Information Retrieval and Web Search

Lecture 01

Lecture by Dr. Inbal Budowski-Tal
Typeset by Steven Karas

2018-03-14
Last edited 20:02:25 2018-03-14

Disclaimer These lecture notes are based on the lecture for the course Information Retrieval and Web Search, taught by Dr. Inbal Budowski-Tal at IDC Herzliyah in the spring semester of 2017/2018. Sections may be based on the lecture slides written by Dr. Inbal Budowski-Tal.

1 Course Admin

Office hours will be held immediately following the lecture.

The course book is available online. This course will be largely based upon the Stanford course.

There will be 4 homework assignments, 2 theoretical and 2 practical[1]. The theoretical assignments must be done individually, but the practical programming assignments may be done in pairs. The theoretical assignments will make up 20% of the grade, the programming assignments will make up 30% of the grade, and the final exam will be worth 50% of the grade.

2 Introduction

Information retrieval is finding material of an unstructured nature that satisfies an information need from within large collections.

The information need is context dependent. For example, given a query of pizza, sometimes we are looking for recipes, other times we want to order, and sometimes we're looking for the history of pizza.

Structured Data Structured data is data that has structure, typically tabular in nature. In this course, we will discuss unstructured data, which makes up the majority of all data stored in the world.

3 Boolean Retrieval

In the boolean retrieval problem, we construct a query as a logical expression of search terms and operators such as AND, OR, and NOT. The search engine should provide all results that satisfy the expression.

Many practical systems in use today provide boolean retrieval. For example, Westlaw provides a searchable database of legal documents.

An example of a query to Westlaw is `"trade secret" /s disclos! /s prevent /s employe!`. The operator `/s` is a positional conjunction where the terms should appear in the same sentence. Similar operators are provided for constraining the locations to within words, paragraphs, etc.

Such queries are typically incrementally developed, and preferred for their precision, transparency, and control.

3.1 Incidence Matrix

An incidence vector is a vector of boolean true/false values (typically represented as 1/0) that represents the appearance of each term in a document. An incidence matrix is a matrix of all

[1] The theoretical homeworks will be easy, but the programming assignments will be focused on building simple search engines using Lucene in Java.

such incidence vectors for a set of documents. Typically the set of terms is taken as the set of all distinct terms in the entirety of all the documents[2].

This data structure is extremely inefficient for larger corpuses, and the resulting matrix will generally be extremely sparse.

3.2 Inverted Index

This data structure stores the documents a term appears in as a linked list, so as to avoid the wasted space of an extremely sparse incidence matrix.

3.3 Preprocessing

When building an inverted index, we first need to tokenize the document into a list of terms, and some preprocessing is applied, for example to normalize the terms into the same form (e.g. "Hello, World!" would become `hello`, `world`). We would also mark the appearance frequency and track the original location of the term.

Later in the course, we'll cover efficient construction, index compression, and ranked retrieval.

3.4 Processing Queries

Constructing the intersection of two sorted linked lists is an operation that can be done in $O(n)$. Constructing the union of two sorted linked lists can also be done in $O(n)$. Constructing the negation of a sorted list can be done in $O(u)$, but there is a more efficient way of doing this, which we will need to discover on our own for the first assignment.

3.5 Query Optimization

When searching for multiple terms, we can process in increasing order of list size. We can estimate the size of AND clauses as the pairwise maximum, and the size of OR clauses to be the sum.

[2] As part of presenting this, we discussed Shakespeare's Plays as an example corpus.