**CS 6200**

**INFORMATION RETRIEVAL**

**Fall 2017**

**Building and Evaluating Search Engine**

**Project Members: Instructor:**

Javesh Monga Nada A. Naji

Sakshi Tonwer

Priyal Mittal

**Table of Contents**

1. Project Details………………………………………………………… 3
2. Introduction…………………………………………………………… 4
3. Literature and Resources……………………………………………… 6
   1. Retrieval Models………………………………………………... 6
   2. Query Refinement………………………………………………. 8
   3. Snippet Generation……………………………………………… 8
   4. References………………………………………………………..
4. Implementation and Discussion………………………………………..
   1. Retrieval Models…………………………………………………
   2. Query Refinement………………………………………………..
   3. Stopping and Stemming………………………………………….
   4. Pseudo Relevance Feedback……………………………………..
   5. Snippet Generation……………………………………………….
   6. Evaluation………………………………………………………...
   7. Query-by-Query Analysis…………………………………………
5. Results…………………………………………………………………...
6. Conclusion and Outlook…………………………………………………
7. Bibliography……………………………………………………………..
8. **Project Details**
9. **Course Details**:

Name: CS6200 Information Retrieval

Semester: Fall 2017

Instructor: Prof. Nada A. Naji

1. **Project Members’ Name and Batch**:

Javesh Monga (Tuesday-Thursday)

Sakshi Tonwer (Tuesday)

Priyal Mittal (Tuesday-Thursday)

1. **Introduction**

The goal of this project is to put the core information retrieval concepts, such as Parsing, Indexing, Retrieval and Evaluation, into practice and build various components of a search engine.

This is implemented by using various retrieval models, performing several optimization techniques on these models and finally assessing their performance (in terms of effectiveness), by calculating various evaluation measures.

Retrieval models implemented are:

* BM25
* Tf-Idf
* Lucene
* Smoothed Query Likelihood

Optimization techniques performed on retrieval models are:

* Query Expansion using Pseudo Relevance
* Stopping
* Stemming

Evaluation measure used to evaluate the retrieval models are:

* MAP
* MRR
* P@K, K= 5 and 20
* Precision and Recall

Documents used:

* CACM corpus which contained 3204 HTML documents.
* CACM queries
* CACM stemmed queries (used during Task3)
* CACM stemmed corpus (used during Task3)
* CACM relevance data

**Contribution of Members:**

*Javesh Monga:*

*Sakshi Tonwer:*

*Priyal Mittal:*

1. **Literature and Resources**

Every query submitted by a user to a search engine has an underlying need of information associated with it. Hence, Information Retrieval is defined as the process of obtaining information resources, from a collection of resources, relevant to the query.

This process involves computing a score on how relevant is an obtained piece of information to the query, and rank them according to this value.

This is performed by using various Retrieval Models.

The retrieval models used in our project are as follows:

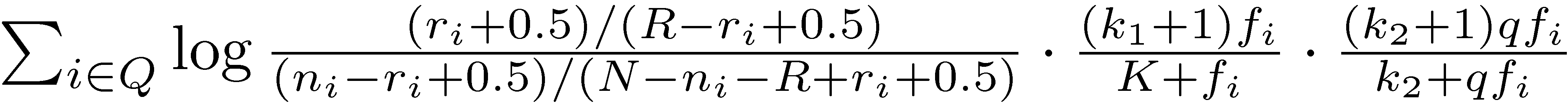
* 1. **RETRIEVAL MODELS:**

**3.1.1) BM25:**

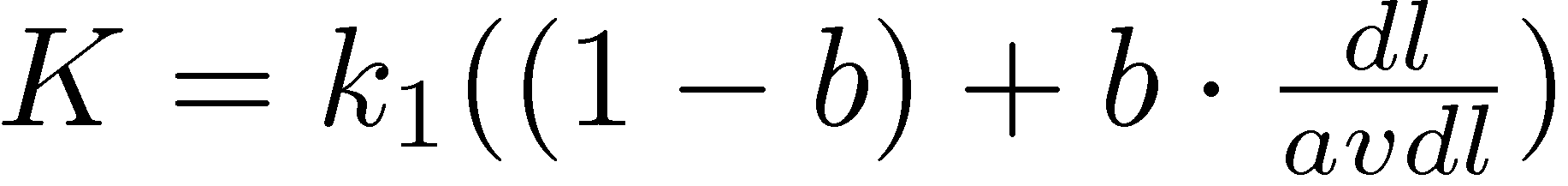
The disadvantage of the Binary Independence retrieval model was that it didn’t take into account the tf component.

BM25 extends the scoring function for the binary independence model to include document and query term weights. The extension is based on probabilistic arguments and experimental validation.

The formula for BM25 is:



where,



* k1, k2, and K are parameters whose values are set empirically
* Typical TREC values for k1 is 1.2, k2 varies from 0 to 1000, b = 0.75
* qfi is the within query frequency
* fi is the within document frequency
* n is the number of documents in the collection indexed by this term
* N is the total number of documents in the collection
* r is the number of relevant documents indexed by this term
* R is the total number of relevant documents
* dl is Document length and avdl is Average Document length
  + 1. **Tf-Idf**:

It stands for term frequency-inverse document frequency. tf.idf weight is often used in information retrieval to evaluate how important a word is to a document in a corpus. The importance of the document increases proportionally with the frequency of occurrence of a word in the document and inverse proportionally with the occurrence of the word in the corpus.

**TF**: Term Frequency, measures how frequently a word occurs in a document in a corpus.

TF(w) = (Number of times word w appears in a document)

(Total number of words in the document)

**IDF**: Inverse Document Frequency, measure how important a word is, as per the corpus.

We weigh down the frequent words and scale up the rare ones, by computing:

IDF(w) = log (Total number of documents in corpus)

(Number of documents with word w)

* + 1. **Lucene**:

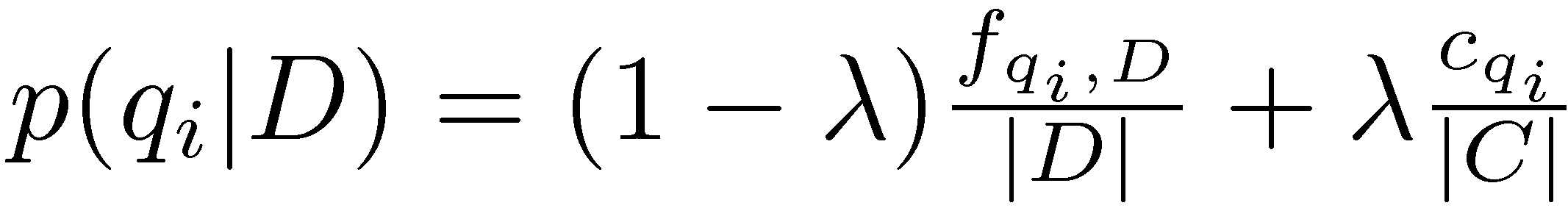
Lucene is an open source and full-featured text search engine library written entirely in JAVA. The components and concepts of Lucene scoring are defined by Similarity.

It combines Vector Space and Boolean models. Since, tf-idf values are believed to produce results of the highest quality, hence, the Similarity in Lucene uses tf-idf weighting.

* + 1. **Smoothed Query Likelihood**:

This is a language model that ranks documents by the probability that the query text could be generated by the document language model. This model is based on topical relevance of documents as per the query.

We use *Smoothing* to avoid the estimation problem of obtaining a zero query likelihood score for any of the query words missing from the document. In our project, we have used the *Jelinek-Mercer Smoothing* technique by using the following formula:



* 1. **QUERY REFINEMENT:**

**3.2.1)** **Stopping and Stemming:**

Stopping is removing frequently occurring words from the tokens of a corpus that become the index terms. These common words are usually called *functional words* as they help form sentence structures but are less descriptive corresponding to a topic.

Stemming, on the other hand, is the process of grouping words that are derived from a common ‘*stem*’.

While performing text transformation, it is important to note that the initial stages of processing a text query should mirror the processing steps that are used for documents. Their execution in our project is mentioned in the next section.

**3.2.2) Pseudo Relevance Feedback:**

For query refinement of the provided queries, *Pseudo Relevance Feedback* technique is used which is mentioned in the book *Search Engines- Information Retrieval in Practi*ce- Croft and the slides provided by Prof. Nada A. Naji.

In this approach, instead of interfering a user for identification of relevant documents, the system simply assumes the top-ranked documents as relevant. For this we choose the top “k” documents as a threshold value and the system reformulates the query by reweighting the words in the relevant documents with a higher weight and decreasing the weight for the words that appear in the non-relevant documents.

* 1. **SNIPPET GENERATION:**

Snippet is a short text summary, that is used to convey the content of the page, wherein, most of the query words are highlighted (displayed in a bold font) to make them easier to identify.

Snippets generated are query-dependent summaries, since the snippet that is generated for a page depends on the query that retrieved it.

In our project, we have used the simple summarization approach, mentioned in the slides provided by Prof. Nada A. Naji, as part of the curriculum.

Applying this approach, we divided each original document into fragments (summary), based on the locations of matched terms in the document. We then assigned a score to each fragment and ranked each of these based on the score and finally presented the top-scoring fragments (in human-readable form) to the user.

1. **Implementation and Discussion**

**Task 3.** **Stopping and Stemming:**

For stopping, we used the words present in the “common\_words.txt” file and tokens generated from the corpus and compared them. If we find a word of the file in the corpus, it is removed else is kept as it is. This process is repeated for the tokens in all the corpus documents and the query, in order to avoid errors in ranking.

For stemming, we used the stemmed corpus file “cacm\_stem.txt” as input to the aforementioned retrieval models and generated top ranked documents, individually, for stemmed queries present in the file “cacm\_queries.txt”.