What is Boolean Logic?

Boolean Logic, also known as Boolean algebra, is a method for describing a set of objects or ideas. It was invented in the 1800s by George Boolean, an English math teacher, but it is has become part of the foundation for controlling computers. The binary 0 and 1 states are naturally related to the true and false logic variables. By inserting key words, or operators, between terms in a statement you can describe the relationship among the terms.

Why do we need Boolean Logic?

One of the most recent uses of Boolean logic is in search engines. By using the operators AND, OR and NOT you can effectively communicate exactly what you are looking for to the search program. Without a standard language such as this, everyone may have a different way of expressing the same search request. Without this precision, searches would be less than effective, and retrieving valuable information would be even more difficult. The better you know Boolean logic the more effective your searches will be and the quicker you will be able to find what you are looking for.

Where do we use Boolean Logic?

The most practical places for you to use Boolean logic are in search engines. An engine is a search program located on a powerful computer. The engine allows you to search its database of the Internet. The following are links to engines that allow users to use Boolean logic. There are many more engines that allow Boolean but these are generally regarded as the best. If you have any doubts, you can check the engine's help section to learn specific syntax required for queries.

- **AltaVista**: A fast and powerful engine with advanced Boolean capabilities. Its database (full text of Websites) is best when looking for hard-to-find information (www and usenet).
- **HotBot**: A powerful search engine that allows basic Boolean features. It also provides a pleasing user interface (www and usenet).
- **OpenText**: Another powerful search engine with a full set of Boolean operators. Its strength is the ease of searching its full text database of the Web (www only).
- **Lycos**: One of the search engine pioneers since 1994. It provides good listings of search results, and allows users to search more than just the Web (www, FTP, Gopher).

Boolean Searching on the Internet

When you use an Internet search engine, the use of Boolean logic may be manifested in three distinct ways:

1. Full Boolean logic with the use of the logical operators
2. Implied Boolean logic with keyword searching
3. Boolean logic using search form terminology

- **Full Boolean logic with the use of the logical operators**
  - This is classic Boolean searching. However, few search engines nowadays offer the option to do full Boolean searching with the use of the logical operators. It is more common for them to offer simpler methods of constructing search statements, specifically implied Boolean logic and search form terminology. These methods are covered below.
  - If you want to construct searches using Boolean logical operators, you will need to experiment with search engines and see what happens. You can try some of the search statements shown below. Keep in mind that the search engine might require that the Boolean operators be typed in CAPITAL LETTERS. Don't forget that most search engines provide help pages that explain the kind of searching you can do on their sites.

- **Implied Boolean logic with keyword searching**
  - **KEYWORD SEARCHING** refers to a search type in which you enter words in a search box representing the concepts you wish to retrieve. Boolean operators are not used.
  - **IMPLIED BOOLEAN LOGIC** refers to a search in which symbols are used to represent Boolean logical operators:

    \[ (+ \text{ refers to AND}) (|| \text{ refers to OR logic}) (- \text{ refers to NOT logic}) \]
In this type of search, the absence of a symbol is also significant, as the space between keywords defaults to either OR logic or AND logic. Nowadays, virtually all general search engines on the Web default to AND logic. In other words, when you type words into a search box and generate your search, Boolean AND logic is going on behind the scenes.

Implied Boolean logic is so common on Web search engines that it can be considered a de facto standard.

- **Boolean logic using search form terminology**
  - Many search engines offer an advanced search page with a search form which allows you to choose the Boolean operators from a menu. Usually the logical operator is expressed with substitute terminology rather than with the operator itself.

**OR**

OR is a Boolean operator used to broaden your search by retrieving any, some, or all of the keywords used in the search statement. Since the Web is already huge, using OR helps you make sure you aren't missing anything valuable.

**Query: College OR University  (I would like information about college.)**

- In this search, we retrieve records in which AT LEAST ONE of the search terms is present. We are searching on the terms college and also university since documents containing either of these words might be relevant.
- This is illustrated by:
  1. The shaded circle with the word college representing all the records that contain the word "college"
  2. The shaded circle with the word university representing all the records that contain the word "university"
  3. The shaded overlap area representing all the records that contain both "university" and "college"

OR logic is most commonly used to search for synonymous terms or concepts.

**Here is an example of how OR logic works:**

<table>
<thead>
<tr>
<th>Search terms</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>college</td>
<td>17,320,770</td>
</tr>
<tr>
<td>university</td>
<td>33,685,205</td>
</tr>
<tr>
<td>college OR university</td>
<td>33,702,660</td>
</tr>
</tbody>
</table>

OR logic collates the results to retrieve all the unique records containing one term, the other, or both. The more terms or concepts we combine in a search with OR logic, the more records we will retrieve.

<table>
<thead>
<tr>
<th>Search terms</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>college</td>
<td>17,320,770</td>
</tr>
<tr>
<td>university</td>
<td>33,685,205</td>
</tr>
<tr>
<td>college OR university</td>
<td>33,702,660</td>
</tr>
<tr>
<td>college OR university OR campus</td>
<td>33,703,082</td>
</tr>
</tbody>
</table>
AND

And is a Boolean operator used to narrow your search by ensuring that all keywords used appear in the search results. Since the Web is already huge, it is important you use AND effectively.

Example#1

Query: I'm interested in the relationship between poverty and crime.

Poverty AND Crime

- In this search, we retrieve records in which Both of the search terms are present.
- This is illustrated by the shaded area overlapping the two circles representing all the records that contain both the word "poverty" and the word "crime."
- Notice how we do not retrieve any records with only "poverty" or only "crime."

Here is an example of how AND logic works:

<table>
<thead>
<tr>
<th>Search terms</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>poverty</td>
<td>783.447</td>
</tr>
<tr>
<td>crime</td>
<td>2,962,165</td>
</tr>
<tr>
<td>poverty AND crime</td>
<td>1,677</td>
</tr>
</tbody>
</table>

The more terms or concepts we combine in a search with AND logic, the fewer records we will retrieve.
**NOT**

Not is a Boolean operator used to eliminate an unwanted concept or word in your search statement.

**Example**

*Query: I want to see information about pets, but I want to avoid seeing anything about cats.*

**Pets NOT Cats**

- In this search, we retrieve records in which ONLY ONE of the search terms is present.
- This is illustrated by the orange area with the word pets representing all the records containing the words "pets".
- No records are retrieved in which the word "cats" appears, even if the word "pets" appears there, too.

**Here is an example of how NOT logic works:**

<table>
<thead>
<tr>
<th>Search terms</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>pets</td>
<td>4,556,515</td>
</tr>
<tr>
<td>cats</td>
<td>3,651,252</td>
</tr>
<tr>
<td>pets NOT cats</td>
<td>81,497</td>
</tr>
</tbody>
</table>

**NOTE:** NOT logic excludes records from your search results. Be careful when you use NOT: the term you do want may be present in an important way in documents that also contain the word you wish to avoid.

**Truth Table**

A truth table shows the resulting value when a logical operator is used to join two propositions, forming a new, complex proposition.

Suppose the two propositions being joined are P and Q. Each of these propositions will have two possible truth values: true, or false. This gives us four possible combinations. These are represented on a table, as follows:

<table>
<thead>
<tr>
<th>P</th>
<th>Q</th>
</tr>
</thead>
<tbody>
<tr>
<td>T</td>
<td>T</td>
</tr>
<tr>
<td>T</td>
<td>F</td>
</tr>
<tr>
<td>F</td>
<td>T</td>
</tr>
<tr>
<td>F</td>
<td>F</td>
</tr>
</tbody>
</table>
In the space to the right, a complex proposition is displayed. Beneath the complex proposition are the truth values which result given the four possible truth values of $P$ and $Q$. For example, here is the truth table for the complex proposition $P \text{ AND } Q$ and $P \text{ OR } Q$:

<table>
<thead>
<tr>
<th>$P$</th>
<th>$Q$</th>
<th>$P \text{ AND } Q$</th>
<th>$P \text{ OR } Q$</th>
</tr>
</thead>
<tbody>
<tr>
<td>T</td>
<td>T</td>
<td>T</td>
<td>T</td>
</tr>
<tr>
<td>T</td>
<td>F</td>
<td>F</td>
<td>T</td>
</tr>
<tr>
<td>F</td>
<td>T</td>
<td>F</td>
<td>T</td>
</tr>
<tr>
<td>F</td>
<td>F</td>
<td>F</td>
<td>F</td>
</tr>
</tbody>
</table>

Notice: that the complex proposition may be true or false depending on the different truth values of $P$ and $Q$. Thus, if we know what the truth values of $P$ and $Q$ are, we know what the truth value of $P \text{ AND } Q$ or $P \text{ OR } Q$ is.

**Exact Matching**

The user would always like an exact and perfect answer to the query posed to the system. This, however, is only possible when the query is exactly posed and the data in the database can be definitely identified as responding to the query or not. The most obvious exact match situations arise with numerical or business databases. The terms in such databases are organized into definite fields, and it is possible to determine exactly whether the value of a given term matches the value specified in the query.

**Similarity Measures**

When the query and the document representations are similar, the query can be considered as a point in the document space. In such case, the mental image is that of relevant documents being clustered near the query point. The evaluation function defines a contour that separates the relevant documents from the irrelevant ones, or a series of contours corresponding, as they broaden, to documents of increasingly less relevance to the query.

The determination for each document is whether it is relevant to a given query. As relevance is ultimately in the mind of the user, it is difficult to measure directly. Retrieval systems must rely primarily on measurements that can be made within the system on the basis of the document and query representations. As their cores, most retrieval systems equate relevance, in the form of topicality, with lexical similarity.

Many measures of similarity are calculated as values within some closed interval, say $[0,1]$, with the interpretation that high value in the interval (1) represents the greatest similarity and the low value (0) represents the least similarity. There is, however, no reason why the scale must be so chosen. A similarity scale can be chosen to be open-ended, or to have 0 as the measure of greatest similarity.

**Proximity Matching**

Many retrieval systems permit the user to apply proximity criteria in the retrieval process. That is, the user may be able to specify that terms $A$ and $B$ occur within three words of each other, within the same sentence, or some similar criterion. Proximity criteria are based solely on the occurrence or nonoccurrence of terms. It is possible to introduce concepts like the conditional probability that term $B$ occurs given that term $A$ has occurred, but these are generally based on simple co-occurrence, not on a proximity measure. Thus proximity criteria can be used independently to further refine the set of documents identified by one of the other matching methods.

One modification of proximity criteria which can increase effectiveness is to use phrases rather than simple word proximity. The incorporation of the phrases into the similarity judgement process requires that a given set of words occur in a highly specific sequence. This focuses the search criteria more sharply but has the negative effect of discounting expressions that are equivalent to, but different from, the specified phrase.

Another modification, less strict than the use of phrases, is the use of ordered proximity to aid in the retrieval decision. Clearly, the order in which words occur affects their interpretation. One classic example with a clear distinction
in the interpretation is junior college versus college junior. Although ordered proximity will distinguish between these two phrases, selecting only documents with desired interpretation, it will sometimes cause relevant documents to be missed.

Truncation

Truncation allows you to widen a search on a textual or name field by including truncation symbols (* or ?) in your search term. Truncation uses one or more question marks at the end of a search term to search on a word stem or incomplete term. Word stems should be at least three characters long. The * symbol is used to match any number of characters and the ? symbol is used to match a single character.

- **Open Truncation**
  You can truncate a search term by appending a question mark to the end of a term. For example, enter `transport?` and all records containing the word stem transport are retrieved, including such terms as transportation, transported, transporting, and transportable.

- **Restricted Truncation**
  To restrict truncation, enter one question mark for each character you wish to allow. For example, enter `transport??` and all records containing the word stem, transport, plus three characters are retrieved including transport and transporting. A question mark followed by a space and another question mark retrieves records containing the word stem and one additional character following the word stem. This strategy is frequently used to retrieve a specific search term in both its singular and plural forms.

- **Internal Truncation**
  A question mark can replace a character in a search term. For example, to retrieve records that contain the terms woman and women, enter `wom?n`. More than one question mark can be inserted in a term, however, the terms retrieved will have one character for each question mark used. For example, enter `psych??ist` and records containing the terms psychologist and psychiatrist are retrieved. You can also use internal truncation with open and restricted truncation. Internal truncation cannot be used to retrieve variations in spelling that result in words of different lengths, for example, color and colour. To retrieve variation in spelling, enter both terms separated by the OR operator, for example, enter color or colour.

**Examples:**

1. Searching for variants of words.
   - `DEVELOP*` to find Developing, Development, etc.

2. Searching for a word where a single character may vary.
   - `WOM?N` to find Women or Woman.

3. Right hand truncation is used to search for words that end with a stem. For example:
   - `*OXIDE` to find Dioxide, Hydroxide, Oxide, Peroxide, etc.
   - Please note that the amount of time a right hand truncated search takes to evaluate depends on the number of terms in the field being searched

Stemming

Stemming is related to substring, but involves an even greater understanding of the language. Rather than requiring the user to enter root terms in a query, stemming involves trimming a query word to its root and then looking for other words that match the same root.

**Example:**

The word "wallpaper," for example, has "wall" as its root words; so does "wallboard", which the user might never have entered as a separate query. When a user enters "wallpaper," a stemmed search might serve up unwanted additional references to "wallflower," "wallbanger," "Wally," and "walled city," but it would also catch "wall" and "wallboard" and probably provide useful information that way.
Reference

- Robert R. Korfhage. "Information Storage and Retrieval". Publisher: John Wiley & Sons