Structured Document Retrieval and Ranking System using N-gram Language Models

Himanshu S. Bhatt and Samarth Bharadwaj

Abstract

The primary focus of this document is to discuss plan of action to implement n-gram language model based document retrieval model. First, language models are described based on the review article by [6]. Structured documents such as HTML, XML documents contain fields that are usually identified by tags. In order to optimally query these documents, the query must be structured (such as XPath). We propose to implement Probabilistic Retrieval Model for structured documents (PRM-S) [2], a probabilistic language model that is able convert unstructured query to structured query. Further, n-gram language model is used to relax the independence assumption of the PRM.

I. Language Models

Language modeling system is based on the query likelihood scoring method first proposed by Ponte and Croft [4]. For each document in the collection, a language model is estimated and the documents are ranked based on the likelihood of the query according to the estimated language model. In a language model, the query is assumed to be a sample of words drawn according to a model estimated based on a document.

Query likelihood model is the basic language model, where a language model M_d is constructed from each document d. The ranking of documents is given by $P(d|q)$, where the probability of a document is interpreted as the likelihood that it is relevant to the query. Using Bayes rule:

$$p(d|q) = p(q|d)p(d)/p(q) \tag{1}$$

The language modeling approach attempts to model the query generation process and the documents are ranked by the probability that a query would be observed as a random sample from the document model. The documents are ranked using a multi-nomial uni-gram language model where the documents are the classes and each document is treated in the estimation as a separate ”language”. In a uni-gram model, each term is assumed independent of other terms. Using this:

$$P(q|M_d) = K_q \prod_t P(t|M_d)^{tf_{t,d}} \tag{2}$$

Where, K_q is the multi-nomial coefficient for the query q, given as:

$$K_q = L_d!/(tf_{t1,d}!tf_{t2,d}! \cdot tf_{tM,d}!) \tag{3}$$

The retrieval of documents based on language models can now be described as :

1) Estimating the language model M_d for each document in the collection.
2) Estimate $P(q|M_{di})$, the probability of generating the query according to each of these document models
3) Rank the documents according to these probabilities

The language model assumes that the user has a prototype document in mind and generates a query based on distinguishable words that are likely to appear in the document.

A. Estimating the query generation probability

Using the uni-gram assumption (each query term is independent for each other), the probability of producing the query given the language model M_d of document d using maximum likelihood estimation (MLE) is computed as:

$$\hat{P}(q|M_d) = \prod_t \hat{P}_{mle}(t|M_d) = \prod_{t \in q} \frac{tf_{t,d}}{L_d} \tag{4}$$

Where, M_d is the language model of document d, $tf_{t,d}$ is the term frequency of term t in document d, and L_d is the number of tokens in document d.

B. Variants of the Basic Language Modeling Approach

If for some words that do not appear in the document, $P(t|M_d)$ is estimated to be zero. This means that documents will only give a query non-zero probability if all of the query terms appear in the document. Therefore, to smooth probabilities in the document language models, we need to discount non-zero probabilities and give some probability mass to unseen words.

The basic language modeling approach (i.e., the query likelihood scoring method) can be instantiated in different ways by varying:

- Q_D (e.g., multiple Bernoulli or multi-nomial).
- Estimation methods of Q_D (e.g., different smoothing methods).
- The document prior $p(D)$.

The original proposed language model proposed by Ponte and Croft used the multiple Bernoulli models that ignore query term frequencies. However, the multi-nomial model captures the term frequency in documents (as well as the query). In multiple Bernoulli model, the presence/absence of a term is assumed to be independent of that of other terms, whereas in multi-nomial model, every word occurrence is assumed to be independent, including the multiple occurrences of the same term.

Estimation of Q_D is critical for the success of language models, and a particularly important issue is how to smooth the maximum likelihood estimate which assigns zero probability to unseen words. For this, different smoothing techniques have been proposed in literature.

Dirichlet prior smoothing: works well especially for keyword queries as it adjusts the amount of smoothing according to the length of a document. The Dirichlet prior smoothing method can be derived by using Bayesian estimation:

$$p(w|D) = \frac{c(w|D) + \mu p(w|C)}{|D| + \mu}, \tag{5}$$

where $p(w|C)$ is a background (collection) language model and μ is a smoothing parameter.

Linear Interpolation smoothing: Uses a mixture between a document-specific multi-nomial distribution and a multi-nomial distribution estimated from the entire collection:

$$\hat{P}(t|d) = \lambda \hat{P}_{mle}(t|M_d) + (1 - \lambda)\hat{P}_{mle}(t|M_c) \tag{6}$$

where $0 < l < 1$ and M_c is a language model built from the entire document collection. This mixes the probability from the document with the general collection frequency of the word.

C. *Types of language models*

To find the probabilities over sequences of terms, the chain rule is used to decompose the probability of a sequence of events into the probability of each successive event conditioned on earlier events:

$$P(t_1 t_2 t_3 t_4) = P(t_1)P(t_2|t_1)P(t3|t_1 t_2)P(t_4|t_1 t_2 t_3) \tag{7}$$

Uni-gram models: The simplest form of language model simply throws away all conditioning context, and estimates each term independently. Such a model is called a uni-gram language model:

$$P_{uni}(t_1t_2t_3t_4) = P(t_1)P(t_2)P(t_3)P(t_4) \tag{8}$$

Bi-gram models: In a bi-gram language model, the generation of a current word would be dependent on the previous word generated, thus it can potentially capture the dependency of adjacent words (e.g., phrases). Specifically, the query likelihood would be

$$P_{bi}(t_1t_2t_3t_4) = P(t_1)P(t_2|t_1)P(t_3|t2)P(t_4|t_3) \tag{9}$$

$$P(Q|D) = p(q_1|D)\prod_{1=2}^{m} p(q_i|q_{i-1}, D) \tag{10}$$

Where, $p(q_i|q_{i-1}, D)$ is the conditional probability of generating q_i after we have just generated $q_i - 1$. Such n-gram models capture dependency based on word positions. With increasing n, complexity of the models increases that makes it difficult to obtain accurate estimation of the model.

D. Structured Document Retrieval

Information retrieval systems generally work on bag of words representation of the documents and ignore the structural information of the document. A document may have intra-document structures (e.g., title vs. body) and inter-document structures (e.g., hyperlinks and topical relations), which can be used to improve the performance of the retrieval system.

In this project, we will divide each document is composed of several fields/context (such as title and body). The field structure of documents allows each field to have weights depending on the field and query-term. Each document d in the collection is composed of fields $(f_1, f_2, \cdot f_n)$. The project will assign different weights to different parts of the document. The query generation process consists of two steps:

- A part f_i is selected from the structured document D according to a selection probability $p(f_i|D)$. It can be interpreted as the weight assigned to D_i and can be set based on prior knowledge or estimated using training data.
- A query is generated using the selected part f_i. Thus, the query likelihood is given by:

$$p(Q|D) = \prod_{1=1}^{m} p(q_i|D) = \prod_{i=1}^{m}\sum_{j=1}^{k} p(q_i|D_j)p(f_j|D) \tag{11}$$

II. INTRODUCTION TO GALAGO

Galago is an open-source toolkit for development of information retrieval engine. It is based on query language from Indri and Lemur developments. This query language framework provides several advantages such as flexibility, multiple document representations, efficient implementation and formal grounding. Information on Galago and related information is obtained from Galago source wiki [1], Lemur docs [5] and developer and research blogs [3].

Query language: Query languages enable a syntactical representation of a query towards a database or a retrieval system. Galago's query language is based on Indri and Inquery query languages of the Lemur project. However, unlike other search engines, Galago's query language extensively embedded into retrieval model allowing for greater flexibility. For example, scores of retrieval of the query *hello world* are

```
combined as #combine( #text:dog() #text:cat())
```

The Galago query language is based on sequential operators, in its simple form can capture several ranking operations. Unlike other existing systems, the query language itself directly provides handles to request information such as inverted index count of a term etc. Such information that is usually available in the indexing module. The query is presented as a graphical model, with nodes containing various tags and meta-parameters such as smoothing parameters etc. Query operations of Galago query language provide information of the following categories:

- Index operators: They are processed directly from the indexes. Eg: #counts and #extents
- Proximity operators: Provide proximity information. Eg: #inside, #ordered and #unordered
- Scoring operators: The runQuery method requires scores provided by these operators. Eg: #feature:dirichlet
- Score combination operators: Combine scores from different scoring operations. Eg: #combine and #weight
- Other operators: It is possible to implement new operators by implementing StructuredIterator interface.

We now briefly discuss the various operators of Galago, focusing primarily on those which are deemed relevant to this project.

- #extents This operator provides the context/fields of the term as per the posting list.
- #count This operator provides the document count of a given term obtained from the posting list

- #prior Provides the document prior in the posting list. In the case that the prior is zero, minScore is assigned. (minScore = ln(0.0000000001))
- #lengths Provides document lengths
- #feature Provides smoothed log probabilities of a document

III. Out Approach

1) Inverted Index: We will use galago to implement the inverted index with the position listing. Using the inverted index, we can estimate the probability of a query term occurring in the document as well as the whole collection.
2) To incorporate n-gram language models: we will estimate the conditional probabilities for sequence of events conditioned on earlier events. This will form the basis for extending the model beyond the uni-gram model (where each term is independent of each other). We will analyze the affect of different smoothing techniques such as Dirichlet smoothing and linear interpolation.
3) Algorithm: We will implement the query likelihood model for semi-structured documents using n-gram language models. For each document, a language model will be constructed for different fields and will be combined with different weights.
4) Evaluation: We will use user feedback to annotate the top ten retrieved results as relevant or irrelevant. Further we will evaluate the performance of our retrieval system in terms of precision. We will also check our system for keyword queries as well as verbose queries.

References

[1] galagosearch. Galago query language. http://code.google.com/p/galagosearch/wiki/GalagoQueryLanguage.

[2] J. Kim, X. Xue, and W. B. Croft. A probabilistic retrieval model for semistructured data. In *Proceedings of the 31th European Conference on IR Research on Advances in Information Retrieval*, ECIR '09, pages 228–239, Berlin, Heidelberg, 2009. Springer-Verlag.

[3] J. Y. Kim. How would you search for your favorite movie? http://lifidea.wordpress.com/2009/08/25/how-would-you-search-for-your-favorite-movie/.

[4] J. M. Ponte and W. B. Croft. A language modeling approach to information retrieval. In *Proceedings of the 21st annual international ACM SIGIR conference on Research and development in information retrieval*, SIGIR '98, pages 275–281, New York, NY, USA, 1998. ACM.

[5] T. L. Project. Indri query language quick reference. http://lemurproject.org/lemur/IndriQueryLanguage.php.

[6] C. Zhai. Statistical language models for information retrieval a critical review. *Found. Trends Inf. Retr.*, 2:137–213, March 2008.