6. IR Standard - Z39.50

Z39.50 is a national (ANSI/NISO) standard for information filtering. Its two primary functions are search and filtering. These functions are initiated by an entity called the Origin, which is contained in an application called the Client, residing on the Client system. The Origin communicates with the Target, which is contained in an application called the Server, residing on the Server system. The Server is the database provider. [ANSI/NISO Z39.50, 1995] “Unlike other Internet protocols such as HTTP or WAIS, Z39.50 is a session-oriented protocol. That means that a connection to a Z39.50 server [from a Z39.50 client] is made and a persistent session is started. The connection with the server is not closed until the session is completed.” [LeVan, OCLC]

Searching is the selection of database records, based on origin-specified criteria, and the creation by the target of a result-set representing the selected records. Filtering, idiomatically speaking, is the transfer of result set records from the target to the origin [but see below]. [ANSI/NISO Z39.50, 1995].

Z39.50 started life as a standard of the library community, specifying a protocol for “searching and filtering USMARC-formatted bibliographic records … However, the standard has grown considerably … Today, there are organizations using Z39.50 to deliver full-text documents based on natural language queries.” [LeVan, OCLC]. Today, Z39.50 is evolving to meet the complex requirements of the IR community. To the extent that it satisfies these needs, and is supported by a wide variety of commercial and governmental IR providers, it will become (and appears on the way to becoming) the standard language for accessing IR engines. As such, it may play a role in the IR community analogous to the role of SQL in the structured DBMS community.

In comparing Z39.50 to SQL, one essential caveat is in order. As the preceding sections of this paper make abundantly clear, there is an uncertainty in Information Filtering which is not present in filtering from a DBMS. This is reflected in a basic difference between SQL and Z39.50. The SQL standard defines, at least in principle, the semantics as well as the syntax of the “black box” filtering that is to be performed by a conforming DBMS in response to an SQL query. What does that mean? Suppose that the same data, i.e., the same tables (or “relations”), are loaded into three distinct DBMSs,
implemented by three distinct DBMS vendors, but all conforming to the same level or version of SQL. Suppose further that an SQL query is formulated against this data (and data structure) and executed by each of the three DBMSs. Then as long as the query conforms to the level of SQL supported by the three vendors, and doesn’t use any non-standard vendor-specific features, exactly the same data should be returned by each DBMS. The internal details of how the query is executed may vary considerably from one DBMS to another, depending on how its optimizer works, how the tables are indexed, and so on; correspondingly, the response time may vary substantially from one DBMS to another. But exactly the same data should ultimately be returned.

Plainly, this is not (and cannot possibly be) the case with Z39.50 (and will be even less the case as Z39.50 evolves to support more powerful and diverse IR engines and queries). The behavior of IR engines varies far too widely. A term vector submitted to a term-based vector space IR engine will produce a different result than the same vector submitted to an LSI-based vector space engine.

The result will be different again, if the IR engine is based on a probabilistic model, and will vary from one probabilistic model to another. An extended boolean query will produce different results depending on which boolean model it employs. (And different results than a strict boolean engine which must ignore clause weights!) Even two IR engines that use the same term-based vector space approach may differ if they employ different index weighting schemes, employ different stop lists, employ different query expansion schemes, employ different query/document similarity measures, etc.

So, it should be understood that what Z39.50 provides is a consistent way of talking to diverse IR engines. The results returned may vary widely depending on all of the factors mentioned above, and more.

6.1 Searching via Z39.50

The Z39.50 search process starts (as noted above) with the specification by the origin of search criteria. These criteria are specified by a Z39.50 query. The queries currently supported by Z39.50 are called Type-1 and Type-101. The functionality of a Type-1 query is described briefly in this section. (Type-101 is functionally identical to Type-1. The only difference is that the definition of Type-101 is independent of the version of Z39.50, i.e., it works with both Z39.50-1992 and Z39.50, 1995, known as version 2
and version 3 respectively.) Z39.50 also supports some other query types with grammars that are severely limited in extensibility, are not widely used, and are not mandatory in the standard; these other query types should probably not be used and are not discussed further here. [LeVan, 1995]

Z39.50 allows the sender to specify strict term-based boolean queries using the operators AND, OR, AND-NOT, and Prox. The latter is a proximity operator that tests whether two terms are (or are not) within a specified distance of each other, where the distance is measured in units and the possible choices for the unit include: Character, Word, Sentence, Paragraph, Section, Chapter, Document, Element, Subelement, ElementType, Byte or privately defined unit. The order of the terms may be specified. [ANSI/NISO Z39.50, 1995]

Each operand of the boolean query may consist of a term and a list of attributes that qualify the term. Attributes specify something about the semantics of the given term. The attributes are drawn from an “attribute set.” An attribute set specifies a “list of the types of things that can be searched for.” [LeVan, OCLC] A number of attribute sets have been defined, and other sets can be defined in the future. A query can draw attributes from more than one set. The core attribute set, reflecting the origins of Z39.50 in the library community, is called “bib-1.” The bib-1 attribute set has six types: Use, Relation, Position, Structure, Truncation, and Completeness. “The “Use” attribute allows the client to specify how the term would have been used in the records to be retrieved.” For example, the term might be used as a Title, as (the name of an) Author, etc. At present (in the 1995 version of the standard), 99 values of Use are defined. Most of them are clearly related to bibliographic reference, e.g., Dewey and other classification numbers, Date of Publication, etc. Some have more general applicability, e.g., Personal Name, Corporate Name, Conference Name, Name Geographic, etc.

The Structure attribute specifies the structure, e.g., WORD, PHRASE, DATE, NUMERIC STRING, FREE-FORM-TEXT, etc. The POSITION attribute specifies the position of the term in the structure, e.g., FIRST IN FIELD. (These attributes are actually represented in Z39.50 by numeric codes.) [ANSI/NISO Z39.50, 1995]

The end result of the Z39.50 search process is “the creation by the target of a result-set representing the selected records.” These result set records are then transferred by the target to the source during the filtering process.
Note that Z39.50 Type-1 queries are always structured, term-based queries. Z39.50 does not support unstructured queries, e.g., documents as queries. (But see the relevance feedback feature of the proposed Type 102 query discussed below.)

6.2 Filtering via Z39.50

The “transfer of a result set record” more accurately means: the transfer of some subset of the information in a database record (represented by that result set entry) according to some specified format [called a filtering record].

Z39.50 filtering supports the following basic capabilities:

* The origin may request specific logical information elements from a record (via an element specification …).
* The origin and target may share a name space for tagging elements … so that elements will be properly identified …
* The origin may request an individual element according to a specific representation or format …
* The origin may specify how the elements, collectively, are to be packaged into a filtering record … [ANSI/NISO Z39.50, app. 14]

The structure of a filtering record may be hierarchical, e.g., may include sub-fields, sub-sub-fields, etc. “An origin might request, for example, ‘the fourth paragraph of section 3 of chapter 2 of book1.’” Or the retrieved data might be more conventional structured data, e.g., a product availability field may contain a “distributor” sub-field, which may, in turn, contain the sub-sub-fields for the name, organization, address, and phone number of the distributor.

6.3 Type 102 Ranked List Query (RLQ) - A Proposed Extension to Z39.50

A number of IR features discussed extensively in this paper are notably lacking from the Z39.50 query capability discussed above: extended boolean queries, weighting of terms or clauses, ranking of the retrieved results, relevance feedback, etc. These “ranked searching technologies [are] used by the majority of large-scale commercial information providers and information industry software vendors. This includes 80-90%
of the mainstream commercial ranked searching technologies …” The proposed type 102 Ranked List Query (RLQ) has been designed to meet these requirements. [Type 102, 1995] This query has been developed by the Z39.50 Implementor’s Group (ZIG), which includes such organizations as Chemical Abstracts Service, Clearinghouse for Networked Information Discovery & Filtering (CNIDR), Excalibur Technologies Corp., Knight-Ridder Information Services, LEXIS-NEXIS, National Institute of Health (National Library of Medicine), and West Publishing Company.

A weight may be attached to each operand in a Type 102 query. A Type 102 query “is a recursively defined structure of operators and weighted operands.” Since the query is recursively defined, a clause, e.g., an operator and its associated operands, can itself be an operand, and hence can itself be weighted. The weight attached to an operand “specifies the value to be placed on the operand with respect to its importance in selecting records from the designated collection(s).” [Type 102, 1995]

Type 102 supports operators that may take more than two operands (as required by some extended boolean models).

Type 102 supports extended, i.e., relevance ranked, boolean operators (called “relevancy-based” operators in the type 102 spec). For example, instead of a strict boolean AND operator, there is an operator called “rqAND.” Each operator and its associated operands comprise a clause. A number may be attached to each clause. These numbers determine the degree to which the clause is to be given a strict or extended boolean interpretation. Note that some servers may ignore these numbers. These numbers should not be confused with term weights.

Type 102 supports the filtering of ranked output, i.e., each result record is associated with a Filtering Status Value (RSV) which is a measure of its degree of relevance to the given query. Moreover, the type 102 query allows the user to limit the number of records retrieved, either by specifying the number of records to be returned, e.g., the top-ranked N records, or by threshold value, e.g., all records with RSV above threshold value RTHR. Note that the interpretation of the RSV is server-dependent, e.g., it might be a cosine similarity in one system and a probability of relevance in another. Also note that while the result set will normally be ordered by RSV, other orderings, e.g., by date, can be requested.
A Type 102 query can be applied to one or more record collections. (The Type 102 spec uses the term “record” instead of “document.”) The query can restrict the collections to which the given query is to be applied, or specify particular collections to which it is not to be applied. (The Type 102 spec uses the terms “collection” and “database,” apparently interchangeably.)

A Type 102 query can specify the degree to which recall is to be emphasized (at the possible price of loss of precision).

A Type 102 query may specify whether the original query may be reformulated, e.g., expanded, by the filtering engine. Moreover, the query may specify that only the reformulated query is to be returned. Alternatively, the query may specify that only the retrieved records are to be returned. Or, both the retrieved records and the reformulated query may be requested.

A Type 102 query allows the user to specify relevance feedback info, either within the original query or within a resubmission of a reformulated query. The feedback info takes the form of a list of records with a relevance measure (in the range from -1 to +1) attached to each record. Notice that the negative numbers allow the user to specify the degree to which given records are undesirable.

The Type 102 query can request the return of demographic data pertaining to the collection being queried, or to the result set, or to the retrieved record, etc. The collection-level metadata that can be returned includes: number of records, number of unique terms (either including or excluding stopwords), total number of term occurrences, total number of records in which each term occurs, and total number of occurrences of each query term in the collection.

Type 102 supports proximity as does Type 101. However, in Type 102, there is no boolean proximity operator. Instead, a proximity condition (called a “qualifier”) is attached to a boolean clause to indicate that all operands within that clause (structured operand) must be satisfied within the same proximity unit.