7. A Brief Review of some IF Systems

This section is a brief review of some of the leading commercial and research IF engines. The engines reviewed here are chosen to be representative, not exhaustive.

7.1 LEXIS/NEXIS

LEXIS/NEXIS is a commercial system for retrieving legal (LEXIS) or newspaper (NEXIS) documents.

LEXIS/NEXIS [qrel] supports traditional “strict” boolean queries, i.e., booleans that return exact matches only. Specifically, it supports queries formulated with the boolean operators (called “connectors” in LEXIS/NEXIS) OR, AND, and W/n. The latter is a proximity operator, e.g., homeless W/5 shelter specifies that homeless and shelter must occur within five words of each other. It also supports two “wild card” characters (called “universal” characters in LEXIS/NEXIS): The character! specifies any suffix that can be added to the root word, e.g., “transport! finds transportation, transporting, transported, etc.” The character * specifies any single character. It “must be filled in if its in the middle of a word, but not if it’s at the end. (EXAMPLE: wom*n finds woman, women; transport** finds transport, transports, transported, but not transportation, etc.)”

More recently, LEXIS/NEXIS has followed the trend toward natural language queries (called “FREESTYLE™ search descriptions” in LEXIS/NEXIS); these queries do not require (or permit) boolean connectors. This is, in essence, the vector space approach described earlier in this paper. It “identifies significant terms and phrases from your search description, removes irrelevant words from your search description [e.g., applies stoplists, etc.], applies a formula that weighs the statistical importance of the significant terms and phrases from your search description and compares them to the documents in the library and file(s) in which you are searching [e.g., weights the significant terms and computes the similarity of query to documents in the target collection] — the more uncommon or unique the word, the greater the statistical weight [e.g., uses tf*idf weighting or the like].”

LEXIS/NEXIS provides a number of ways of qualifying or enhancing a natural language query. The user can tell LEXIS/NEXIS to treat two or more consecutive words as a phrase by bracketing them with quotation marks. In addition, LEXIS/NEXIS itself will recognize certain word combinations as phrases, and put quotation marks around
these combinations automatically; the user can override this feature by editing out the quotation marks. The user can specify that certain words or phrases are mandatory, i.e., they must appear in any retrieved document. (Note that such a feature is meaningless in a strict boolean query, since the boolean operators themselves determine whether a given term is mandatory, and under what conditions it is mandatory.) The user can specify “restrictions,” i.e., constraints (other than mandatory words) that must be satisfied by retrieved documents; for example, a legal document may be constrained by date or court. The legal user may invoke an online thesaurus of legal terms. “A list of the terms in your search description for which synonyms [or alternative forms] are available will appear.” The user has the option of displaying the synonyms and alternative forms for a given word, and adding any of these additional terms that she chooses. Hence, query expansion via thesaurus is manual, with the thesaurus providing online guidance, but the user deciding which terms (if any) to add. Finally, the user may specify how many documents to retrieve. (Again, note that this feature would be meaningless for a strict boolean query, since a strict boolean determines a set of documents that exactly satisfy the query; there is no notion of degree of relevance in a strict boolean filtering.)

LEXIS/NEXIS provides some result display options that are only available (indeed, only applicable), to a natural language, i.e., vector space, query. For example, the user can display “the most heavily weighted block of text — the portion that most closely matches [her] search description.” More interesting is the “WHY” option. This option “shows how your search was interpreted...,displaying the order in which your terms were ranked, the total number of retrieved documents with each of your terms, … and the importance assigned to each term.” Note that this is closely related to the Z39.50/type 102 query features that allow you to look at the system expansion of your query, and the demographics of its terms.

7.2 Dialog

DIALOG [QT] is a commercial system for retrieving documents from databases in such topic areas as: Business, Intellectual property/Law/Government, Medicine and Pharmaceuticals, News, People, Sciences, Social Sciences & Reference, and Technologies.

The user selects a topic. Then she selects a database (or group of databases) within the topic. The search options vary with the database. For example, options for a newspaper
database include: Subject (keyword), Title/Lead Paragraph, Author, Journal Name, Section/Subject Heading, and Limit options.

DIALOG supports a strict boolean query capability very similar to that of LEXIS/NEXIS, e.g., AND, OR, and w (proximity) operators. Like LEXIS/NEXIS, DIALOG supports a wild card character (?) that can only be used to specify any suffix to a common root, e.g., “smok?” will find smoke, smoker, smokers, smoking, etc. There is no thesaurus; it is up to the user to think of appropriate synonyms.

In “menu” mode, the user enters a term and any synonyms connected by “OR”. Then she can modify the query by either broadening it, i.e., adding additional terms implicitly connected by “OR”, or narrowing it, i.e., adding additional terms implicitly connected by “AND.”

In “command” mode, the user can generate nested boolean expressions. To make the expressions simpler to read and generate, the user generates terms, e.g., “smoking OR tobacco,” “heart disease OR heart attack”. Each term is assigned an id by DIALOG, e.g., the first term may be assigned the identifier “S1,” and the second term may be assigned the identifier “S2.” The user can then generate compound boolean expressions using these identifiers, e.g., “SELECT S1 AND S2.” DIALOG will now assign an identifier, e.g., S3, to the compound expression “S1 AND S2.” At each stage, DIALOG tells the user how many documents are retrieved, e.g., how many are retrieved by the term “smoking” by itself, how many are retrieved by “smok? OR tobacco,” how many are retrieved by “S1 and S2,” etc. In this way, the user can decide when he has limited his filtering set sufficiently. At all stages, filtering involves a strict boolean match.

DIALOG allows the user to save a query. Thereafter, if the query matches a new document that has been added to the given database, the user is alerted. The set of saved queries for a given user is called an “alert profile.”

7.3 Dow Jones News/Filtering

Dow Jones News Filtering [Dow QR] is a commercial system that can search up to 1900 news sources, e.g., newspapers, newsletters, news magazines, etc., general interest and specialized. As with the other commercial engines described here, it supports strict boolean queries with a somewhat broader set of operators, e.g., AND, OR, NOT, SAME, NEAR, etc. A query can be further restricted by specifying a date, categories
and subjects, document sections, and specific sources, e.g., specific publications. The system displays a set of available subject and category codes; not all codes work in all publications. Similarly, not all document section types are available in all publications.

Retrieved documents can be sorted, highlighted, etc. One can retrieve a hit paragraph rather than an entire document. One can retrieve the headline and lead paragraph, or the full text of an article.

7.4 Topic

Topic [Topic Intro] is a commercial IF engine, marketed by Verity, Inc. In contrast to the three commercial IF services described above, Topic is not an IF service maintaining indexed document collections, but a stand-alone IF tool that can be used by any purchaser to provide IF services. Verity also markets an application program interface to Topic, the Topic Development Kit (TDK). This allows Topic to be incorporated into application systems, and other vendor products.

Before a collection of documents can be searched by Topic, it must be loaded “into” Topic, a process that involves sophisticated indexing. In this respect, of course, it resembles most other IF search engines, whether commercial or research, as well as the information services described above.

The basic text search condition or query in Topic is called a “topic” (formerly called a “concept tree”). Topics are hierarchically structured. Each topic has a name which is the root of its “tree”. Below the root are any number of child sub-topics, also named. A sub-topic may itself have any number of named sub-topics. Hence, there may be any number of levels of sub-topic. At the lowest level (the leaves of the tree) are “evidence topics” which specify the actual words or phrases for which Topic is to search each document in a given Topic collection. For example, the terms ballet, drama, dance, opera, and symphony may be evidence topics for a sub-topic named performing-arts. “[T]he sub-topics performing-arts, film, visual-arts, and video [may be] children of the art topic. The art [sub-]topic itself [may be] a child of the liberal-arts [sub-]topic.” The liberal-arts topic might be the root topic. Alternatively, “[t]he liberal-arts topic could in turn be a child of successively higher parent topics within the [topic] structure.” [Topic Intro]

Topic performs relevance ranking as described below. When a topic is executed against a given collection, it is evaluated against each document, and the document is assigned
a score in the range 0.01 to 1.0. The higher the score, the better the document matches the topic (according to Topic’s matching formula, of course). Documents are returned to the user in descending order of score.

An operator may be associated with each topic or sub-topic node. There are three classes of operator. The evidence operators specify the string or set of strings for which each document is to be searched. Hence, they only appear at the lowest level, i.e., below any of the other operators. For example, “WORD” specifies a word, actually a string of up to 128 alphanumeric characters, but usually an ordinary word or number, e.g., “microchip” or “80386.” The STEM operator specifies the usual stemming, e.g., to stem “transport” is to search for “transports,” “transported,” “trans- porting,” etc. (Note that stemming must be specified explicitly for each evidence word which means that the user can avoid stemming of a given word if he wishes. In contrast, some of the other systems discussed above and below performed stemming automatically.) The WILDCARD operator allows specifying of search patterns. It uses a richer set of wildcard characters than the commercial services described above, perhaps because the creator of a topic is assumed to be more sophisticated than the typical user of those IF services. Wildcard characters may occur any- where in a search pattern, not just at the end, and support single characters (?), zero or more characters (*), any one of a specified set of characters, any character in a range (e.g., [A-F]), etc. The NOT operator may be used to exclude documents that contain a specified word or phrase.

Above the evidence operators in precedence are the proximity operators: PARAGRAPH, SENTENCE, PHRASE, NEAR and NEAR/N. Each of these operators specify two or more words that must satisfy the given proximity constraint. A proximity operator may be assigned to any sub-topic above the evidence level. PARAGRAPH and SENTENCE are self-explanatory. PHRASE specifies a string of consecutive words, e.g., “arts and crafts”. NEAR/N specifies that the specified words must not be separated by more than N words. NEAR differs from NEAR/N in that “[d]ocument scores are calculated based on the proportion of instances found in relation to the size of the region containing the words … Thus, the document with the smallest region containing all search terms always receives the highest score.” An ORDER operator can be used with SENTENCE, PARAGRAPH, and NEAR/N to specify that the search terms must occur in a specified order.
Above the proximity operators in precedence are the “concept” operators, which are the boolean operators: AND, OR, and ACCRUE. The AND operator is a strict boolean, i.e., it only selects documents that contain all its children (operands). In other words, it contributes a score of zero for each document that does not contain all its children. However, if all its children are present, the score returned by AND is not simply 1.0 (as it would be for a conventional strict boolean) but will be the minimum of the scores of its children. OR is also a (kind of) strict boolean in the sense that it returns a score of zero only if all its children have scores of zero, e.g., if its children are words and phrases, at least one must be present in the document if the OR is to contribute a non-zero score to that document’s total score. Moreover, the score it contributes to a given document if any of its children are present does not depend on how many of the children are present. However, it is not 1.0 either; instead, it is the maximum score of any of its children. ACCRUE is a (kind of) extended boolean OR, i.e., the more of its children are present in a document, the higher the score that it contributes to that document. However, the score it returns is the maximum of its children’s scores (like an OR) plus a little extra for each child that is present.

A further degree of ranking can be specified by using the MANY operator in conjunction with an evidence or proximity operator to rank “documents based on the density of the search terms they use.” In other words, MANY normalizes term frequency by document length so that “a longer document that contains more occurrences of a word may score lower than a shorter document that contains fewer occurrences.”

Finally, the user may assign weights in the range of 0.01 to 1.0 to operators. Specifically each child of a logical operator (AND, OR, and ACCRUE) may be assigned a weight. Since a logical operator may be a child of another logical operator, the logical operators themselves may be assigned weights. Similarly, evidence operators (WORD, STEM, etc.) may be assigned a weight. Proximity operators may not be assigned a weight. Weights determine the relative importance of search terms or higher level children. For example, the score for a given document contributed by a given AND operator is not merely the minimum score of any of its children but rather the product of that score and the weight assigned to the child having that minimum score.

Finally, there are a number of operators that only apply to structured fields of a document, e.g., title, subject, author, etc. These operators do not rank documents but filter them, e.g., one can specify only documents by a given author or only documents whose titles contain a given sequence of words.
7.5 SMART

The SMART system [Salton & McGill, 1983], developed at Cornell, is the “granddaddy” of IF systems that (1) use fully automatic term indexing, (2) perform automatic hierarchical clustering of documents and calculation of cluster centroids, (3) perform query/document similarity calculations and rank documents by degree of similarity to the query, (4) represent documents and queries as weighted term vectors in a term-based vector space, (5) support automatic procedures for query enhancement based on relevance feedback. SMART has been widely used as a testbed for research into, e.g., improved methods of weighting and relevance feedback, and as a baseline for comparison with other IF methods.

Note that extended boolean filtering, i.e., the $p$-norm method, was developed in the SMART “shop”, although it does not appear to be formally incorporated into the SMART testbed.

All of the above topics have been discussed extensively above; the discussion need not be repeated here.

7.6 INQUERY

INQUERY [Callan et al., DB&ExSysApp, 1992] is a probabilistic IF research system, developed at the University of Massachusetts, and “designed for experiments with large [text] databases.” INQUERY is based on the inference network model, which is discussed in an earlier section of this paper. Here, a brief overview of the INQUERY system will be given.

[The inference net model] implemented in the INQUERY system emphasizes filtering based on combination of evidence. Different text representations (such as words, phrases, paragraphs, or manually assigned keywords) and different versions of the query (such as natural language and Boolean) can be combined in a consistent probabilistic framework. [Callan et al., IP&M, 1995]
The INQUERY document parser analyzes the overall structure of the document, converts it to a canonical format, and identifies those sections to be indexed. Then, it performs lexical analysis to extract words, fields, etc., “recognizes stop-words, stems the words, and indexes the words for filtering.” Stop-words are not indexed but they are retained in the text so that subsequent textual analysis (syntactic analysis, feature recognition [see below]) may make use of them.” [Callan et al., IP&M, 1995] (See discussion above of Riloff’s work on the use of stop-words in semantic analysis.)

INQUERY feature recognizers (earlier called “concept recognizers”) “search text for words that correspond to simple semantic components,” e.g., numbers, dates, person names, company names, country names, U.S. cities, etc. The set of feature recognizers is open-ended. The number recognizer maps multiple forms of a number, e.g., 1 million, or 1000000, or 1,000,000, into a common, canonical format. The company name recognizer “looks for strings of capitalized words that end with one of the legal identifiers that often accompany company names (e.g., “Co,” “Inc,” “Ltd,”…”).” [Callan et al., DB&ExSysApp, 1992] In addition to such heuristics, databases, e.g., of known person or city names, are used.

“Queries can be made to INQUERY using either natural language, or a structured query language.” Query pre-processing includes stop-phrase removal, stop-word removal, stemming, and “conversion of hyphenated words and sequences of capitalized words into proximity constraints”. [Callan et al, IP&M, 1995] (The latter corresponds to the fact that, e.g., hyphenated words, are indexed as separate words but with their textual positions retained.)

Queries are expanded using an INQUERY tool called PhraseFinder, which builds a database of “pseudo-documents” based on a given collection of actual documents.

Each pseudo-document represents a concept, in this case a noun sequence, that occurs in the document collection. The “text” of the pseudo-document consists of words that occur near the concept in the document collection. For example, a PhraseFinder document for a Wall Street Journal collection contains an amnesty program pseudo-document indexed by 1986, act, control, immigrant, law … A query is expanded by evaluating it against a PhraseFinder database, selecting the top ranked concepts, weighting them and adding them to the query. [Callan et al, IP&M, 1995]
Concepts are ranked by performing a conventional match between the given query and the collection of pseudo-documents. The concepts associated with the highest ranking pseudo-documents are added to the query. The assumption is that concepts, e.g., noun sequences, co-occurring with some of the same terms in both a query and a document context, e.g., a pseudo-document, may be related semantically. Hence, if “amnesty program” co-occurs with “immigrant” and “law” in both a query and a document, it probably refers to the same entity in both places. Evidently, PhraseFinder performance is sensitive to how near to a concept a term must be to be included in its “pseudo-document.”

INQUERY 3.0 [INQ 3.0, ACSIOM, 1995] supports a number of structured operators. Operators can be nested within operators. Each of these operators returns a belief value, e.g., a score or weight, or a proximity list that can be converted into a belief list. The beliefs returned by the clauses of a structured query contribute to the belief that the given document satisfies the information need expressed by the total query in which these clauses occur. The primary boolean operators, #and and #or, are extended booleans. (By convention, all the INQUERY operators start with #.) For example, the interpretation of #and is that “[t]he more terms contained in the AND operator which are found in a document, the higher the belief value of that document.” Plainly, extended booleans are more in tune than strict booleans with the probabilistic nature of INQUERY, since they return a degree of satisfaction of the boolean condition rather than an all- or-nothing true or false. There are also some strict boolean operators: #band, and #bandnot. (The latter is satisfied if the first term is in a given document and the second term is not.) There is also a #not command which awards higher belief to a document that does not contain its operand terms.

There are several proximity operators: The Unordered Window operator #uwN requires that its operand terms co-occur in the document in any order but within a window of $N$ words. The Ordered Distance operator, #odN, is similar except that in addition to co-occurring within a window of $N$ words, the terms must occur in the specified order within that window. The #phrase operator evaluates terms to determine if they occur together frequently in the collection. If they do [i.e., if the phrase occurs frequently in the collection as a phrase], then they are required to occur in the specified order within a three word window, i.e., #phrase is evaluated as #od3. Otherwise, #phrase reduces to
#sum (see below); the more operand terms a given document contains, regard- less of their proximity, the higher belief (rank) the document receives. The #passage operator is similar to #uwN except that instead of looking for an \(N\)-word window in which all the specified terms occur, it looks for the “best” passage, i.e., the \(N\)-word window that most nearly satisfies the specified operands; “the document is rated based on the score of its best passage.”

A synonym operator, #syn, allows the user to specify that its operands are to be treated as synonymous terms.

There are two sum operators: Sum (#sum) and Weighted Sum (#wsum). The former sums the beliefs of its operands. The latter takes a set of weight/operand pairs and computes the weighted sum, i.e., \(W_1*T_1 + W_2*T_2 + \ldots + W_n*T_n\). (One can also specify a scaling factor, an overall weight, for the entire weighted sum.) A weighted sum allows the user to say that some operands, e.g., terms, are more important than others, e.g., contribute more by their presence in a document to the belief that the document satisfies the information need expressed by the query. The beliefs computed by these sum operators are normalized, e.g., the weighted sum is divided by the sum of the specified weights.

Note that, in general, an operator such as a weighted sum can be attached to any node of the inference network. If it is attached to a node of the query network, it is a component of the query, and the operands are the parents of the given node, which may be lower level (more nested) query components, or term representation nodes for the document under consideration. If a given document is instantiated, each term is assigned a belief, e.g., it may be a \(tf*idf\) weight or a one (for strict boolean evaluation). The evaluation of the query will then evaluate the specified weighted sum which will then sum the beliefs of each query term in the document, weighting them by both the belief in the term, and the weight assigned by the user to the given term in the query. The result may be equivalent to a cosine similarity calculation, or a strict boolean evaluation, or an extended boolean evaluation, or something more complex, depending on the operator and weights assigned to each node.

Table 3 below summarizes some of the important characteristics of the IF systems described in this section, as determined by their documentation.
### Table 3: Characteristics of IF Systems

<table>
<thead>
<tr>
<th>Feature</th>
<th>Lexis/Nexis</th>
<th>Dialog</th>
<th>Dow Jones</th>
<th>Topic</th>
<th>SMART</th>
<th>INQUERY</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Strict Boolean</strong></td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>N</td>
<td>Y</td>
</tr>
<tr>
<td><strong>Extended Boolean</strong></td>
<td>N</td>
<td>N</td>
<td>N</td>
<td>Y</td>
<td>N</td>
<td>Y</td>
</tr>
<tr>
<td><strong>Proximity Operator</strong></td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>N</td>
<td>Y</td>
</tr>
<tr>
<td><strong>Terms/Keywords</strong></td>
<td>N</td>
<td>N</td>
<td>N</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td><strong>Wild Card Terms</strong></td>
<td>Y</td>
<td>Y</td>
<td>N</td>
<td>Y</td>
<td>N</td>
<td>N</td>
</tr>
<tr>
<td><strong>Stemming</strong></td>
<td>N</td>
<td>N</td>
<td>N</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td><strong>Phrase</strong></td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td><strong>User-assigned Weights</strong></td>
<td>N</td>
<td>N</td>
<td>N</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td><strong>NL Query</strong></td>
<td>Y</td>
<td>N</td>
<td>N</td>
<td>N</td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td><strong>Ranked Output</strong></td>
<td>Y</td>
<td>N</td>
<td>N</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td><strong>Probabilistic</strong></td>
<td>N</td>
<td>N/A</td>
<td>N/A</td>
<td>N</td>
<td>N</td>
<td>Y</td>
</tr>
<tr>
<td><strong>Document ranking by Similarity</strong></td>
<td>Y</td>
<td>N/A</td>
<td>N/A</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
</tr>
</tbody>
</table>

a. Exact Match (does not produce ranked output).
b. Does not require exact match (produces ranked output).
c. Roughly equivalent to a strict boolean with all terms connected by OR’s.
d. ACCRU operator is roughly equivalent to a keyword vector but uses a similarity function different from cosine similarity.
e. Unweighted sum operator (#sum) roughly equivalent to term vector.
f. Wildcard terms can be used as an alternative to stemming, or in addition to stemming. g. Stemming can be requested explicitly for any given word.
h. Automatic stemming
i. Automatic stemming
j. May be subject to semantic analysis, thesaurus expansion, or merely reduced to a term vector.
k. Ranked output may be produced by evaluating extended boolean, query/document similarity, or probability that document satisfies query.