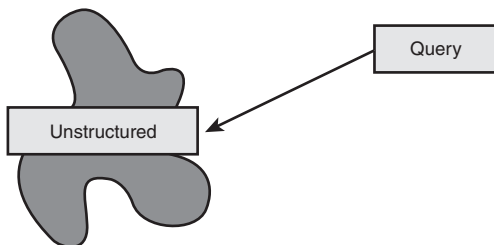


3

First Generation Textual Analytics

Search technologies have existed for as long as there has been textual data. In time, these search technologies have become more powerful and more sophisticated. In many ways, it was with the Internet and the Internet search engines and the queries that they support that the world took its first good look at search technology.

The first generation of textual analytics is defined by technology whose primary purpose is to access and analyze unstructured textual information based on a search word. Figure 3-1 shows the basic form of first generation textual analytic technology.



The Unstructured Environment

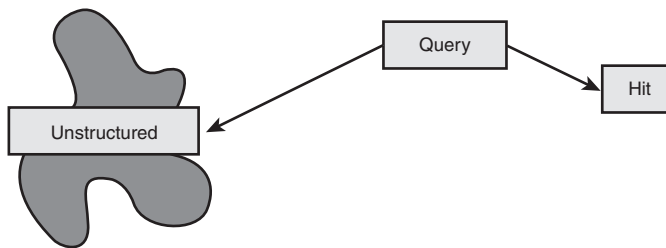
Figure 3-1 First generation unstructured processing

Simplicity

In concept, the first generation textual analytic software is simple. The purpose is to look at a body of unstructured information and to find the documents where certain unstructured information is located. The search is based on an *argument*. The argument is the search criteria that is defined by the analyst doing the search and has been passed to the search engine. The analyst defines the search argument, specifies the unstructured data to be analyzed, and then feeds that information into the search technology. The search technology then rummages through the unstructured documents and the text that is contained therein, and, upon finding a match with the search argument, creates a *hit*.

A hit simply means that the search argument is found in at least one place in the document.

In some cases, the search technology looks for only one occurrence of a hit within a document. In other cases, the search technology looks for all hits found in the document. Figure 3-2 shows the mechanics of the search technology and a hit that has been made.



The Unstructured Environment

Figure 3-2 The query operates on unstructured data. Upon finding a match, a hit is created.

Search Engines That Look for Patterns

There is more than one type of search engine. The one that has been described is used for looking through documents in which there is little or no pattern of text. Occasionally, unstructured data occurs in a repeatable format. In some cases, the repeatable pattern is so prevalent that the data can be called semistructured data. In other cases, the patterns appear in only a small part of the unstructured data and are not considered to be semistructured data. An example of a pattern search in semistructured data is a collection of resumes. Each resume has a certain similarity. There might be a collection of chemical information sorted by chemical. Each chemical has essentially the same information about the properties of the chemical. There might be a collection of movie reviews sorted by movie.

In each of these collections of unstructured information, there is unstructured data. Each collection also has a certain sameness of text. In such a case, the search technology does not look for an argument. Instead, the search technology looks for patterns of data, such as a telephone number or a social security number. Or, the search technology looks for text that follows certain key text, such as the text that follows the following words: name, address, movie title, etc.

This type of search technology is called semistructured search technology. Of course, patterns might exist in unstructured text, too.

Search Engine—The Hit

The output of the search—the hit—can be formatted in many different ways. One way is to create the output in an XML format, as shown in Figure 3-3.

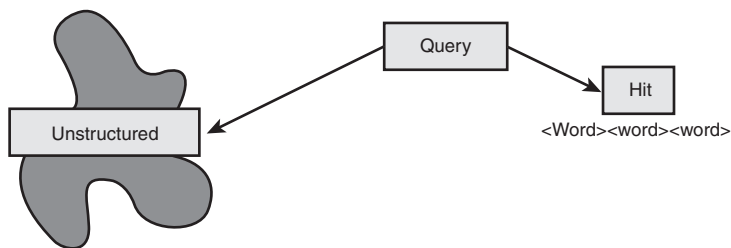


Figure 3-3 The format of the hit can be in XML.

When the hit is packaged in an XML format, it is wrapped up in a metadata infrastructure that identifies the meaning of the hits.

The hits can, however, be packaged in many other ways. A simple list displayed on a personal computer, an Excel spreadsheet, or an Access database are a few ways the hits can be gathered and delivered to the analyst.

Where Unstructured Data Resides

The unstructured data can reside in a wide variety of media. The classical media is in a textual format inside a server. In this case, the text might reside in a .doc format, .txt format, or some other format.

Text can reside in a wide variety of other formats. Another format text can reside in is a structured format where the text is collected and stored as a COMMENTS or MISCELLANEOUS field. In this case, each record in the file contains a field in which unstructured text can be stored.

Other places where you can find unstructured data might be in transcribed telephone conversations. In this case, the telephone conversations are taped, and the taped conversation is subjected to voice recognition software where the voiceover is converted into a textual form. Or, the conversation can be manually transcribed. One of the characteristics of voice transcription is the inevitable inaccuracy of the transcription. Because of accents, whispers, foreign language, and a variety of other reasons, there is no such thing as a 100 percent accurate transcription.

Another place where you can find unstructured text is in video and movie voiceovers.

All these places, and more, are where unstructured text can reside. They are also targets for a first generation search. Figure 3-4 shows that first generation textual analytic technology can be used against a wide variety of sources.

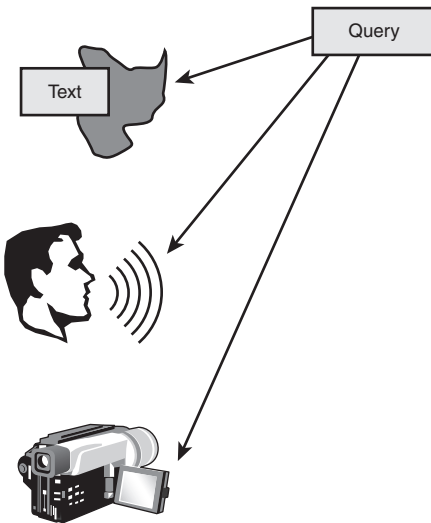


Figure 3-4 You can make a search on a variety of media.

Searching an Index

The diagram shown in Figure 3-4 is simplistic and, in a way, misleading. Although first generation search technology can do what has been described, it is much more normal for the search to go against an index that has been prepared in advance of the actual act of searching. The index is created by a one time pass through the base data where the index is created. Then subsequent passes through the search engine do not have to access the raw data but can access the index much more quickly.

Figure 3-5 shows an index that has been built for the purpose of supporting a search.

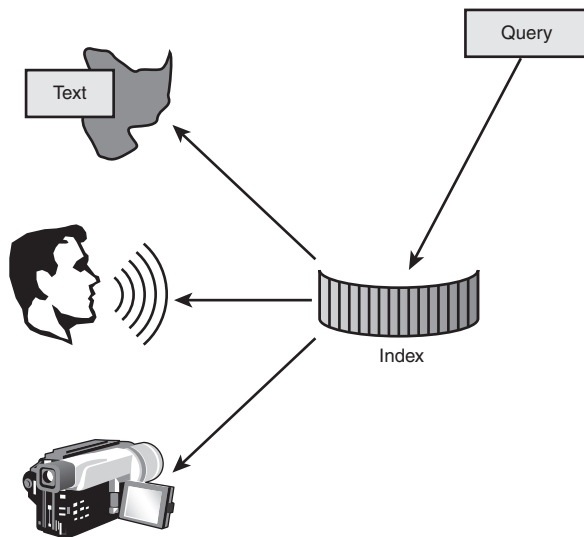


Figure 3-5 The search is often made against an index created by crawlers.

There is a good reason why search engines do not normally go against the actual unstructured text as it resides in the media in which it is found. The system performance of going against raw, native data would be poor. It might take an hour or more to find, read, and analyze the contents of a single source. No user wants to wait an hour or longer for his query to complete. People expect and demand fast response times.

A Crawler

To this end, the search technology creates a *crawler*. A crawler—in advance of any search—constantly goes after information at the raw level. The crawler loads the index that describes that information. The job of the crawler is to constantly look at new data and update existing data. When new or updated data is encountered, the crawler takes the resulting data and places it in the index.

Now, when the analyst wants to do a search using an argument, the search technology knows to go against the index, rather than the raw data. It is much faster to go against the index than it is to go against the raw data.

Furthermore, the index provides a basis for repeatability. If just one query has to be satisfied, the creation of an index might not be justified. However, when there are many queries that need to be satisfied, an index is the best way to satisfy the many permutations of arguments that are sent to the search technology.

Search Arguments and Schedulers

The search arguments can be submitted manually to the search technology. Another possibility is for the search arguments to be submitted according to a scheduler. With a scheduler, the query is submitted repeatedly according to the time set in the scheduler. The query might be submitted hourly, daily, weekly, or at whatever interval is needed. When the search technology finds a hit, the message that a hit has been found is sent to the analyst.

A scheduler submission of an argument is often called an *alert*. Figure 3-6 shows an alert.

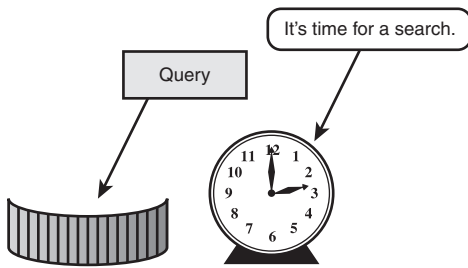


Figure 3-6 A search can be periodically triggered. This kind of search is typically called an alert.

Of course, with an alert, the message that is sent does not have to be sent to the analyst who initiated the job. Instead, the message can be sent to someone else—a manager, an auditor, a security specialist.

Tagging

In some cases, the source text can be *tagged*. When the source text is tagged, the search engine goes into the source text and adds markers to the source text that can be used in searching. At a later time, if a search is done, the source text can be directly and quickly referenced by use of the tags.

Of course, tagging depends on the ability of the source text to be modified. If the source text cannot be modified or if it is awkward to modify the source text, tagging is a less than desirable option.

Figure 3-7 shows tagged source text.

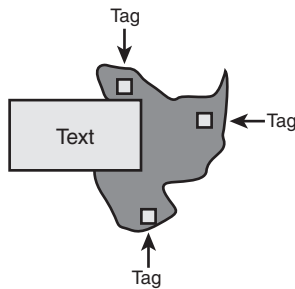


Figure 3-7 Textual data can be tagged.

Searching in Multiple Languages

Another feature of first generation search technology is the capability of the search technology to handle multiple languages. Some search engines can handle different languages and some cannot. Search engines that operate on the basis of natural language processing (or semantics) typically have a harder time handling multiple languages at the same time as do other search technologies.

Figure 3-8 shows a search technology that can handle multiple languages.

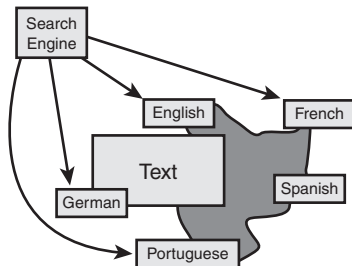


Figure 3-8 Search processing can be for many languages.

The output from a search is normally in the form of a hit. However, first generation search technologies are not limited to hits as the only form of output.

Collecting Output in a Taxonomy

Another form of output might be a categorization of words or a taxonomy of words that have been generated by the search technology. In this case, there are numerous hits that are made, and these hits are turned into a glossary or a taxonomy. Figure 3-9 shows such output.

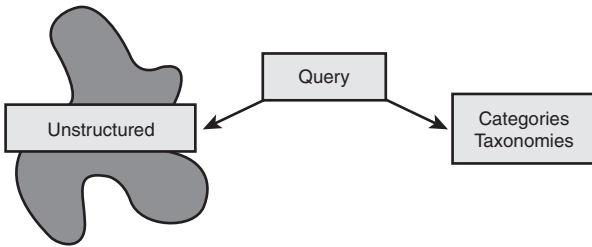


Figure 3-9 The output can also be formatted into a category or a taxonomy.

There are several ways that categories, or taxonomies, can be created. One simple way is by ranking words according to the number of occurrences. In this case, the document or documents that are used as the source contribute their words and are pooled together. The words are then counted and categorized by the number of occurrences.

Another way to create categorizations is by word proximity. In this case, the proximity of words to each other determines their categorization.

A simple form of a categorization is a taxonomy. Other forms of a categorization include a glossary or an ontology. A *taxonomy* is a collection of related words used by the corporation. Different corporations will have different taxonomies. The taxonomy can be generated by reading the unstructured documents and editing the words that need to go into the taxonomy.

Hyperlink References

Another task that can be assigned to the search technology is that of creating hyperlink references throughout the source documents and the output of the search, whatever its format. A hyperlink is merely a reference to a document or a location in a document than can be followed by the system. When the reader encounters a hyperlink, she has the choice of continuing to read the document or clicking on the hyperlink and having the system go to the document that is found in the hyperlink.

Figure 3-10 illustrates a hyperlink.

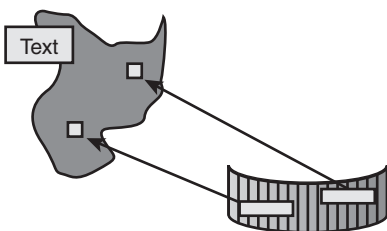


Figure 3-10 Hyperlinks can be created if desired.

Federated Queries

One of the features of first generation search technology is the support of federated queries. A *federated query* is a query that goes against multiple sources of data at the same time. The results are returned as if the query had operated against a single source. Figure 3-11 shows a federated query.

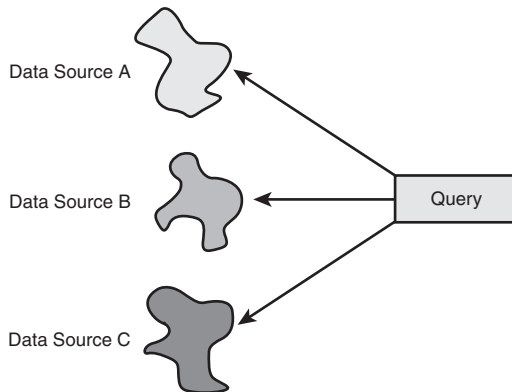


Figure 3-11 Federated queries can be created.

Federated queries have been around for a long time. They originated when queries went against multiple sources of data in the structured environment. However, federated queries that processed against legacy, structured applications always had a problem: With structured systems, there was no guarantee that there was any integration of data at the source level.

For example, in structured systems, suppose there are three sources of data: A, B, and C. Suppose A holds a record with a value of \$400 US. B holds a record with a value of \$1,000 Canadian. C holds a record with a value of \$10,000 Australian. In a typical federated query against A, B, and C, the values are accessed and added together. The total dollar value is \$11,400. The addition of money in different currencies, of course, makes no sense. If you want a meaningful result, you have to convert currencies to a common value.

Integration

The world of unstructured data does not have the integration problem, at least in the same way that the structured world does. The world of unstructured data operates against text. Assuming that text is in the same language—say, English—then the text is integrated. It doesn't matter that some of the text comes from a telephone conversation,

email, a report, and spreadsheets. At the end of the day, it is still just text and is still subject to the rules of language. Therefore, the data in the unstructured environment is integrated because all of it is text, and a federated query against text is a far cry from a federated query against unstructured, legacy data.

Stated differently, because all text arrives at the same point of integration, and because all text obeys (to a smaller or a greater extent) the rules of grammar and punctuation, integration of text needs to be done only once. Contrast this with structured data that must be reintegrated each time a new source of data is added. Because of the uniformity of language, integration must be done only once, regardless of the physical source of unstructured data.

Enhancing the Search Argument

One of the features of first generation search technology is the enhancement of the search argument. In the simplest case, suppose the search argument is “Roy Rogers.” A search can be made—literally—against the term “Roy Rogers.” Wherever that exact phrasing is, a hit is generated. However, what if in the text there is the term “roy rogers?” Is a hit generated? The answer is no, unless case-sensitivity is removed as a consideration. If the search is case-sensitive, the system finds no match between “Roy Rogers” and “roy rogers.”

Enhancing the search argument goes even further. Suppose the system recognizes “Roy Rogers” as a man’s name. The system could suggest that a search be made on “R Rogers,” “Mr. Roy Rogers,” “Mr. R Rogers,” and other permutations of “Roy Rogers.” The system could query the analyst and ask if one or more of the permutations could be used as a search argument.

The first generation search technology system could go even further. It could suggest to the analyst that other permutations might also be used as a search argument. The system might suggest the use of wild cards.

Wild Cards

A wild card is a specification to the system that any letter found where the wild card is specified be accepted as satisfying the search criteria. For example, suppose the analyst specified “R*y R**ers.” When the system encounters “Ray Rebers,” the systems will qualify that entry as a hit.

In these ways, the first generation search technology can enhance the search argument to broaden the search to be much more analytical. Figure 3-12 shows the enhancement of the search argument.

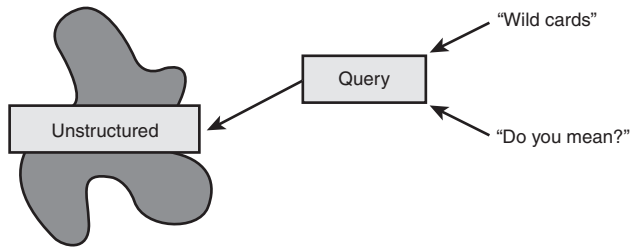


Figure 3-12 Some of the ways the argument can be extended and made more sweeping.

Boolean Expressions

Another standard feature of first generation search technology is that of the usage of Boolean expressions to qualify the search argument. Typical Boolean expressions are the AND function and the OR function.

The OR function says that if text from either A or B is found, then the source document will be included in the hit list. For example, if the search argument were “rocks OR stones,” any document that contained either “rocks” or “stones” would be included in the hit list.

The AND function works differently. If the expression “rocks AND stones” were specified as a search argument, only if a document had both “rocks” and “stones” would it be included in the hit list.

Figure 3-13 shows the support of Boolean expressions.

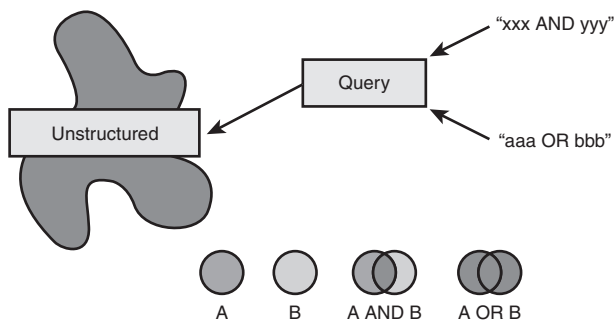


Figure 3-13 Boolean expressions can be used to qualify arguments.

Visualizing Text—First Generation

There have been attempts at visualizing text. One such attempt is a cluster diagram (see Figure 3-14).

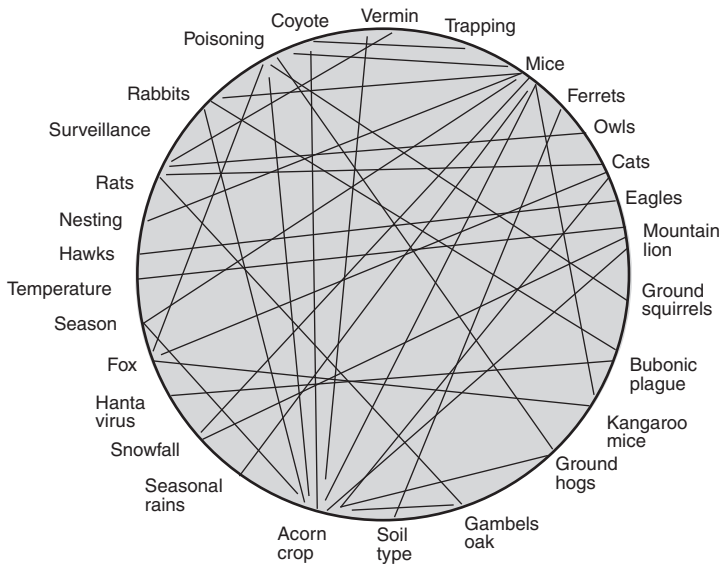


Figure 3-14 A cluster diagram

A cluster diagram takes a body of text and outlines the associations. The nouns that are in the body of text are positioned around a circle. When one word relates to another word, a line is drawn inside the circle. When one or two words start to have a majority of lines drawn in their direction, the implication is that there is a significant clustering of information around the words that are popular. When the clustering of the words is known, the first step has been taken to capturing and understanding the nexus of the text.

Understanding Context

Enterprise Content Management (ECM) is related to first generation search technology. ECM technology is essentially storage management technology that finds unstructured data and then stores it in a form that can be accessed at a later time. Typically, ECM technology provides a limited amount of first generation search functionality.

In some first generation search technology, one of the approaches to the management of text is called *natural language processing* (nlp). Another term sometimes used in conjunction with nlp is *semantic processing*. With nlp and semantic processing, the intent is understanding the context of language. The theory is that you can't understand language unless you understand the context of language.

As an example of the importance of the context of language, consider the short sentence—“She smacked him.” What does this sentence mean? To understand the sentence, we have to understand the context of each word. Who is she? Is she someone who was mentioned before? Someone standing in the room whom we know is there? A ship on the ocean? When we

understand who she is, we are on our way to making sense of the sentence. Then there is the word “smacked.” What does it mean? Does it mean she killed him, as a gangster might say? Does it mean she kissed him? Does it mean she gave him dope as he was standing in line in an alley? Who is “him”?

It is obvious that to make sense of the sentence, it is necessary to understand the context of the words in the sentence. That is what natural language processing attempts to do.

In everyday life, we understand context. Our brains help us to resolve the different meanings of speech. Brains certainly use more than words. We use location, present company, the subject of discussion, past events, and many more factors to understand context. Our brains do it naturally.

For nlp to operate properly, the computer needs to be taught how to resolve context. One of the problems is that much of context resolution is not word- or text-related. In addition, trying to teach the computer how to do context resolution outside of text is difficult.

Unfortunately, there is a trap with nlp. The trap is that you have to consider context to understand words. When that assumption is made, the process of making sense of unstructured data—text—becomes a complex, involved process.

Summary

First generation search technology is technology that is used to operate directly on unstructured data. Here are many common features of first generation unstructured search technology:

- Support of many unstructured sources
- Enhancement of the search argument
- Creation of alerts
- Operation against indexes that have been prepared from the unstructured data sources
- Support of different languages
- Federated queries
- Boolean expressions

ECM technology is technology that captures and stores textual data. Nlp technology attempts to understand the context of data as part of the understanding of how to manage text.

There is another approach to the treatment of the unstructured environment. The key is the integration of textual data, as discussed in Chapter 4, “Integrating Unstructured Text into the Structured Environment.”

