. For example, performance can get a lift when the exact index and

My e-mail compliance solution features a search engine. One of our customers is an organization with over 100,000 employees. There may be as many as 10 million e-mails per day (and growing), some of which will contain attachments. Assuming we have 13 million documents per day, and we store e-mails for three years, we'll have approximately ten billion e-mails to index. How should I design my index?

Although there may be 10 billion documents to index, searches will only be performed if a compliance investigation takes place – hopefully an infrequent occurrence for your organization. So the best solution is an index configuration that's optimized for low query volume and high data volumes, not for document updates and deletions.

query requirements are analyzed so that any superfluous document processing stages are removed. Performance can also often be improved by increasing the size of the batches submitted.

Total system throughput is boosted by reducing latency with the right mix of hardware components. For example, more memory, high-speed hard disk drives, and high-speed networks between the search system components will all contribute to faster overall response times. In particular, disk performance is the single most important hardware factor in determining the overall performance of a search cluster. Striped RAID systems with large stripe sizes are best. When considering storage area network (SAN) disk drives, it is important to take into account the combined disk speed requirements of all search nodes pointing to the SAN, and to design the I/O specification appropriately. For some systems, data freshness may be driven by the high update frequency of one particular field. Think about an e-commerce application for a moment: the price and stock count of items must be kept up-to-date because they change rapidly, whereas item categories and descriptions are relatively constant. In this instance, the best solution may be to off-load query and content storage to a search engine that has been optimized for that type of operation. A database is used in parallel for the frequently updated fields since it will support transactions with ACID write semantics.

Optimizing search speeds is a matter of always executing the best query. For example, overriding the default API flags and toggling query settings that cause unnecessary extra work for the search engine. That raises the issue of the trade-off between the search user's ways of working and the IT manager's design parameters. On the one extreme, a search that only ever returns five documents sorted by relevance from one or two word queries, with no filters or navigators, will always run fast on minimal hardware, allowing IT managers to meet their targets. On the other extreme, a fully fledged interface with numerous navigators, thesauri and spell-checking, long queries, complex wildcards, and so on, will give the users much more flexibility but will require more hardware to enable the same response times and QPS.

In practice, the feature set design will be driven by the business owner. However, owners must take into account the scaling trade-offs of different features. Once a set of functionality is defined, tuning is required to optimize performance for the given query profile.

Poor tuning of the search index can have a perceptible negative impact on the user experience. So it is almost always helpful to use subject matter experts to size and design the search solution so that it strikes the right balance between speed, size, and cost. It is then possible to tune the index configuration and query parameters optimally based upon the expected queries.

Performance tuning should not be seen as a one-off exercise. The types of queries that users are submitting may vary over time compared to the pre-launch predictions. Also, the actual bottlenecks within the system may not have been correctly identified. Benchmarking of the live system can identify choke points and assist in the continual improvement of the search experience.

The fundamental steps to improve search

Scalability is a function of the search engine technology used. A successful implementation of course calls for selecting the appropriate software but it also requires a clear definition of the performance metrics that the business demands from search. It is important to calculate the TCO for those criteria and for the architecture that's best suited to the business and technology environment. To develop the most cost-efficient system, the solution should be designed to scale independently and linearly based on query volume, document volume, and ingestion rate.

The key performance indicators (KPIs) involved when specifying the search implementation are:

» Document ingestion rate.

» Total document volume.

» Document churn (frequency of updates and deletions).

» Desired freshness.

» Expected QPS.

» Total average response time, and response time demanded from the core engine.

» Total cost.

Once the implementation has been specified, the search provider then carries out a sizing exercise, taking into account the features used by the UI, such as linguistics, navigators, and wildcards. (The exercise assumes typical selections of hardware, although this is of course another variable that can be used to tweak estimates.) This exercise will derive an optimal number of documents per search node to support the freshness and average response time metrics identified. Based on these calculations, the search provider can be sure that the number of documents per server will be able to sustain a given QPS.

The final stage of the search implementation will result in a matrix whose dimensions are determined by scaling the number of columns from *X*" to the total data expected and the number of rows from *Y* to the desired QPS. If failover is demanded, rows will be added based on the redundancy obligations.

Once the sizing exercise is complete, there is still additional scope for improvement. Then the objective is to maximize the utilization of all hardware against the performance objectives. The areas to consider here are:

» Optimize disk performance.

» Trim the document processing stages.

Remember: System dimensioning is an **iterative** process. There are many (at least 50) possible parameters that determine your system's dimensions and these change over time, as your users and content evolve with improvements to your search application.

Example parameters are:

» Maximum or average content update rates and latency

» Maximum or average query rate and latency

» Content complexity and volume.

- Analyze the queries to tune the system for them.
- Tune the index based upon the freshness and the static or dynamic nature of content.
- Leveraging solutions services from the search provider, and benchmarking normal operational cycles of the end-to-end search process.
- Balancing the need for speed, size, and cost.

MINI CASE STUDY: E-commerce site handles 500 updates and up to 1,200 queries per second.

WHO: Japan's largest online shopping mall.

CHALLENGE: To handle high QPS rates and freshness over six million items from 14,000 e-tailers .

SOLUTION: By employing a distributed system across 60 nodes, 500 updates per second are handled with a freshness of 90 seconds. With the improved architecture and search features in place, consumer queries have increased by 200%, peaking at 1,200 queries per second while reducing TCO by 67% compared to the solution used previously by the mall.

FAQ

What is meant by content volume?

Content volume is the total size of all ingested documents. This is normally measured in terms of gigabytes of text data. If the volume is measured in terms of the number of documents, it is important to have a gauge of the average document size.

What is the ingestion rate?

The ingestion rate is the number of documents per unit of time processed by the document processing stage. This is dependent on the hardware, on the format of documents, and on the number of data transformations (lemmatization, entity extraction, etc.) being performed.

What is total index latency?

Total index latency is the time between the discovery of a document by a connector or its being pushed using the content API and the point at which the document is searchable.

What is query volume?

Query volume is the total number of queries served over a fixed period of time. It is typically measured in terms of queries per second (QPS).

High Availability Search

Enterprise search is mission critical for many enterprises - for example, an eCommerce seller or a criminal investigator cannot afford for their search boxes to be out of order. In face of inevitable hardware failures, search systems use carefully architected redundancy for constant availability. Search providers should differentiate between availability on content ingestion side and on user query side.

Critical IT systems are often described as fault-tolerant, redundant, or displaying high availability. That is, should something go wrong (for example, the office cleaners pulling the plug on the servers by mistake) the systems have been designed to maintain a certain level of service.

The standard solution is to purchase more hardware in order to mirror vital elements of the system. The downside of this simple solution is the price of the equipment and the internal cost of managing a duplicate system. Consequently, the extra expense of redundancy is tallied against lost revenue from downtime and the odds of certain fatal errors occurring in the different parts of the system.

For example, an e-commerce site will know its sales distribution for one day, and can estimate the potential sales losses for downtime during peak hours. In the same way, a pay-per-click search service can measure the loss of revenue associated with downtime. In addition, the search provider must evaluate the damage to user loyalty from an unsuccessful search experience, and the cost of rebuilding the index.

Equally important are content indexing and searching. In an e -commerce application, delaying the addition of new products is less costly than the loss of querying. But for a military intelligence or financial trading application, having out-of-date information may be worse than having no information at all.

5 things you should know about high-availability search

1. Computers fail for many reasons: hardware, software, power, connectivity, etc.

2. Service degradation can be abrupt or graceful

3. High availability is ensured with redundant systems

4. Different parts of a system can have varying degrees of fault tolerance

5. Downtime costs money, but so does redundancy!

This chapter will tackle the different ways of creating a highly available search system, and the different scenarios that search providers must cater to.

Tackling failures to enhance search

Backbone failures

It's important to decide which eventualities to plan for when specifying a redundant search infrastructure.

Network outages require two separate *ISPs* (Internet service providers) with independent cabling for correct fault tolerance. Short power failures can be avoided with *UPS* (Uninterruptible power supply); longer ones with back-up power generators. Users can also pull in power from different parts of the power grid for extra redundancy. Dependent on the location and sensitivity of the system, physical attacks can be protected against with various security measures (locked doors, armed guards, etc.) and robustness in building construction can be considered to protect against natural phenomena such as earthquakes.

When considering independent data centers, it's important to weigh the administrative and maintenance effort of running two separate systems (an administration team in each location, large amounts of data-traffic between locations, for example). Two inexpensive data centers in distinct locations may be cheaper than one fully redundant and secure one. On the downside, because the two installations would be fully independent, data synchronization is not guaranteed, which can affect some applications.

> The extent of built-in redundancy required needs to be measured against the opportunity cost of a failure.

Component and service failures

Companies typically have a corporate or service-wide policy regarding power supply and data center security, and search will likely conform to this policy. But search services will also have specific high-availability considerations, requiring hardware redundancy combined with intelligent recovery operations to ensure search uptime.

Search application services to consider are:
Content aggregation and processing. This refers to the crawling or capture of data from source repositories and any pre-indexing transformation. Failure would interrupt the continual flow of new or updated information into the search index.

Search engine. The search engine is where the actual queries are assessed and documents returned. No search is possible without it; although in a system distributed over multiple nodes, a partial service may be available if some nodes are still alive (searches will of course be against an incomplete index).

Search broker. This is where the merging of content from multiple search nodes and any query or results processing are performed, as well as the federating of queries. No search is possible without this.

User interface. The application with which users interact with the search engine. No searches are possible without it. The presentation layer infrastructure (Web servers, Java/.NET environment, etc.) is typically independent of the search technology provider and in certain situations it is acceptable for the user interface to be available while search is offline.

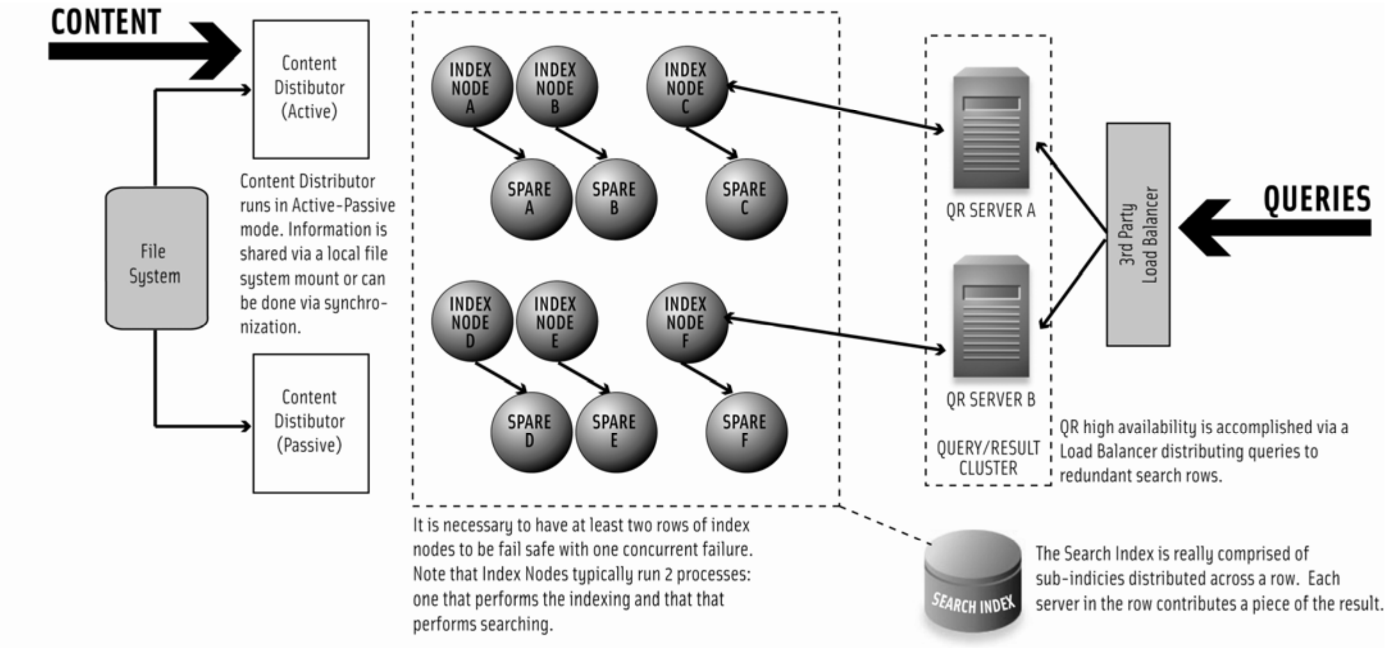
Administration and management tools. Search services will run without such tools, but the ability to monitor for fatal failures will be reduced.

Factors that affect high availability

Before implementing a search solution, we recommend that users investigate:

- » Whether a particular component should have some fault tolerance.
- » What procedures should be in place for rapid restoration of the system.
- » Whether the service should have a live and failover node, or a redundant system with graceful degradation.

Elements of High Availability of Search



Content Distributor runs in Active-Passive mode. Information is shared via a local file system mount or can be done via synchronization.

It is necessary to have at least two rows of index nodes to be fail safe with one concurrent failure. Note that Index Nodes typically run 2 processes: one that performs the indexing and that that performs searching.

QR high availability is accomplished via a Load Balancer distributing queries to redundant search rows.

The Search Index is really comprised of sub-indicies distributed across a row. Each server in the row contributes a piece of the result.

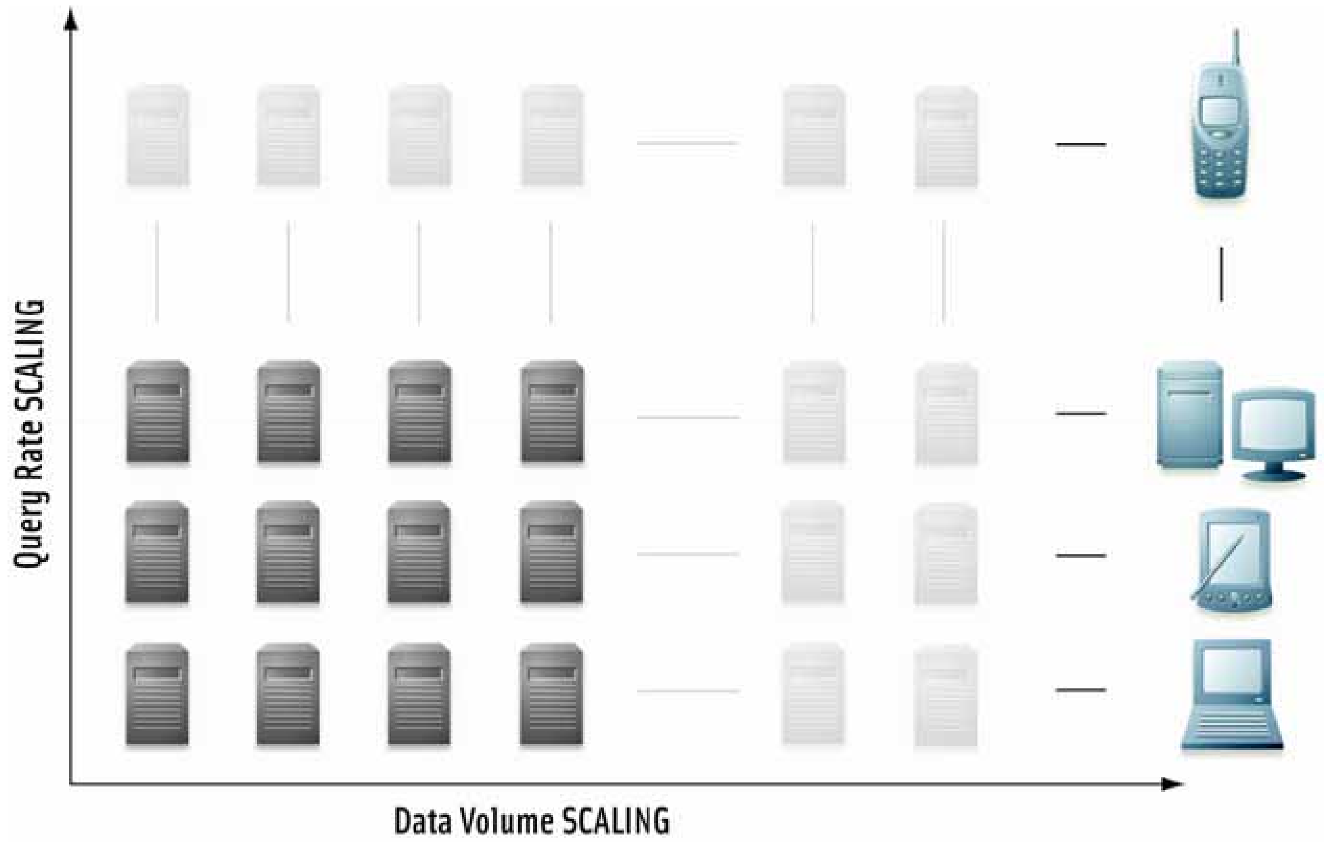
Content aggregation and processing: A second set of all relevant components is necessary to offer high availability of content ingestion into the search engine. For the content distributors and document processors, parallel installations can either run continuously or in hot fail-over mode. Connectors, which contain state information, typically deliver redundancy with one installation in active mode and a second in passive mode. To achieve this, the state of the connector (for instance, which documents have been indexed, when the last full batch was completed, etc.) can be stored in a shared environment displaying high fault tolerance, such as a *SAN* (storage area network) or database.

Importantly, although connectors may display high availability, both set-ups will run against the same data source. Therefore, guaranteeing the uptime of data feeds requires administrators to adopt policies and redundancy to ensure the same level of service from the underlying repository.

It is also considered best practice to use separate network infrastructure for the content processing and search engine.

Search engine: Fault tolerance is guaranteed by duplicating a *row*. Typically an actual index in a large system will be shared across multiple machines in a grid configuration, where each segment of the content is contained within one *column* of servers. Smaller systems may have only one column. To increase query performance or fault tolerance, more *rows* are added, containing duplicates of the index.

Two Dimensions of Scalable Search



These spare nodes can be set up as backup servers in an active/passive configuration. They can also all be active, allowing a graceful degradation in the maximum query throughput that the system supports.

Is it necessary for a corporate intranet search to have full failover?

Not if there is no immediate financial loss incurred by an outage; the cost of equipment duplication will be prohibitive. It pays to look at the time to rebuild an index. If this is longer than the accepted downtime, we recommend either having a regular back-up of the index or at the very least a passive back-up on a spare machine.

Generally, it is best to share the load across all servers and not to use active/passive failover. This improves availability by reducing the average load on one server and thus improving mean time between failures. Also, a server may sometimes fail when it experiences large changes in load, which makes active/passive configurations somewhat dangerous. However, an active/passive set-up is appropriate where the spare machines are used for other purposes – for example, if one server hosts the backup for three live servers.

In the case of a distributed grid configuration, a failure could result in an incomplete index. A cluster of 200 million documents may consist of five columns, each with 40 million entries. If one full column fails (that is, all the servers in that column), queries are performed on a subset of 160 million possible hits. If this is acceptable, the service may well stay up. If it's unacceptable, search should be taken off-line until at least one node from each column is back up.

Fault tolerance of a search engine is guaranteed by duplicating the search rows

It is important to realize the
place and value of SEARCH
within your system...

With regard to large deployments, it is important to understand that two actions are performed by a search engine: indexing (converting the text data into a searchable index) and searching, where the binary files created by the indexing process are used to serve queries. Some options when setting up the search engine for fault tolerance using multiple rows therefore include: 1) run indexing and search on every row, which will provide fault tolerance for both indexing and search but will hurt query performance since machine resources are used during the indexing stage; 2) run with a single indexer and multiple search rows, providing fault tolerance for search but not indexing, but allowing the search performance to scale better since each search row is 100% dedicated to search; and 3) a hybrid approach.

An important factor when designing mirrored systems is the number of co-dependencies. For higher fault tolerance, each row calls for its own network switch and power layout.

In addition, each server should have built-in robustness to protect against hard-disk failures, which can be achieved with *RAID* (Redundant Array of Independent Disks) disk configurations. For machines with raw data, such as the crawlers, fully mirrored disks should be used. For search servers, the extra disks may be better used for increasing performance with disk striping rather than redundancy, or by choosing the RAID 5 level (striping with a parity disk) since the index can most often be rebuilt from the content in the case of disk failure.

Correctly and diligently assessing the probability of failure makes system design a cost-neutral and **well-justified business decision**, rather than a gamble on the unforeseen

The use of a SAN (or NAS) is another option for improving resistance to disk failures, since networked disk systems can be designed for high performance with redundancy.

Search broker: High availability of search necessitates redundant search brokers. Traditionally, multiple instances must be installed with a third-party load balancer which will distribute queries and accommodate for failed servers. The load balancer can be either hardware- or software-based.

Realistic design of high-availability systems

There is a basic trade-off between hardware and maintenance costs and service uptime when designing a high-availability system. Correct and diligent assessment will make the system design a cost-neutral and well-justified business decision rather than a gamble on the chances of a system outage.

Depending on your uptime requirements, some essential metrics and costs to take into account are *mean time between failures* (MTBF), *mean time to recovery* (MTTR), the cost of hardware and maintenance, and the costs of indexing and search down-

My e-commerce site generates revenues of thousands of dollars per minute. How much would it cost to guarantee 100% uptime?

Unfortunately, no amount of money can guarantee absolute 100% uptime. Having three fully independent data centers, each with 99.5% SLAs (that is, fully mirrored redundant systems), along with highly experienced systems administrators to repair systems as soon as they break down, is your best guarantee against downtime.

time. These are important when reviewing the *SLA* (service level agreements) of data centers, hardware suppliers, and support contracts.

Best practice is also to compare premium hardware service contracts (e.g., four-hour onsite repairs from Dell, IBM, Cisco, etc.) versus anticipatory purchasing of spare hardware such as network switches. In reality, hard-disk failures are the most common cause of server outages. Assuming the search index has redundancy, these malfunctions can be easily guarded against with spare disks.

Consideration must be given to the *procedures to recover* from failures – replacing faulty hardware and rebuilding the servers. Frequently the most robust setups are not necessarily those with the most costly redundant hardware; rather, they are the setups that were planned for and tested for failures. It's critical to have a recovery plan for every kind of disaster and to test the plan before an actual event occurs. The plan and the tests should include everything from relatively simple recovery procedures such as recovering and re-synchronizing a single row to restoring an entire index from backup and setting up basic search on a totally new set of servers.

Different companies, different solutions

In the case of a mid-sized company using search across a knowledge management system or an intranet, search is non-critical to the company's business continuity. The downtime cost is tied to the extra time users must spend looking for information. A redundant search node will protect against failure of the core index without the extra complexity of load balancers and a fully distributed system.

On the other hand, search is critical for an e-commerce player since revenue generation and brand strength rely on provision of quality service. The application requires a data center with power generators, a secure server room, a highly available grid of servers, load-balancers and so on if it is to provide search redundancy. In this instance, there is redundancy of the search service, with graceful degradation of performance when nodes fail, but no indexing redundancy.

For cases where uptime is vital, such as mission-critical applications, infrastructure management tools can be used to monitor the health of the hardware and software. In addition, applications such as military intelligence may consider a fully robust system with independent secure data centers, on-standby passive search nodes, and fault-tolerant hardware.

The fundamental steps to improve search

Designers of fault-tolerant systems often make two common mistakes. First, they fail to address fault tolerance until after designing the initial system, making an architecture rework prohibitive. Second, some systems are over-engineered to deal with unrealistic threat levels.

It is important to know the place and value of search within your system. For example, a search engine uses an index that

My archiving solution application is replicated for redundancy. How do I ensure the same level of protection for search?

When there is a pre-existing multi-server distribution of data within an application, the simplest method is to replicate that structure. F. ex., if each archiving box stores 50 million documents, each should have a search node of the same size. If it's feasible, this can be installed on the same server as the data. In this case, the redundancy model will automatically conform to the data fault tolerance levels.

can be rebuilt from the original data, so it's crucial to assess the true cost of downtime (for example lost revenue, or IT staff being taken off projects to rebuild systems) to determine the main concerns.

The size of the main system will have cost effects on the failover system. *So it's vital to size the system carefully.*

Simple mirroring of the search nodes will provide high availability to the level required for most applications. Then the decision must be taken whether each complete node should be designed to handle all expected traffic, or whether there is a degradation below the required service level (query speed, average response time, etc.) during a partial failure. Systems like these typically run for more than two years without fatal outages. When search is mission-critical or downtime comes at a very high financial cost, service providers should consider a complete second data center or higher level service agreements with hardware support providers.

Lastly, a system is deemed "available" if the end-to-end application is online. Therefore, the investment in enabling highly available search must be made within the context of a robust IT infrastructure.

MINI CASE STUDY: 100% uptime for over two years on a 200 million-document index .

WHO: Major online science journal index.

CHALLENGE: To support 100% of queries against an agreed SLA even during power outages.

SOLUTION: Three mirrored nodes, each sized to handle the projected loads alone, and each with independent network switch and power layouts. There are dual network feeds to the data center as well as multiple front end and query servers. Four-hour service contracts or spare hardware are available for all servers and networking gear .

FAQ

What's a fail-soft system?

When system components fail, a fail-soft system continues to operate, but with reduced functionality. Such systems are also often said to provide "graceful degradation".

What's a fail-stop system?

A fail-stop system will not provide any functionality if system components fail. It may return false results – a situation often referred to as a "Byzantine failure".

What's a SAN or a NAS?

The acronyms refer to "storage area network" and "network attached storage". The servers used are remote high-performance drives shared across multiple machines, and often connected with a fiber channel. SAN uses a disk controller and acts as a local disk, communicating via disk-access protocols, whereas NAS uses network protocols to communicate.

What is RAID?

It is short for "redundant array of independent disks" – a configuration of multiple drives used to provide fault tolerance (via mirroring or parity checking) or higher performance (via striping).
RAID 5, with striping and parity checking, is frequently used to increase search- engine performance and provide a certain level of redundancy.

What is MTBF?

It is the mean time between failures - the total elapsed time subtracted by downtime divided by the number of failures of the component.

NO CUT
MES

Reporting & Benchmarking Search

A search engine may be viewed as a set of servers that will run with little or no intervention. However, to guarantee user satisfaction, search performance and quality must continually be improved. The prerequisite is to monitor the performance and behavior of the system.

Reporting and benchmarking creates a structured method for measuring and validating the success of search with respect to these varied stakeholders. Without adequate measurements, enhancements may not be possible. The most popular queries, broken links, and user surveys are often used to evaluate the quality of search.

For example, by tracking the average response times for queries over time, the service provider can see whether the system has enough capacity to meet peak loads.

Different stakeholders have quite different objectives. For example, a Yellow Pages user will be looking for local information while someone adding a search function to an IYP (Internet Yellow Pages) site wishes to increase advertising sales.

Within each vertical industry sector there are four major categories of people interested in benchmark data: the consumer, the executive team, the business unit manager, and IT. This chapter will address how different areas of reporting and benchmarking are useful to these groups, and how they can be used to ensure that the needs of all players are satisfied.

What information is most useful to...

Reporting can be divided into two categories: searchable data and index information, and search engine usage metrics. The first category includes the number of documents in target repositories, an audit of those documents, and information on hardware usage. Usage measurements include hard numbers such as click-counting, and subjective measurements about the quality of the interface and the results ranking.

My e-commerce generates a lot of traffic, but not as much revenue as expected. How do I rectify this?

It's likely that visitors are coming to your site intending to purchase, but they're not finding compelling products to buy. Do many of their queries result in zero hits? If so, synonyms, lemmatization, and spell-checking can be used to increase recall. If lots of queries result in pagination, the ranking model may need to be changed.

By monitoring user behavior, it's possible to understand what products visitors are looking for, why the sales are low, and then act on that information to improve your site's profitability.

These categories can be split up further according to the stakeholder groups and the reports relevant to each. Identifying which group has an interest in which metrics helps to assign the ownership of search components to different groups; who is responsible for identifying sources of content; whose role it is to determine what good and bad results are; who is responsible for building and maintaining the platform?

Below, the different benchmarks and which category they are important to will be examined in turn.

...the consumer?

Search engines must keep track of *how many documents the index contains*, what the growth rate is, and how *fresh the information is.* These are typical of the factors that influence how users choose a Web search engine. So it's clear that good reporting of index volume can be used to influence a site's overall traffic.

Comprehensive user-facing index statistics reports give information discovery users the confidence to use search for their professional investigations. For instance, if an attorney is using a litigation support tool to find all WorldCom e-mails that mention former CEO Bernie Ebbers during 2003, he or she will need to know that the search is performed against 100% of all of the e-mails in that period. More realistically, it will be acceptable that some documents are not searchable, but there will have to be a detailed report of why those documents aren't searchable.

From a query point of view, it is sometimes worthwhile to show end users the *most popular searches or items.* For e-commerce, for example, the most popular sales reflect what the user is most likely to be looking for or may be tempted buy. Reporting that information directly to end consumers can drive sales by placing popular items prominently – the Internet equivalent of candy bars at the checkout line.

... to IT?

The IT department is responsible for maintaining the health of the search service. So the team must be able to monitor the *vital statistics of the system*, see what values are causes for concern, and identify the remedies for these anomalies.

The size of the index (in gigabytes of data on disk) and other information about the hardware will be monitored. The *percentage of downtime* for vital components or the repair time will help determine whether the high-availability strategy is appropriate for the SLAs (Service Level Agreements) that the IT team has committed to.

Search-specific reports have to be monitored including:

Index throughput: A sudden drop in the number of documents indexed per day often indicates a problem with a connector. For example, the password to the repository may have expired or the quality of the network connection to the source has degraded. A surge in the index rate may change the hardware requirements of the search installation.

Index freshness: This means the delay between adding documents to a repository and making them searchable. This is critical for certain applications such as news search.

Index size: This refers to the number of documents indexed and the size of the search nodes on disk. It's important to monitor this data because the number of servers and their configuration will have been planned based on a sizing estimate. If the index size greatly exceeds the prediction, performance will suffer.

IT is also concerned with *total query volume*, *query complexity*, *average query response time* and its distribution over time. By monitoring query speed over peak periods, the user experience will meet the targets set by the project's business leaders.

Reports on the *size of the index distributed by department* (or other numbers such as registered users or query volume per division) can be used by IT systems administrators to charge back IT resources to other business units. Correct reporting from the search engine is essential to streamline this accounting process.

... the business-unit manager?

The business manager's targets are usually the financial and qualitative success of the search implementation, which can be measured with relevant query-related reports including:

Top queries: - the most popular keyword searches. These should be tested most frequently for relevancy. The topics searched may help when deciding what types of content should be added to the index.

Query volume, number of unique users, and average number of searches per user: These figures are used to measure the pick-up of search.

Average click distance: This describes the number of clicks it takes to find the correct information after initiating a search. For instance, repeatedly hitting "next" counts towards the number of clicks. A low click distance is a good indicator of an effective results presentation and appropriate rank tuning.

Futile queries: The most common queries that return zero results and the proportion of queries that are futile. These metrics are critical to the success of any system. An empty hit list

5 things you should know about reporting & benchmarking search

1. It's hard to improve something that's not measured.

2. Continuous benchmarking is vital to the ongoing success of a search system.

3. Too many metrics can confuse administrators, so you should measure only what is relevant and manageable.

4. Different stakeholders (users, systems administrators, business-unit managers, executives) will have different benchmarking needs.

5. The best search engines supply a wide variety of reporting tools and APIs that can be leveraged.

I have embedded a search engine in my DMS. Should I expose any reports to end users?

Showing too many statistics to end users will confuse them. Search should simply be a seamless part of your DMS.

The uniqueness of a DMS is that users often search documents that they created, or that they heard of from a document's author. So it's important to provide user-facing reports on the document-processing stage – for example, an audit of documents where indexing is pending, or of documents that weren't indexed for some reason.

indicates an unsuccessful search, and must be rectified by adding data sources or changing searchable fields. For example, if users often search by department ID because it was the easiest way to find information in a legacy database, then the department ID field should be indexed. Synonym lists can be used if the search user's vocabulary does not correspond to that of the content.

Abandonment rate: the percentage of times an end user leaves a search page without clicking on a result. This indicates that users are dissatisfied with the results.

Click-through rate (CTR): the number of times a user clicks on at least one result. For sales-driven sites, CTR may mean the proportion of searches that result in a purchase. For Web sites with pay-per-click revenue models, the number of times that someone follows an advertising link is vital. A poor advertising CTR may indicate that the banners are not aligned with the users – products do not correspond to the user demographic or the algorithm used to link a search to a banner is not appropriate.

Within knowledge management and intranet projects, business managers target satisfaction rather than revenue. Here, the CTR and the frequency of "satisfied" users must be measured through user surveys.

In both cases it is important to monitor page impressions, i.e. how often documents or products are viewed. This may be fed back to relevancy models for example.

Focus groups and test users can determine how "good" results are. Typically, the top N most frequent searches will be reviewed to determine whether the results were appropriate. It can help to perform the same tests on competitors' sites. The business manager must define what constitutes satisfactory quality. To score relevancy models, a more structured approach involves devising rules or tests. Examples include:

- At least two of the top ten results should be no more than 48 hours old.
- All test queries should return at least 20 hits.

Two types of rules exist: global rules and query-specific rules. Global rules are evaluated against many searches; query-specific rules are evaluated against result sets for specific queries. Automated benchmarking tests can be used to monitor the quality of search without daily manual intervention.

Once tangible numbers are measured, they become critical information for the search provider, and a feedback loop can be constructed to improve user satisfaction. In addition to these query satisfaction measures, the business manager will likely be given objectives such as revenue targets.

For an OEM launching a search-derivative application, the target will be linked to the *volume and average price of sales* of the final software product. For an e-commerce site that tunes ranking models, a change in the value and number of sales per search will demonstrate its effectiveness – or lack thereof.

...to executive management?

The executives are focused on the strategic goals of the organization, not on the detail of IT implementations. Nevertheless, search is often a strategic ally. Small companies rely on word of mouth and personal relationships to propagate expertise. If a business experiences rapid growth, an intranet project with search can be used to spread knowledge and best practices among the employees. A *reduction in the time spent looking for information* and the amount of duplicate work done would both be measures of the success of such a search project. The uptake of search, such as the *volume of traffic*, will also be of interest.

When the core business function of certain employees is linked to search, their *efficiency* will be representative of the efficacy of the search tool. Information discovery workers and litigation support staff are two such examples.

For both OEMs and e-commerce sites, sales revenues and profit margins will be monitored. Executives will be interested in measuring incremental sales following the addition of a search function or an upgrade to a more advanced search engine.

Search implementations all share one point in common – the decision to invest in the software, hardware, and manpower required for the project will have been made based on an ROI (Return on Investment) calculation. This shows that the cost of implementation was outweighed by, for example, the cost of the increased productivity, increased traffic, advertising income, improved litigation success rate, or higher sales, which are all measurable quantities.

How to generate reports

Each stakeholder will have an interest in different reports to quantify the value and success of search. IT managers will monitor QPS rates, business managers will have an eye on the quality of results, and the chief technology officer will want to calculate the total ROI of the implementation. It is normally the responsibility of the technical teams to provide adequate reporting tools for each group. There are varying methods and applications to generate reports, and some general types of reporting to consider:

I get lots of complaints from my intranet users that they can't find what they're looking for.

What are your users looking for? The first step is to mine your query logs to learn what are your most popular queries. Then you should approach test users to find out what documents they would expect from their queries. Either the correct corporate resources have not been indexed, or the popular and authoritative sources need to be boosted. You should boost specific documents for "expenses form", "lunch menu" and other common queries where the text may not appear in the document.

Content reporting

Basic content statistics include the total number of documents indexed, the number of documents per collection, and the last time the index was refreshed. Information on the content aggregation and document processing will be viewed using each connector's administrative UI. More advanced reports will need to use callbacks from the content API of logs to provide the most complete vision of indexing.

> Only the curious will learn and only the resolute overcome the obstacles to learning. The quest quotient has always excited me more than the intelligence quotient.
>
> *- Eugene S. Wilson*

Query reporting

Query reporting will often be a mixture of directly accessible numbers and those culled from log files. This includes QPS, average and individual response times, most popular searches, and futile searches. More flexible search systems will provide an API to access log information and a mechanism to store past audits of statistical values. This must be aligned with the organization's log file retention policies.

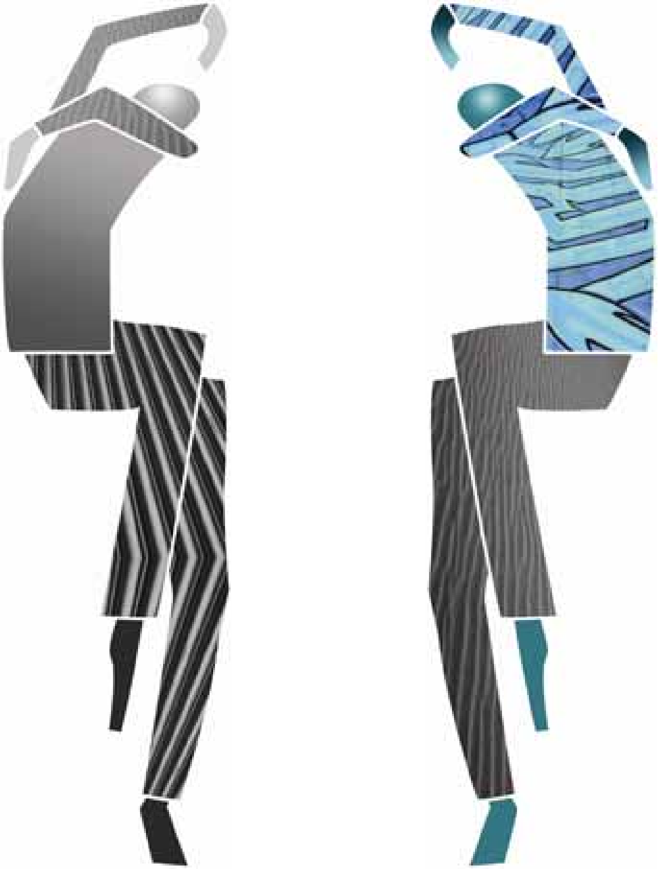
Click-through reporting

Recording user behavior is UI-dependent since the search engine only has a record of searches, query refinement, and pagination requests, not clicks-through. A link proxy page or similar mechanism must be built into the interface to allow CTR logging. To fully leverage this information, it may be useful to store the search that led to the click-through and information about the user (IP address, time of query. etc). This will allow more complete usage profiling.

Performance monitoring

Performance numbers are required to scrutinize the behavior of the search engine's hardware and software elements. Operating-system monitoring and enterprise management tools will provide information on disk, memory, and CPU usage, network bandwidth capacity, and other information with which to benchmark the system's performance. To some extent, this is true regardless of the nature of the software. For search engines, local disk I/O activity or the performance of the storage area network are critical to indexing performance while the document processing function is the most CPU-dependent.

To create reports, it is important to leverage the search engine's out-of-the-box reporting tools as much as possible. These tools typically include a simple graphical interface using data driven by internal log parsing and generated statistics. Should the system use an open architecture, the development team may decide to modify or augment the off-the-shelf tools to meet the enterprise's needs.



Alternatively the data available in a standard format such as XML or tabulated format can then be input into the reporting and trend analysis tool of choice.

Guidelines and recommendations for different segments

An understanding of the different benchmarking tools and the business goals is necessary in order to optimally monitor and improve search. This applies to the design of the system as well as to ongoing surveillance and improvements.

E-commerce and other business-to-consumer applications will be most interested in analyzing user behavior. Understanding the user base will lead to targeted improvements. Commercial considerations are also critical. Use of reporting and a feedback loop will allow for improvement.

In knowledge management scenarios such as archiving and compliancy, it's essential to monitor completeness of the data set. So the most important statistics to track relate to the connectors' behavior – crawl and index rate, document processing or indexing failures, and the size of the index.

The success of an *internal* search project is gauged by user satisfaction. If, after installation of search, pick-up is lower than expected, this may mean that employees are not satisfied with the results, preferring to rely on other methods of finding information. Satisfaction surveys and looking at top queries can also improve the search. There may be some common queries ("lunch menu", "expenses form", etc) where boosting can be used to make sure a certain document always comes to the top of the list.

Application integration (OEM) projects are an area where it is difficult to generalize reporting. Sales growth is a strong indicator, although others will be key depending on the final application. With any OEM project, the important decision is to determine how much of the search engine should be exposed to the end users and administrators of the product. This applies to reporting information – too little will increase maintenance and support issues but too much will confuse.

THE SUCCESS of an INTERNAL SEARCH project is gauged by user satisfaction

The fundamental steps to improve reporting

Precise and clear gathering and analysis of index and user statistics is essential for benchmarking and reporting. Without these measurements, there is no concrete method for defining the success of search. Without metrics, it is difficult to determine which parts of the application require tweaking and whether system modifications have had any impact on the user experience – or on the profitability of the implementation.

Other problems can arise when benchmarking becomes excessive. Too much information will likely confuse the system's administrators; it usually indicates a fundamental lack of understanding of the challenges and goals of the implementation. So with any search initiative, it's crucial to identify exactly what to measure, and how and when to measure it.

First, the key goals and challenges for the project can be identified, which leads to identifying the critical metrics. Then it's necessary to map the responsibility of the feedback loop for each area to the various project stakeholders. This "responsibility mapping" must be assigned to a trusted team member - someone who has the leverage and knowledge to get the necessary changes made.

Unfortunately, search reporting and benchmarking is often forgotten. By choosing a search engine that provides the appropriate benchmarking tools, stakeholders can more easily measure clearly identified sub-systems and act on the results to continually assess and improve search.

MINI CASE STUDY: E-commerce site uses product popularity to boost sales.

WHO: Large multi-brand e-commerce site aimed at teenagers.	CHALLENGE: To tap trends in teenage fashion to boost sales.
SOLUTION: Monitoring of the most popular sales, and segmenting that information by different criteria, such as sales of brand X or sales of items in category Y. This information is then used to prioritize the brands and categories which are most popular at any point.	TECHNOLOGY: Sales reporting and monitoring using industry standard e-commerce analysis tools. This information is then used by the content managers to tweak the ranking in the search engine for particular areas using static boosting.

FAQ

Who should be in charge and what will they have to do?	Different stakeholders are responsible for different areas of search. Make a plan for how your team will use reports and summary data to improve and refine the user's search experience.
What do I use to test the system?	The utilities and administrative APIs of the search engine are the first points to consider. Then we recommend that you leverage production system testing and analysis tools to gain valuable insights into the system's operational profile. Also, deploying an off-the-shelf Web server monitoring tool can be a quick way to monitor traffic information.
What do I change?	Based on the data in the reports, you can update the document processing, index schema and rank profile, and update the query and results processing to improve the user experience and profitability of the system
How do I fix empty result sets?	Futile queries are those that return zero results. You should investigate why these might occur – for example, because of spelling errors, semantics, or UI difficulties – and then work to correct the underlying causes.
Do I have to deploy enough hardware to maintain peak loads to ensure a timely and high-quality search response?	As you look to improve the search system, you need to consider how to minimize TCO while still providing world-class service. Consider renting servers for peak loads – for example during holiday seasons for an e-commerce site.

DICTIONARY

A

Absolute boosting - Absolute boosting enables a document to be consistently displayed at a given position in the result set when a user searches with a specific query. It also prevents individual documents from being displayed when a user searches with a specific query.

Access control list (ACL) - A data set which defines permissions, or access rights, for users and groups for a specific system object, such as a directory or file.

Alert - A message that the enterprise search engine broadcasts (for example, to a front-end application, or a messaging system such as e-mail, SMS or IM) when a document satisfies a stored query. Alerts are either near real -time or configured as asynchronous events run on a scheduled basis.

Anti-phrasing - Identifying word sequences in queries that are irrelevant for the search.

Application programming interface (API) - A programmatic interface that enables software developers to access features and functions of a hardware or software platform. An API is the specific method prescribed by a computer operating system or by an application by which a programmer writing an application program can make requests of the operating system or another application.

Authority - In relation to relevancy, the document is considered to be an authority for this query. That is, the document is being referred to by others, for example, through web anchor texts. Many items can be part of the analysis of documents to determine this parameter – Web link cardinality, article references, page impressions, and product revenue, to name a few.

Average response time - Average time it takes for the enterprise search platform to respond to a given query. There are typically two times that can be measured: 1) the average response time of the search engine itself, and 2) that of the complete system for an end -to-end query (i.e. including the application and web server times).

B

Benchmarking - A process that allows organizations to evaluate various aspects of their processes in relation to best practice, usually within their own industry sector. Benchmarking also allows organizations to develop plans on how to adopt such best practices, usually with the aim of increasing performance. Benchmarking may be a one-time event, but is often treated as a continuous process.

Boosting - Boosting increases the relevancy value of a document, typically because it is perceived to be a more valuable resource. It is the addition or subtraction of a value to a document's rank (relevancy). By default, documents with the highest rank values are returned to the user before documents of lower rank values. Boosting can be absolute or relative.

Boolean search - Boolean operators let you define whether multiple search terms are matched within a text block. A Boolean expression is constructed by joining terms together with the special operators, such as AND, OR, NOT, and the use of parentheses.

C

Call-backs - Programmatic alerts produced by an API. For a search platform, this is usually related to the content processing and indexing status of a document.

Content management system (CMS) - A software system for organizing and facilitating collaborative creation and publishing of documents and other content.

Collection - Content that is to be processed, made searchable, and retrieved as a logical unit. Content types can be grouped by source and by the processing rules that are to be applied to this type of content.

Collection-level security - The application tier will assign different authorization levels to various collections within the search index. End users then have access to the set of collections that map to their authorization levels.

Completeness - In relation to relevancy, a gauge of how well the document matches superior document contexts such as the title or the URL. It describes what matches the query: document title, author, mention in the body text, metadata linked to the document, both root, and expanded form of words.

Concept extraction - The ability to mine concepts from data using linguistic analysis.

Connector - An integration point module that extracts data from one system and submits it for processing to the enterprise search platform.

Content - Content is the external data input to the enterprise search platform. Content is converted into internal document representation after being fed into the system.

Content aggregation - The bringing together of content from multiple source repositories for retrieval at a later time. In some cases, this term is also used for the amalgamation of search results into a comprehensive whole.

Crawling - The act of accessing Web servers and/or file systems in order to extract information to feed into the enterprise search platform.

D

Deep navigators - A type of dynamic drill-down navigator. Drill-down navigators are created across all results of a query.

Dictionary/Thesaurus - A compiled structure that enables lemmatization/ synonym expansion, and look-ups. In advanced enterprise search platforms, the compiled form of a dictionary or thesaurus takes the form of an automaton.

Directed search - A narrow search within a specified area of the indexed content. Users may choose to search within "news" if they want the latest updates on today's game, for example, instead of having to search within "news", "culture", and "sports."

Document - A piece of content that is normalized with respect to the enterprise search platform's document structure, as opposed to the content itself.

Document-processing stage -The document-processing stage may modify, remove, or add information to a document, such as adding new meta information for linguistic processing, or extracting information about the language the document is written in.

Document-level security - The protection of individual documents from access by other authorized users of the system.

Dynamic concept extraction - The ability to mine concepts from data present in the result set of a query through statistical and linguistic analysis. Can be used to group similar results together.

Dynamic drill-down - A powerful navigation tool for structured data; it provides multidimensional drill-down in structured data based on facets of content.

Dynamic rank - The process by which rank components are computed during matching related to the level of match between document and query.

E

Entity extraction - The ability of an enterprise search platform to parse and recognize informational entities, such as geographic names, persons, and company names.

ETL-type tools - Extract, transform, and load (ETL) is a data-integration function that involves extracting data from outside sources, transforming it to fit business needs, and ultimately loading it into a data warehouse. In search functions, it is often used for merging of database records and content normalization.

F

False positives - When a search returns results that do not contain what was searched for.

Federated search - In a federated search, users receive results from multiple "targets” – for example, from other search engines, commercial information services, or internal databases. Federation is the blending of results from multiple, often non-compatible search systems.

File traverser - Tool for accessing files (e.g. MS Word, HTML, and XML files) that live on a standard file system in order to bring them into the index of the enterprise search platform.

Footprint - The portion of computing resources – typically RAM, CPU time, and disk space – required by the software component in question.

Freshness - The "age" of the document compared to the time of the query. For an index, how up-to-date the index is with respect to the original data source.

G

Geo/Location - In relation to relevancy, the importance of location in relation to the query term.

Golden set - A number of documents and queries that are to be used for testing; a minimum of 2,000 documents and at least 50 queries. Typically these are manually selected.

I

Index profile - Configuration file that defines the fields and properties of the index, similar to an XML schema, but also specifying field types and search engine-specific field features.

Index-based security - Resolution of a repository's document ACL permissions at query time by the index itself through the use of stored meta-data. With this method, results lists only include hits for which the searcher has viewing permissions. Compared to post-processing, the index-based security method gives higher query performance and enables the search engine to return correct counts for navigators and related concepts.

Indexing latency - The time between when a document is added and when the change to the index is made.

Ingestion rate - The number of documents per unit time that an enterprise search platform can process.

L

Lemmatization - Utilizing lemmatization enables the search system to recognize and match different grammatical forms of a word. For example, searching for "mouse" will also produce hits on "mice."

Lemmatization by reduction - The type of lemmatization, also referred to as "base form reduction," that reduces queries to the base form of the entered query term. For example, "ate" becomes "eat."

Lemmatization by expansion - The type of lemmatization which expands words into their inflected forms. This can be done either on the indexing side or query side.

Linguistics - The study of the nature, structure, and variation of language. In advanced enterprise search platforms, linguistics analysis enables transformation of content and queries for the purposes of improving relevancy, recall, and precision.

Link cardinality - The number of links in a set that refer to a given document. It is best used to determine the relevancy of a Web page by factoring in how many other pages refer to the page under consideration.

M

Metadata - Metadata is often described as "data about data." It typically augments the full text of a document to help with recall, precision, creating filters, and working with navigators.

Mining - Finding useful facts in databases of text; evaluating large amounts of stored data and looking for useful patterns.

Morphologic analysis - Used in query analysis, this analysis includes all forms of a given word via linguistic normalization (lemmatization).

N

Name-value pairs - In a search context, name value-pairs are raw data that is normalized into a structured "tree" of information. They are then sent downstream to waiting document processors. For example, name value-pairs can be data about cars that is structured into categories containing information about "make", "color", "year", and "mileage."

Natural language processing (NLP) - The process of using linguistic analysis to infer meaning from human-written text that could not be extracted using the individual word meanings.

Navigators - A navigator is a construct that enables filtering and grouping of search results. On an international site, you may have a navigator that enables you to only display results with content in a given language – for instance, “Display English results only.”

Node - In general, a node is a basic unit used to build data structures, such as linked lists and tree data structures. In an enterprise search system, a node is usually refers to one server in a distributed installation.

O

OEM - Original Equipment Manufacturer - a company that builds products or components that are used in products sold by another company.

Ontology - Ontology defines concepts, providing a way to move towards consistency in vocabulary. It provides a working model of the entities and interactions of a particular topic, such as dentistry or anthropology. It also has a specific knowledge related to a given domain name -for example, in finance or pharmaceuticals.

Orthographic analysis - Orthographic analysis is used in checking for typing errors and official variants (for example, German spelling).

P

Parsing - The process of analyzing input to determine its grammatical structure with respect to formal grammar. A parser is a computer program that carries out this task. Parsing transforms input text into a data structure, usually a tree, which is suitable for later processing and which captures the implied hierarchy of the input. Generally, parsers operate in two stages, first identifying the meaningful tokens in the input and then building a parse tree from those tokens.

Phonetic search - Phonetic search is the analysis of words that are pronounced similarly in order to detect all possible variants.

Phrasing - The recognition and grouping of an idiom such as “home run” or "Christmas tree."

Precision - Precision is the ability to retrieve the most precise results. Higher precision means better relevance and more precise results, but may imply fewer results returned.

Proximity boosting - Documents that contain the query terms closer together are ranked higher than documents that contain these terms distributed throughout the document.

Q

Queries per second (QPS) - The number of queries that the enterprise search platform will process in one second. This is normally a function of hardware (capability) and licensing (what is allowed due to contract terms).

Quality - In relation to relevancy, the quality of the document, and how important it is as viewed by the content owner or search application.

Query - The combination of the word or words used for searching, and any options allowed by the search engine.

Query and result processing - The application of algorithms to the original query or to the raw results returned by the search engine. This is useful for modifying queries to reflect an inferred behaviour – for example, using synonym expansion or business rules to modify the results (resorting, teaser modification etc), and to customize the search experience. The overall goal is to analyze and identify the essence of the searcher’s intent from the query, and to return the most relevant set of results.

Query syntax - The semantic rules that must be observed when submitting queries to a search engine – for example, the use of parenthesis and Boolean operators. Sometimes, a query transformation stage may be used to allow end users to use a different syntax from the one expected by the search engine.

Query transformation - The analysis and subsequent rewriting of a query, using linguistic transformations such as lemmatization and spell- checking. Custom query transformation stages may also be used if necessary. Equivalent to Query Processing (above).

R

Range restrictions - The ability to limit a search to a specified range of a numerical metadata field. For example, a search for a digital camera between $250 and $400.

Rank profile - The concept of a rank profile enables full control of the relative weight of each component of relevancy (for example, how important an article's title is relative to the main text or how important is proximity versus freshness). This enables individual relevance tuning of different query applications.

Ranking - Ranking is a way of arranging result documents according to their relevancy value.

Ranking models - Models used to determine how closely content matches a particular query, and whether it should be included in the search results.

Recall - For a query, recall means the ability to retrieve as many documents as possible that match or are related to a query. Recall may be improved by linguistic processing such as lemmatization, spell-checking, and synonym expansion.

Relative boosting - This enables a document to always be displayed among the first 20 documents in the result list, provided a user searched with a specific query. For all other queries, the ranking position of the document will not be affected.

Relevancy - Relevancy is the measure of how well the indexed page answers the question. Only the searcher can actually define how relevant a document is, in relation to their query: there is no way to automate it. When there are many query matches, the search engines must rank the results by relevance score, sorting the results listing so that the pages most likely to be useful will appear first. Varying algorithms are used to define relevancy.

Results clustering - Grouping similar results together to make it easy to see which results relate to each other. This can be supervised or un-supervised.

Results transformation - The algorithmic processing of search results, which includes result-set reordering (e.g. duplicate removal), adding navigation information, and result content conversion or reformatting. Equivalent to Results Processing (above).

Result-side (shallow) navigators - A type of dynamic drill-down navigator. Drill-down navigators are created across an extended but non-exhaustive result set (for example, the 200 highest ranked results).

S

Scalability - Scalability indicates the capability of a system to increase total throughput under an increased load when resources (typically hardware) are added.

Scope fields - A scope field contains hierarchically structured content. It enables schema flexibility and the ability to conserve hierarchical relationships rather than flattening the data as is often required by meta-data engines.

Semantic analysis - This means applying a combination of general and specific thesauri and ontologies, and automatic phrasing, – for example, to understand the intention of the query.

Sentiment analysis - The evaluation of the sentiment - typically positive or negative - of the text based on the usage of language. Determining the sentiment (general tone) of a document based on the application of computational linguistics algorithms.

SME - A subject-matter expert (SME) is a person knowledgeable about a given topic or subject area.

Statistics - In relation to relevancy, statistically how well the content of the overall document matches the query. One measure is the number of times the query terms appear in the document, and how rare that term is within the complete corpus. Another is the proximity of the words in the document – how close they are to one another.

Stop words - Words which are very frequent and have little meaning. They can be omitted from searches or from the index all together. In advanced enterprise search platforms, customers can control the list of stop words by managing the stop word dictionary.

Structural analysis - Structural analysis allows documents to be classified based on structure and linguistic analysis (for example, the home page of an Internet service provider (ISP)), as well as the detection and extraction of more complex elements such as the opening hours of the ISP's customer service operations.

Supervised clustering - Supervised clustering provides a grouped view based on pre-defined categories, and maps results to pre-determined categories (that is, category information provided for the documents prior to indexing).

Synonym expansion - When a query or document is expanded with a defined list of synonyms for the words it originally contains.

Syntactical patterns - Used for detecting information entities such as people, places, product codes, and prices.

Syntactic analysis - Used to analyze query through entity/phrase extraction, anti-phrasing, and to remove word-sense ambiguity (the color orange versus the fruit, for example).

T

Taxonomy - Taxonomy is a defined hierarchy of categories – a treelike structure of customer- or market-specific terminology that defines how categories relate to one another. It provides a conceptual framework for discussion, analysis, or information retrieval. For example, a car manufacturer may have a taxonomy based on the type of car (convertible, SUV, wagon, etc.). Taxonomies help partition the search environment and experience, based on a pre-defined knowledge of categories. This helps limit the number of "noisy" results returned to the user.

TF-IDF - TF and IDF are used together as a measure of the statistical strength of a given word relative to a query. TF (term frequency) is the measure of how often a word appears in a document. IDF (inverse document frequency) is the measure of the rarity of a word within the body of the document.

Tokenization - Tokenization involves detection of white space characters and other symbols that separate words from each other and that are not relevant to the matching process. It is part of the linguistic analysis, where text is split into word entities. More complex tokenization is used for CJK languages, where semantic analysis is required to identify word boundaries.

U

User interface (UI) - The end-user application linking a person to a computer program. Most modern applications leverage a graphical UI (GUI) to accept input and display information in various forms.

Unsupervised clustering Unsupervised clustering provides grouping of related documents on the basis of their content without referring to a taxonomy; it creates a taxonomy "on -the-fly," parceling documents into dynamic partitions.

V

Vectors - Vectors are a kind of document signature (word-weight pairs) representing a document's content in a way that allows comparison between documents. It is the numerical representation of the unstructured textual content of a document. Vectors can be used to enable clustering and refinement operations.

W

Wildcard - A wildcard character can be used to substitute for any other character or characters in a string. Common wildcards include "*" (zero or more characters) and "?" (a single character).

Z

Zero results - A futile query; a query that returns 0 (no) results.

Acknowledgements

This book is a team effort by many employees, customers, partners and friends of FAST. The content is aggregated and compiled from their combined verbal feedback, implementation experiences, presentations, whitepapers, case studies and joint workshops. We would especially like to thank FAST's scientists, engineers, and marketers, for their revisions and input. The core book team, in alphabetic order, is: Kjetil Halvorsen, Tony Hart, Damien Islam-Frénoy, John Puopolo, Silvija Seres, Pernille Wessel and Carine Zeier.

Illustrations are based on drawings by Janine Magelssen - www.janinemagelssen.com.

For feedback or more information, please contact us at info@fastsearch.com.

Would you like to know more about how search can help your business?

Please contact us at:

info@fastsearch.com

or visit

www.fastsearch.com

:::fast

Retail price: $35.00