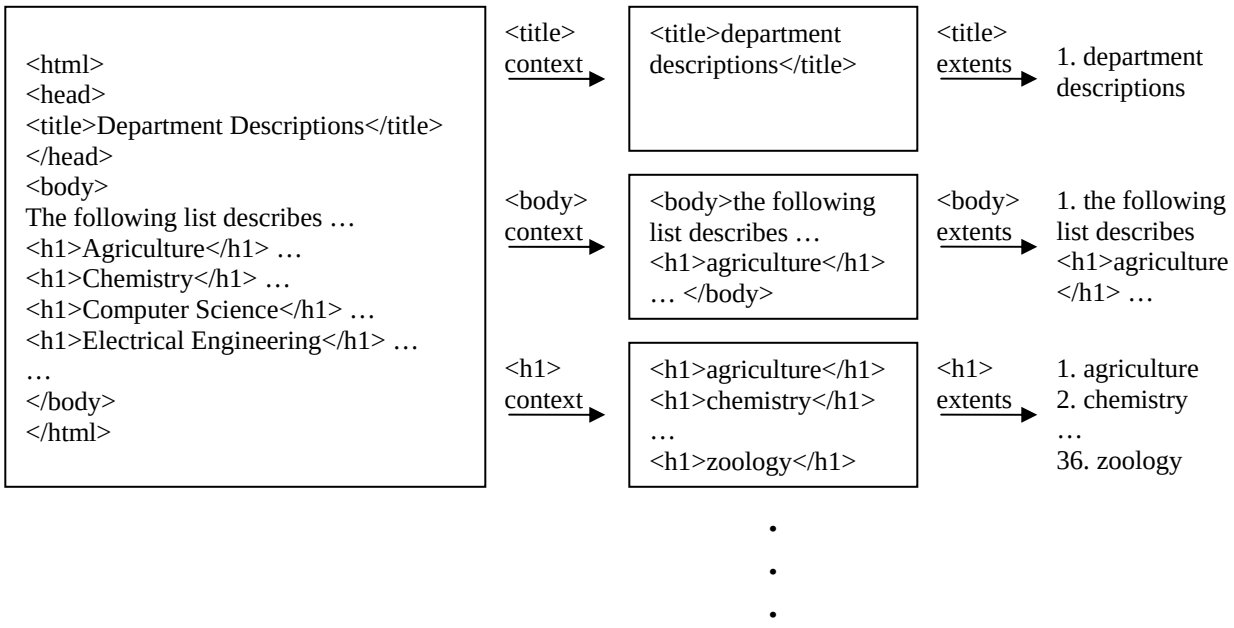
Indri Query Language
Quick Reference Guide

By: Don Metzler (metzler@cs.umass.edu)
(Last revised: 20 September 2004)

Document Representation

Indri handles both structured (text with fields) and unstructured documents. The following diagram illustrates how documents are represented within Indri.



A *document* is viewed as a sequence of text that may contain arbitrary tags. In the above diagram, the document consists of text marked up with HTML tags.

For each tag name *T* within a document (i.e. title, body, h1, etc), the *context* of *T* is defined to be all of the text and tags that appear within tags of type *T*. In the example above, all of the text and tags appearing between <body> and </body> tags defines the body context. A single context is generated for each unique tag name. Therefore, a context defines a subdocument. Note that certain word occurrences may appear in many contexts since tags may be nested. It is also the case that there may be nested contexts. For example, within the <body> context there is a nested <h1> context made up of all of the text and tags that appear within the body context and within <h1> and </h1> tags.

Finally, each context is made up of *extents*. An extent is a sequence of text that appears within a single begin/end tag pair. In the example document above, the <h1> context consists of extents “agriculture”, “chemistry”, etc. Both the title and body contexts contain only a single extent because there is only a single pair of <title>…</title> and <body>…</body> tags, respectively. The number of extents for a given tag type *T* is determined by the number of sequences of the form: <*T*>…</*T*>.

Terms

There are multiple ways to represent terms in Indri. Terms are the basic query elements that can be used to compose more complicated structured queries in combination with the other operators described later. The following table describes each representation, gives an example, and describes how it is matched within a document.

Type	*Example*	*Matches*
Stemmed term	Dog	All occurrences of *dog* (and its stems)
Surface term	"dogs"	Exact occurrences of *dogs* (without stemming)
Term group (synonym group)	<"dogs" canine>	All occurrences of *dogs* (without stemming) or *canine* (and its stems)
Stemmed base64 encoded		Same as dog
Surface base64 encoded		Same as "dogs"
Extent	#any:h1	Occurrences of *h1* extents

Proximity Expressions

Proximity expressions allow different, more powerful methods for matching terms. The following table describes each, where each e_i is a well-formed proximity expression.

Type	*Example*	*Matches*
#odN(e_1 … e_m) or #N(e_1 … e_m)	#od5(dog cat) or #5(dog cat)	All occurrences of *dog* and *cat* appearing ordered within a window of 5 words
#uwN(e_1 … e_m)	#uw5(dog cat)	All occurrences of *dog* and *cat* that appear in any order within a window of 5 words
#phrase(e_1 … e_m)	#phrase(#1(willy wonka) #uw3(chocolate factory))	System dependent implementation (defaults to #odm)
#syntax:xx(e_1 … e_m)	#syntax:np(fresh powder)	System dependent implementation

The system dependent implementation for #syntax:xx can be designed to match strings that have been tagged with xx. Similar system-specific lexical operators that may be implemented are #person that recognizes personal names and #date:xx that recognizes dates that satisfy some optional range requirement specified by xx.

Date / Numeric Expressions

Numeric fields may also be matched using a number of specialized operators. The following table describes these operators.

Example	*Matches*
#date:after(1-JAN-2000)	Date extents with dates after January 1, 2000.
#date:before(1-JAN-2000)	Date extents with dates before January 1, 2000
#date:between(10/20/1980 10/20/1981)	Date extents with dates between October 20, 1980 and October 20, 1981
#less(READINGLEVEL, 5)	READING-LEVEL extents with value less than 5
#greater(READING-LEVEL, 1)	READING-LEVEL extents with value greater than 1
#between(READING-LEVEL, 2, 4)	READING-LEVEL extents with values between 2 and 4

Recognized date formats:

- 11 january 2004
- 11-JAN-04
- 11-JAN-2004
- January 11 2004
- 01/11/04
- 01/11/2004

Context Restriction

Context restrictions allow us to restrict term and proximity expression matching to one or more contexts. This assumes that the indexed corpus is structured (contains tags). Context restriction is defined syntactically as: $e.t_1,\ldots,t_j$, where e is a term or proximity expression and $t_1,\ldots,t_j$ are tag names.

Example	*Matches*
dog.title	All occurrences of *dog* appearing in the *title* context
dog.title,paragraph	All occurrences of *dog* appearing in both a *title* and *paragraph* contexts (may not be possible)
<dog.title dog.paragraph>	All occurrences of *dog* appearing in either a *title* context or a *paragraph* context
#5(dog cat).head	All matching windows contained within a *head* context

Context Evaluation

Context evaluation allows us to specify the context in which matching and scoring is done. The syntactic form is: $e.(t_1,\ldots,t_j)$ where e is a term or proximity expression that is optionally context restricted and $t_1,\ldots,t_j$ are tag names. Here, e is matched and scored using only the text contained in the concatenation of contexts $t_1,\ldots, ,t_j$.

Example	*Evaluated*
dog.(title)	The term *dog* evaluated using the *title* context as the document
dog.(title, paragraph)	The term *dog* evaluated using the concatenation of the *title* and *paragraph* contexts as the document
dog.figure(paragraph)	The term *dog* restricted to *figure* tags within the *paragraph* context.

Belief Operators

Belief operators allow beliefs about terms, proximity expressions, and other complex expressions to be combined. The following table lists the belief operators implemented in Indri. The table also gives the name of the corresponding INQUERY operator along with a rough "soft Boolean" interpretation of the operator.

Indri	*INQUERY*	*Example*	*Interpretation*
`#combine`	`#and`	`#combine`(dog cat)	AND
`#weight` `#wand`	N/A	`#weight`(1.5 dog 1.0 cat) `#wand`(1.5 dog 1.0 cat)	weighted AND
`#or`	`#or`	`#or`(dog cat)	OR
`#not`	`#not`	`#combine`(cat `#not`(dog))	NOT
`#sum`	`#sum`	`#sum`(dog cat)	AVERAGE
`#wsum`	`#wsum`	`#wsum`(1.5 dog 1.0 cat)	weighted AVERAGE
`#max`	`#max`	`#max`(dog cat)	MAX

The weighted operators allow different amounts of importance to be assigned to expressions. Furthermore, it is suggested that `#combine` and `#weight` be used for most queries. Only experienced users who understand the implications should make use of the remaining operators.

Extent Retrieval

By default, belief operators return a single score per document. However, using the syntax `#op`$[t_1,\ldots,t_j]$(e), where `#op` is a belief operator, e is a term or proximity expression and $t_1,\ldots,t_j$ are tag names, Indri returns scores for each extent of type $t_1,\ldots,t_j$. The expression `#op`(e) is evaluated once per extent in the context defined by the concatenation of contexts $t_1,\ldots,t_j$. This list of scores is then returned.

Example	*Evaluated*
#combine[section](dog canine)	Each *section* extent is scored according #combine(dog canine). Returns a list of *section* extents.
#combine[title, section](dog canine)	Same as previous, except evaluated for each *title* and *section* extent. Returns a list of *title* and *section* extents.
#sum(#sum[section](dog))	Returns a single score that is the #sum of the scores returned from #sum(dog) evaluated for each *section* extent.
#max(#sum[section](dog))	Same as previous, except returns the maximum score.

Filter Operators

Indri offers two filtering operators. *Filter require*, denoted by #filreq(*e q*), only scores documents that match expression *e* at least once. Matching documents are then scored according to *q*. The second operator is *filter reject*. It takes the form #filrej(*e q*). Here, only those documents that do not match expression *e* will be scored (again according to *q*).

Note: Expression *e* is required to be a term, proximity, or numeric expression.

Example	*Evaluated*
#filreq(sheep #combine(dolly cloning))	Only consider documents containing the term *sheep* and rank them according to the query #combine(dolly cloning)
#filrej(parton #combine(dolly cloning))	Only consider documents that do not contain the term *parton* and rank them according to the query #combine(dolly cloning)

External Priors

External prior probabilities over documents can be incorporated into a query via #prior(*name*), where *name* is either the name of a file containing a list of priors or the name of a function to be computed at runtime.

Named Language Models

Under construction.