Comparing the Indri Search Engine to Traditional Text Retrieval Methods

Andrew Fitch

Introduction to the INDRI Search Engine

Text retrieval is one the most important areas of study in computer science. Anytime someone searches for something in google or attempts to find a particular article in a research database, they are relying on a variety of search algorithms and ranking functions that make their queries possible. Many state-of-the-art search engines and ranking algorithms support relatively simple queries that execute quickly but do not consider details such as the order of words in the query when ranking results. Indri, a search engine developed by the University of Massachusetts and Carnegie Melon University as part of the Lemur Project, aims to enable users to create more complex queries in an effort to achieve better tailored results[1].

The principal goals of the Indri Search engine can be summed up in the following two points:

- Enable complex queries for text retrieval at different levels of granularity and across multiple types of documents.[2]
- Efficiently support the execution of these complex queries.[3]

In the next two sections of the review, I'll dive into each of these goals discussing how INDRI was able to achieve them and relating them to the text retrieval algorithms and methods that have been covered as part of the CS 410 Curriculum at the University of Illinois, Urbana Champaign.

The Indri Search Engine Query Language

In the CS, we studied the Okapi BM25 ranking model, among others. In our lectures, we learned that to achieve fast and performant results with regards to index creation and search execution, a complexity trade-off had to be made. With regards to the models covered in our class, this complexity trade-off came in the form of limiting the query language and simplifying the document representation in the form of the "Bag-of-Words" language model. In the "Bag-of-words" language model, every word in a document is considered in isolation, with no regard to surrounding words. Thus, the query "happy dog" and "Dog happy" are identical under this model.[4]

There are times when the "Bag-of-Words" language model does not accurately address our goals. This is especially the case when we want to create domain-specific queries. For instance, what if you want to query your domain for just paragraphs with a certain name in them from articles written before 2002? In this case, typical search strategies would come up short, there

[1] The Lemur Project, INDRI Language Modeling Meets Inference Networks. https://www.lemurproject.org/indri/
[2] Indri: A language-model based search engine for complex queries. 1. http://ciir.cs.umass.edu/pubfiles/ir-407.pdf
[3] Indri: A language-model based search engine for complex queries. 1
[4] UIUC CS 410 Coursera Lecture Videos: 1.1

is no way to represent this type of complexity – varying definitions of documents and detailed search criterion, using search engines build on top of Okapi BM25. This is where the INDRI steps in and provides us a tool to answer these more complicated queries.

Like text retrieval methods that rely on Okapi BM25, INDRI search can handle simple keyword queries, however it is also able to handle more complex queries, like the one described above, using its expanded query language. The operators in the INDRI query language are based on the Inquery operators which was an earlier attempt at creating a search engine to handle complex queries.[5] Some of the extra operations supported by the Indri query language include:

- Order operator: ensures terms in the query are found in the provided order.[6]
- Window operator: terms in the query must close enough together to fit in the window.[7]
- Combine operator: Certain terms must appear together, and custom weighting can be applied to terms[8]
- Field operators: Search within certain structures in a document, like a paragraph.[9]
- Date and numeric retrieval: special operations for numeric operators such as "greater than" or "less than".[10]

These operators go far beyond the Bag-of-Words representation of a language model and allow users of Indri to make powerful and precise queries to address specific retrieval problems.

Supporting Query Execution: Implementation of Indri

The Bag-Of-Words language model helps make text retrieval algorithms simple and efficient to implement and support. Consider, again, the Okapi BM25 text retrieval algorithm. Since this algorithm processes words in isolation, the only data needed to evaluate the algorithm is a mapping of words to words counts in a particular document and the length of documents. These two statistics can easily be stored via a reverse index generated efficiently via a distributed, Map-Reduce algorithm. Thus, the relative simplicity of the Okapi BM25 algorithm and its inputs allow for fast and efficient execution of queries.[11]

The INDRI search engine, with its more complex query language, cannot rely on the same techniques that worked so well for text retrieval methods that relied on the Bag-of-Words language model. For instance, in most standard traditional text retrieval methods, a document is represented as feature vectors the length of the vocabulary containing term counts for the document. However, since Indri tries to model more text structures than just the term counts (such as whether a term appears at the end of a sentence or whether two terms appear

[5] Indri: A language-model based search engine for complex queries. 3.1
[6] Indri: A language-model based search engine for complex queries. 3.1
[7] Indri: A language-model based search engine for complex queries. 3.1
[8] The Lemur Project, Indri Query Language Quick Reference. https://www.lemurproject.org/lemur/IndriQueryLanguage.php
[9] Indri: A language-model based search engine for complex queries. 3.2
[10] Indri: A language-model based search engine for complex queries. 3.4
[11] UIUC CS 410 Coursera Lecture Videos: 2.4 – 2.6

together), it needs a more complex way of representing a document. To solve this, Indri stores documents a sets of multiple binary feature vectors, as opposed to just a single feature vector. Each feature vector in the set represents a different fact about a document we wish to represent.[12]

Note that the representation of a document depends on the questions being asked. For every condition in our query, there needs to be an additional feature vector encoding that data. This means that for different queries, we potentially need to re-index some of the documents to include different data.[13] From a performance standpoint, this means Indri works best when a user is making a variety of similar types of queries on the collection. When Indri must re-create indexes to handle new queries, it is able to leverage multi-threading to increasing indexing speed.[14]

The actual retrieval process of Indri relies on the Inference network model. In the inference network model, queries and documents are composed of concepts rather than terms. A directed acyclic graph is constructed such that the top of higher nodes represents more concrete structures in the document and lower nodes/child nodes correspond to concept nodes that may be of interest to queries.[15] Indri will then create query nodes from users and attempt to answer the question: what is the probability a document is relevant to a given query?[16] A multiple-Bernoulli model is used to compute the prior for the likelihood estimate that actually determines document ranking (the multiple Bernoulli model must be used because, recall that documents are represented as sets of feature vectors as opposed to just a single feature vector)[17].

Final Remarks

The Indri query language is clearly a powerful query language that can help users address their complex text retrieval needs. However, the Indri query language is not ideal for every single scenario. To get the most out of it, the domain should be relatively specific (e.g., a domain of news articles) and the user should have knowledge of that domain. The complex query language can also make Indri cumbersome from a user perspective. When the Indri query language is used, there should be some sort of UI that helps users constructs their queries and guides them towards interesting query criteria (which requires extra work by the builder of the document collection). Lastly, the Indri query language is too complex for most search scenarios. Most text retrieval problems are likely addressed by more conventional retrieval methods such as Okapi BM25 and Indri should be reserved for very specific use cases. Thus, while the Indri search engine is an exciting development in the world of text retrieval, it is not setting the standard for or replacing existing text retrieval models and strategies.

[12] Indri: A language-model based search engine for complex queries. 2.1
[13] Indri: A language-model based search engine for complex queries. 5
[14] Indri: A language-model based search engine for complex queries. 4.2
[15] Indri: A language-model based search engine for complex queries. 2.4
[16] Indri: A language-model based search engine for complex queries. 2.5
[17] Indri: A language-model based search engine for complex queries. 2.2