The African Virtual University (AVU) is proud to participate in increasing access to education in African countries through the production of quality learning materials. We are also proud to contribute to global knowledge as our Open Educational Resources are mostly accessed from outside the African continent.

This module was developed as part of a diploma and degree program in Applied Computer Science, in collaboration with 18 African partner institutions from 16 countries. A total of 156 modules were developed or translated to ensure availability in English, French and Portuguese. These modules have also been made available as open education resources (OER) on oer.avu.org.

On behalf of the African Virtual University and our patron, our partner institutions, the African Development Bank, I invite you to use this module in your institution, for your own education, to share it as widely as possible and to participate actively in the AVU communities of practice of your interest. We are committed to be on the frontline of developing and sharing Open Educational Resources.

The African Virtual University (AVU) is a Pan African Intergovernmental Organization established by charter with the mandate of significantly increasing access to quality higher education and training through the innovative use of information communication technologies. A Charter, establishing the AVU as an Intergovernmental Organization, has been signed so far by nineteen (19) African Governments - Kenya, Senegal, Mauritania, Mali, Cote d’Ivoire, Tanzania, Mozambique, Democratic Republic of Congo, Benin, Ghana, Republic of Guinea, Burkina Faso, Niger, South Sudan, Sudan, The Gambia, Guinea-Bissau, Ethiopia and Cape Verde.

The following institutions participated in the Applied Computer Science Program: (1) Université d’Abomey Calavi in Benin; (2) Université de Ouagadougou in Burkina Faso; (3) Université Lumière de Bujumbura in Burundi; (4) Université de Douala in Cameroon; (5) Université de Nouakchott in Mauritania; (6) Université Gaston Berger in Senegal; (7) Université des Sciences, des Techniques et Technologies de Bamako in Mali (8) Ghana Institute of Management and Public Administration; (9) Kwame Nkrumah University of Science and Technology in Ghana; (10) Kenyatta University in Kenya; (11) Egerton University in Kenya; (12) Addis Ababa University in Ethiopia (13) University of Rwanda; (14) University of Dar es Salaam in Tanzania; (15) Université Abdou Moumouni de Niamey in Niger; (16) Université Cheikh Anta Diop in Senegal; (17) Universidade Pedagógica in Mozambique; and (18) The University of the Gambia in The Gambia.

Bakary Diallo
The Rector
African Virtual University
Production Credits

Author
Fekade Getahun

Peer Reviewer
Robert Oboko

AVU - Academic Coordination
Dr. Marilena Cabral

Overall Coordinator Applied Computer Science Program
Prof Tim Mwololo Waema

Module Coordinator
Robert Oboko

Instructional Designers
Elizabeth Mbasu
Benta Ochola
Diana Tuel

Media Team
Sidney McGregor       Michal Abigail Koyier
Barry Savala          Mercy Tabi Ojwang
Edwin Kiprono         Josiah Mutsogu
Kelvin Muriithi       Kefa Murimi
Victor Oluoch Otieno  Gerisson Mulongo
Copyright Notice

This document is published under the conditions of the Creative Commons Attribution http://creativecommons.org/licenses/by/2.5/

Module Template is copyright African Virtual University licensed under a Creative Commons Attribution-ShareAlike 4.0 International License. CC-BY, SA

Supported By

AVU Multinational Project II funded by the African Development Bank.
# Table of Contents

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Foreword</td>
<td>2</td>
</tr>
<tr>
<td>Production Credits</td>
<td>3</td>
</tr>
<tr>
<td>Copyright Notice</td>
<td>4</td>
</tr>
<tr>
<td>Supported By</td>
<td>4</td>
</tr>
<tr>
<td>Course Overview</td>
<td>8</td>
</tr>
<tr>
<td>Welcome to Information Retrieval</td>
<td>8</td>
</tr>
<tr>
<td>Prerequisites</td>
<td>8</td>
</tr>
<tr>
<td>Materials</td>
<td>8</td>
</tr>
<tr>
<td>Course Goals</td>
<td>8</td>
</tr>
<tr>
<td>Units</td>
<td>8</td>
</tr>
<tr>
<td>Assessment</td>
<td>9</td>
</tr>
<tr>
<td>Schedule</td>
<td>10</td>
</tr>
<tr>
<td>Readings and Other Resources</td>
<td>12</td>
</tr>
<tr>
<td><strong>Unit 0: Pre-Assessment</strong></td>
<td>14</td>
</tr>
<tr>
<td>Introduction</td>
<td>14</td>
</tr>
<tr>
<td>Objectives</td>
<td>14</td>
</tr>
<tr>
<td>Learning Activities</td>
<td>14</td>
</tr>
<tr>
<td>Key Terms</td>
<td>14</td>
</tr>
<tr>
<td>Assessment</td>
<td>17</td>
</tr>
<tr>
<td>Summary</td>
<td>17</td>
</tr>
<tr>
<td><strong>Unit 1: Introduction to Information Retrieval</strong></td>
<td>18</td>
</tr>
<tr>
<td>Introduction</td>
<td>18</td>
</tr>
<tr>
<td>Objectives</td>
<td>19</td>
</tr>
<tr>
<td>Learning Activities</td>
<td>20</td>
</tr>
<tr>
<td>1. Adhoc Information Retrieval</td>
<td>20</td>
</tr>
<tr>
<td>Introduction</td>
<td>20</td>
</tr>
<tr>
<td>Key Terms</td>
<td>20</td>
</tr>
<tr>
<td>2. Text Representation</td>
<td>21</td>
</tr>
<tr>
<td>Unit 1: Information Retrieval</td>
<td>21</td>
</tr>
<tr>
<td>-------------------------------</td>
<td>----</td>
</tr>
<tr>
<td>Introduction</td>
<td>21</td>
</tr>
<tr>
<td>The “Bag of Words” Representation</td>
<td>21</td>
</tr>
<tr>
<td>Term Weighting</td>
<td>22</td>
</tr>
<tr>
<td>3. Boolean Model</td>
<td>23</td>
</tr>
<tr>
<td>Introduction</td>
<td>23</td>
</tr>
<tr>
<td>Summary</td>
<td>23</td>
</tr>
<tr>
<td>Assessment</td>
<td>23</td>
</tr>
<tr>
<td>Activity Details</td>
<td>24</td>
</tr>
<tr>
<td>4. Vector Space Model</td>
<td>25</td>
</tr>
<tr>
<td>Introduction</td>
<td>25</td>
</tr>
<tr>
<td>Dot products</td>
<td>25</td>
</tr>
<tr>
<td>Summary</td>
<td>25</td>
</tr>
<tr>
<td>Assessment</td>
<td>25</td>
</tr>
<tr>
<td>Queries as vectors</td>
<td>26</td>
</tr>
<tr>
<td>Summary</td>
<td>28</td>
</tr>
<tr>
<td>Assessment</td>
<td>29</td>
</tr>
<tr>
<td>Unit Summary</td>
<td>29</td>
</tr>
</tbody>
</table>

**Unit 2: Indexing and Query Processing**

<table>
<thead>
<tr>
<th>Introduction</th>
<th>30</th>
</tr>
</thead>
<tbody>
<tr>
<td>Objectives</td>
<td>30</td>
</tr>
<tr>
<td>Key Terms.</td>
<td>30</td>
</tr>
<tr>
<td>Learning Activities</td>
<td>31</td>
</tr>
<tr>
<td>Inverted Index/Files</td>
<td>31</td>
</tr>
<tr>
<td>Introduction</td>
<td>31</td>
</tr>
<tr>
<td>Activity Details</td>
<td>32</td>
</tr>
<tr>
<td>Text Query Processing</td>
<td>39</td>
</tr>
<tr>
<td>Introduction</td>
<td>39</td>
</tr>
<tr>
<td>Activity Details</td>
<td>39</td>
</tr>
<tr>
<td>Assessment</td>
<td>41</td>
</tr>
</tbody>
</table>

**Unit 3: Document Clustering**

| 42 |
Course Overview

Welcome to Information Retrieval

This course focuses on basic and advanced techniques on text-based information retrieval (IR), the study of the processing, indexing, querying, organizing, and classifying textual documents, including hypertext documents available on the world-wide-web.

Prerequisites

Principles of Database, Database Systems

Materials

The materials required to complete this course are:


Course Goals

Upon completion of this course the learner should be able to

- To describe basic notions and concepts of information retrieval
- Distinguish information extraction, text representation, indexing, and text-based analysis techniques
- To identify and describe the inner workings of search engines as well as text search algorithms
- To examine and use algorithms relevant to IR
- To use the knowledge and skills of text-based retrieval techniques for local content

Units

Unit 0: Pre-assessment

Revises database modelling and database design.
Unit 1: Introduction to Information Retrieval

It introduces the concept of ad hoc information retrieval, Text Representation models, Bag of words model, Boolean model, Vector Space Model, compare and contrast the difference between information retrieval and database retrieval.

Unit 2: Indexing and Query Processing

It focuses on introducing the concept of index construction and Text Query processing.

Unit 3: Document clustering

It introduces the concept of clustering documents into groups based on the inherent similarity between its content. Specific clustering algorithms, partitioning and hierarchical, will be discussed.

Unit 4: Web Search Engine

It introduces the different components of web search engine, a taxonomy of search engine, crawling, metasearchers and web directors are discussed.

Unit 5: Evaluation of Information Retrieval system

In this unit, the known methods in measuring efficiency of IR system will be presented. In particular, Recall, precision and F-score will be presented.

Assessment

Formative assessments, used to check learner progress, are included in each unit.

Summative assessments, such as final tests and assignments, are provided at the end of each module and cover knowledge and skills from the entire module.

Summative assessments are administered at the discretion of the institution offering the course. The suggested assessment plan is as follows:
### Continuous assessment

<table>
<thead>
<tr>
<th>Continuous assessment</th>
<th>Weight (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.1 Test I - Information retrieval + Indexing and Query Processing</td>
<td>10</td>
</tr>
<tr>
<td>1.2 Test II - Document Clustering</td>
<td>10</td>
</tr>
<tr>
<td>1.3 Test III - Web Search Engine + Evaluation in Information Retrieval</td>
<td>10</td>
</tr>
<tr>
<td>1.4 Home take assignment</td>
<td>10</td>
</tr>
<tr>
<td>1.5 Laboratory Project</td>
<td>20</td>
</tr>
<tr>
<td><strong>Total</strong></td>
<td><strong>100</strong></td>
</tr>
</tbody>
</table>

### Schedule

<table>
<thead>
<tr>
<th>Unit</th>
<th>Activities</th>
<th>Estimated time in hours</th>
</tr>
</thead>
<tbody>
<tr>
<td>Unit 0: Pre-assessment</td>
<td>Database modelling</td>
<td>4</td>
</tr>
<tr>
<td></td>
<td>Database design</td>
<td>5</td>
</tr>
<tr>
<td></td>
<td>assessment</td>
<td>3</td>
</tr>
<tr>
<td>Unit 1: Introduction to Information Retrieval</td>
<td>Adhoc information retrieval</td>
<td>3</td>
</tr>
<tr>
<td></td>
<td>Text representation method</td>
<td>4</td>
</tr>
<tr>
<td></td>
<td>Boolean model</td>
<td>5</td>
</tr>
<tr>
<td></td>
<td>Vector space model</td>
<td>8</td>
</tr>
<tr>
<td></td>
<td>Assessment</td>
<td>3</td>
</tr>
<tr>
<td>Unit 2: Indexing and Query Processing</td>
<td>Construct inverted index</td>
<td>8</td>
</tr>
<tr>
<td>-------------------------------------</td>
<td>-------------------------</td>
<td>---</td>
</tr>
<tr>
<td></td>
<td>Text query processing</td>
<td>11</td>
</tr>
<tr>
<td></td>
<td>Assessment</td>
<td>3</td>
</tr>
<tr>
<td>Unit 3: Document Clustering</td>
<td>Text similarity</td>
<td>7</td>
</tr>
<tr>
<td></td>
<td>Cluster text document</td>
<td>16</td>
</tr>
<tr>
<td></td>
<td>using hierarchical approach</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Using k-means to cluster documents</td>
<td>8</td>
</tr>
<tr>
<td></td>
<td>Assessment</td>
<td>3</td>
</tr>
<tr>
<td>Unit 4: Web Search Engine</td>
<td>Taxonomy of web searching</td>
<td>3</td>
</tr>
<tr>
<td></td>
<td>Search engines</td>
<td>8</td>
</tr>
<tr>
<td></td>
<td>Web directories</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td>Assessment</td>
<td>3</td>
</tr>
<tr>
<td>Unit 5: evaluation in IR</td>
<td>Introduction to</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>performance measurement</td>
<td></td>
</tr>
<tr>
<td></td>
<td>in IR</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Recall &amp; precision</td>
<td>5</td>
</tr>
<tr>
<td></td>
<td>F-score</td>
<td>4</td>
</tr>
<tr>
<td></td>
<td>Assessment</td>
<td>3</td>
</tr>
<tr>
<td><strong>Total hours</strong></td>
<td></td>
<td>120</td>
</tr>
</tbody>
</table>
Readings and Other Resources

The readings and other resources in this course are:

Unit 0

Required readings and other resources:


Unit 1

Required readings and other resources:


Unit 2

Required readings and other resources:


Unit 3

Required readings and other resources:


Unit 4

Required readings and other resources:

Unit 5

Required readings and other resources:


Optional readings

- Unit 0 Pre-Assessment
Unit 0: Pre-Assessment

Introduction
The purpose of this unit is to determine your grasp of knowledge related to the pre-requisite course principle of database.

Objectives
Upon completion of this unit you should be able to:

- Understand database modelling
- State the different steps in database design

Key Terms

**Database**: Database is organized collection of data

**Database Model**: A database model determines the logical structure of a database, data stored, organized and manipulated. Relational model is one of the popular database models

**Database Design**: It is a process for creating a detailed database model of a database.

**SQL**: Structured Query Language is a database query-language designed for managing data held in a relational database management system (RDBMS).

Learning Activities

1. **Database Modelling**

   In this activity, review of database modeling restricted to relational database will be discussed. A data model is a notation for describing data or information. The description generally consist of three parts:

   - **Structure of the data**: it refers to the physical data model of the database. It contains the structure of the database and its constituting elements
• **Operations on the data**: In programming languages, operations on the data are generally anything that can be programmed. In database data models, there is usually a limited set of operations that can be performed. The database operations are restricted to querying and modifications for data.

• **Constraints on the data**: A database model describe limitations on what the data can be. These constrains can range from the simple (e.g., “a day of the week is an integer between 1 and 7” or “a movie has at most one title”) to some very complex limitations.

The relational model is based on tables. Each table has columns/attributes and tuples. Figure 0-1 shows an example of relational table representing movie having ID, Title, description, Year and Rating as attributes.

Attributes/ columns of a relation appears at the tops of the columns and describes the meaning of entries. The attributes in a relation schema are a set, not a list and means attributes are not ordered. Each attribute takes a value from its corresponding domain.

The rows of a relation, other than the header row containing the attribute names, are called tuples. A tuple has one component for each attribute of the relation.

<table>
<thead>
<tr>
<th>Film_ID</th>
<th>Title</th>
<th>Description</th>
<th>Year</th>
<th>Rating</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>ADAPTATION HOLES</td>
<td>A Astounding Reflection of a L</td>
<td>2006</td>
<td>NC-17</td>
</tr>
<tr>
<td>448</td>
<td>IDAHO LOVE</td>
<td>A Fast-Paced Drama of a Student</td>
<td>2006</td>
<td>PG-13</td>
</tr>
</tbody>
</table>

*Figure 0-1: Movie table*

The attribute Film_ID is key attribute that can uniquely identify each movie, such attribute key is called primary key. A primary key is the smallest collection of attributes that can identify a tuple.

In relational model, a database is collection of related relations. Conceptually such database is represented in E-R model.

In E-R diagram, real world objects/entities are represented in the form of a rectangle having name and attribute definitions. The line connecting entities denote relationship having carnality attached to both ends as label.

**Error! Reference source not found.** shows the sample movie database containing film, actor, and category tables. The relationship between tables have cardinality which is 1-1 or 1-n (many). However, relational model can’t represent n-m relationship rather it represent as combination of two 1-n relationships. For instance, a movie has one or more actor and an actor may work on several movies thus the relationship between movie and actor is n-m and it is represent as a combination of two 1-n relationships having film_actor used as relation table.
2. **Database Design**

Once the database is modelled and presented using E-R diagram the next step is to map it to relational dataset notions (collection of related tables). Before creating the database, each table has to go through normalization. Normalization is step by step activity that transform table to have less redundancy.

Unnormalized relation causes anomalies: insertion, deletion and update anomalies which are caused due to the existence of redundant group of information. It is advisable to apply 1NF, 2NF, 3NF and BCNF.

The main steps taken to design database are:

1. Understand data requirement
2. Draw conceptual dataset design use E-R diagram to present the result
3. Map conceptual dataset concepts into relational dataset concept

   - Entity becomes table. Attributes of the entity become attributes of the table.
   - If the relationship is 1-1, combine the two tables into one
   - If the relationship is 1-n, in the many side add a foreign key that take the value of the primary key of the relation in one side.
   - If the relationship is n-n, create a separate table that contain the primary key of both relations.
Assessment

Consider the textual description associated different entities, researchers, academic institutions, collaborators in academic institution.

A researcher can either be employed as a professor or a lab assistant. There are three kinds of professors: Assistant, associate, and full professors. The following should be stored:

1. A researcher has name, year of birth, and current position, highest degree (BSc, MSc or PhD), main supervisor
2. A researcher works for a specific school and authors or co-author a research paper. A research paper has title, publishing date, author and co-author
3. Academic institution has name, country, and inauguration year.
4. A school has a name (such as School of Law, School of Business, School of Computer Science, . . .), a school belongs to a given exactly one institution
5. Employee An employment history, including information on all employments (start and end date, position, and what school).
6. Information about co-authorships, i.e., which researchers have co-authored a research paper. The titles of common research papers should also be stored.
7. A professor works on research project each has title, start date, and end date, total amount of grant money.

Summary

A database is a repository that store data. This repository needs to be modelled in nice way to represent real world entities and relationship in between. In this section we have used E-R modelling concept represent a database conceptually.

Designing a database follows rigorous steps called normalization. These are taken to minimize the existence of redundancy and the steps mainly results in decomposition of a relation into sub-relations.
Unit 1: Introduction to Information Retrieval

Introduction

Information retrieval (IR) deals with the representation, storage, organization of, and access to information items. The representation and organization of the information items should provide the user with easy access to the information in which he/she is interested. However, the user information need is not a simple problem. Considering for instance the popular plays of Shakespeare, a user might be interested to look for those plays played by Brutus and Caesar but not by Calpurnia. In most common cases user queries are translated to yield set of keywords which summarizes the description of the user need. The task of IR system is to retrieve information which might be useful or relevant to the user. The emphasis is on the retrieval of information as opposed to the retrieval of data.

In its most common form, this translation yields a set of keywords, or index terms, which summarize the user information need. Given the user query, the key goal of the IR system is to retrieve information that is useful or relevant to the user. The emphasis is on the retrieval of information as opposed to the retrieval of data.

The IR system architecture is composed of the following basic steps as shown in Figure 1.1. The first step in setting up an IR system is to assemble the document collection, which can be private or be crawled from the Web. In the second case a crawler module is responsible for collecting the documents. The document collection is stored in disk storage usually referred to as the central repository. The documents in the central repository need to be indexed for fast retrieval and ranking. The most used index structure is an inverted index composed of all the distinct words of the collection and, for each word, a list of the documents that contain it.

Given that the document collection is indexed, the retrieval process can be initiated. It consists of retrieving documents that satisfy either a user query or a click in the hyperlink. In the first case, the user is searching for information of interest; in the second case, the user is browsing for information of interest.

To search, the user specifies a query that represent their information needs. This query is parsed and expanded with relevant information such as spelling variants of a query word so as to get system query. The system query is processed against the index to retrieve a subset of all documents. Followed by ranking (i.e., ordering the result set based on the relevance of documents) the retrieved result and the top some results are returned to the user.
Unit 1: Introduction to Information Retrieval

Objectives

Upon completion of this unit you should be able to:

- Explain high-level system architecture of IR system
- Explain the concept of adhoc information retrieval
- Analyse the different text representation models
- Apply Boolean Model in text representation
- Compare and contrast the difference between information retrieval and database retrieval.

Figure 1-1: High-level system architecture of IR system
Key Terms

**Information retrieval (IR):** IR is finding material (usually documents) of an unstructured nature (usually text) that satisfies an information need from within large collections (usually stored on computers).

**Vector space model:** Vector space model is an algebraic model for representing text documents (and any objects, in general) as vectors of identifiers, such as, for example, index terms along with its weight.

**Boolean Information retrieval:** Boolean information retrieval is based on Boolean logic and classical set theory in that both the documents to be searched and the user’s query are conceived as sets of terms. Retrieval is based on whether or not the documents contain the query terms.

**Text Similarity:** measures the extent two texts (sentences, paragraph or documents) share common words or concepts. It can be measured using cosine similarity approach.

Learning Activities

1. Adhoc Information Retrieval

**Introduction**

In a conventional information retrieval system, the documents in the collection remain relatively static while new queries are submitted to the system. This operational mode has been termed ad hoc retrieval in recent years and is the most common form of user task. A similar but distinct task is one in which the queries remain relatively static while new documents come into the system (and leave). For instance, this is the case with the stock market and with news wiring services. This operational mode has been termed filtering.

In a filtering task, a user profile describe user’s preferences. Users profile are compared against incoming documents to determine those which might be of interest to this particular user. For instance, this approach can be used to select a news article among thousands of articles which are broadcast each day. Other potential scenarios for the application of filtering include the selection of preferred judicial decisions, or the selection of articles from daily newspapers, etc.

More generally, filtering task is more appropriate in subscription based system, i.e. which indicates to the user the documents which might be of interest to her/him. The quality of the filtering system is highly dependent on the user profile. A user profile can be constructed the following two methods.
A simplistic approach for constructing a user profile is to let the user provide all the necessary keywords that describe her/his preferences. However, if the user is not familiar with the service which generates the upcoming documents, it is difficult to provide the keywords which appropriately describe preferences. In addition, the providing set of keywords, understanding the vocabulary of document is tedious and time consuming and might be impractical. An alternative to this could be to collect information from the user about his preferences and to use this information to build the user profile dynamically, i.e., a user provide initial user provide as set of keywords; when a new document arrives, the system uses this profile to select document of interest and show to the user. The user then goes through a relevance feedback cycle to uses formation contained in the returned document to readjust the user profile description and preference. In both conventional information retrieval and filtering ranking the result is ranked based on the relevance of the result to the user query or user preference.

Users searching for information on the Web may have more complex information needs than simply finding any documents on a certain subject matter. For instance they way want to find documents containing other person’s opinions on a certain topics e.g. product reviews rather than product specification. A user is interested in getting news articles with preference represented with set of keywords: football, Cup of nation, African Union. Define this preferences using keywords and will be represented using text resonation method presented in the next section.

2. Text Representation

Introduction

A text document needs to be represented in systematic manner so that later retrieval can be done easily. In this activity the bag of words model will be discussed. A bag-of-words model is one of the simplified representation model using in natural language processing and information retrieval.

The “Bag of Words” Representation

In bag-of-words model, a text is represented as the bag of words extracted from the text disregarding grammar and word order but keeping its multiplicity or frequency as weight.

For instance given the following two texts:

- T1: Congo: Doctor wins top EU prize
- T2: DR Congo doctor Denis Mukwege wins Sakharov prize

A dictionary containing 10 unique words is constructed i.e. Congo, doctor, wins, top, EU, prize, DR, Denis, Mukwegn, Sakharov

Using the terms in the dictionary, the two documents are represented by a 10 entry vector as shown in Table 1-1.
Table 1-1: Term vector of the two example texts

<table>
<thead>
<tr>
<th>Congo</th>
<th>Doctor</th>
<th>Wins</th>
<th>Top</th>
<th>EU</th>
<th>Prize</th>
<th>DR</th>
<th>Denis</th>
<th>Mukwege</th>
<th>Sakharov</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

The entry in the vector represents the weight of the term, the number of times the term appears in the document. Detail discussion of term weight is provided in the next sub-section.

**Term Weighting**

Term weight determines its importance in the document. The vector above shows term frequencies as weight. The weight of a term can be computed using different principles:

- **Binary value** – i.e. 1/0 representing presence/absence. All words in the document are equally important and the frequency of a term is not important.

- **Term Frequency** – IF: The simplest approach is to assign the weight to be equal to the number of occurrences of term t in document d. This weighting scheme is referred to as term frequency and is denoted $t_f, d$, with the subscripts denoting the term and the document in order.

- **Inverse document frequency** – IDF: Raw term frequency suffers from a critical problem: all terms are considered equally important when it comes to assessing relevancy on a query. Certain terms have little or no discriminating power in determining relevance. For instance, a collection of documents on the auto industry is likely to have the term auto in almost every document. Thus, there is a need for attenuating the effect of terms that occur too often in the collection to be meaningful for relevance determination. i.e. scale down the term weights of terms with high collection frequency (i.e. document frequency), defined to be the total number of occurrences of a term in the collection. The idea would be to reduce the tf weight of a term by a factor that grows with its collection frequency.

$$[\text{idf}]_t = \log \frac{N}{[\text{df}]_t}$$

Where:

- **N** - total number of documents in a collection
- **df** – is the document frequency defined to be the number of documents in the collection that contain a term t.
Table 1-2: Example of IDF values computed using Reuters collection of 806,791 documents

<table>
<thead>
<tr>
<th>Term</th>
<th>dft</th>
<th>idft</th>
</tr>
</thead>
<tbody>
<tr>
<td>Car</td>
<td>18,165</td>
<td>1.65</td>
</tr>
<tr>
<td>Auto</td>
<td>6723</td>
<td>2.08</td>
</tr>
<tr>
<td>Insurance</td>
<td>19241</td>
<td>1.62</td>
</tr>
<tr>
<td>Best</td>
<td>25235</td>
<td>1.5</td>
</tr>
</tbody>
</table>

TF-IDF: assigns to term $t$ a weight $\text{TF-IDF}$ in document $d$ given by

$$\text{tf-idf}_{t,d} = \text{tf}_{t,d} \times \text{idf}_t$$

In other words, $\text{tf-idf}_{t,d}$ assigns to term $t$ a weight in document $d$ that is

1. Highest when $t$ occurs many times within a small number of documents (thus lending high discriminating power to those documents);
2. Lower when the term occurs fewer times in a document, or occurs in many documents (thus offering a less pronounced relevance signal);
3. Lowest when the term occurs in virtually all documents.

Summary

In this section, the popular and widely used text representation i.e. bag of words, is used to represent text. The approach represent a text using set of words also called terms. Each word has weigh computing using frequency or inverse document frequency.

Assessment

1. What is the idf of a term that occurs in every document? Compare this with the use of stop word lists?
2. Can the tf-idf weight of a term in a document exceed 1?

3. Boolean Model

Introduction

The Boolean model is a simple retrieval model based on set theory and Boolean algebra. In this model queries are specified as Boolean expressions which have precise semantics. The model is simple and has neat formalism and was very popular in early commercial bibliographic systems.
However, the Boolean model suffers from major drawbacks.

- First, its retrieval strategy is based on a binary decision criterion (i.e., a document is predicted to be either relevant or non-relevant) without any notion of a grading scale, which prevents good retrieval performance. It is a data retrieval model rather than information retrieval.
- Second, while Boolean expressions have precise semantics, frequently it is not simple to translate an information need into a Boolean expression.

The Boolean model considers that index terms are present or absent in a document. As a result, the index term weights are assumed to be all binary, i.e., \( w_{ij} \in \{0,1\} \).

A query \( q \) is composed of index terms linked by three connectives: not, and, or. Thus, a query is essentially a conventional Boolean expression which can be represented as a disjunction of conjunctive vectors (i.e., in disjunctive normal form - DNF). For instance, the query \( q = k_a \setminus (k_b \lor \ldots \lor k_c) \) can be written in disjunctive normal form as \( q_{dnf} = (1,1,1) \lor (1,1,0) \lor (1,0,0) \), where each of the components is a binary weighted vector associated with the tuple \( (k_a, k_b, k_c) \).

Where \( k_i \) are keywords (or query term).

**Activity Details**

This activity shows how to build a document matrix (vector) and identify the list of documents relevant to a given query.

In Boolean model, for each document identify its keywords (i.e. set of non-stop words/terms).

The document vector will have weight of 1 or 0 to represent presence or absence of term in the document. The next table show 7 documents represented with four keywords.

<table>
<thead>
<tr>
<th>Doc</th>
<th>Nuclear</th>
<th>Nonproliferation</th>
<th>Treaty</th>
<th>Iran</th>
</tr>
</thead>
<tbody>
<tr>
<td>D1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>D2</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>D3</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>D4</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>D5</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>D6</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>D7</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>D8</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>
For instance, a user is interested to look for document about: Nuclear Treaty in Non Iran. This query is represented in Boolean model as Nuclear AND Treaty NOT Iran

This query is denoted as union of the following two vectors (1,0,1,0)V(1,1,1,0). Execute compute the similarity between the query vector and the document vectors. The Boolean model returns only document D7 as relevant as it is described with Nuclear and Treaty but not Iran.

Summary

The Boolean model is used to represent both query and document using Boolean expressions. Every user query is represented as Boolean expression that shows the existence of keywords or not (i.e. conjunctive or disjunctive normal forms).

Assessment

1. What is a Boolean model?
2. What are the main limitation of Boolean model in information retrieval?

4. Vector Space Model

Introduction

Earlier we have seen the notion of a document vector that captures the relative importance of the terms in a document. The representation of a set of documents as vectors in a common vector space is known as the vector space model and is fundamental to a host of information retrieval operations ranging from scoring documents on a query, document classification and document clustering. In this activity, basic ideas underlying vector space scoring are presented.

Dot products

Let V \(^{(d)}\) the vector derived from document d, with one component in the vector for each dictionary term. The component of each term is weight computed using TF, IDF or TF-IDF weighting scheme. The set of documents in a collection then may be viewed as a set of vectors in a vector space, in which there is one axis for each term.

To measure the similarity between two documents in vector space model, we consider the magnitude of the two vectors difference between the vectors. This measure suffers from a drawback: two documents with very similar content can have a significant vector difference simply because one is much longer than the other. Thus the relative distributions of terms may be identical in the two documents, but the absolute term frequencies of one may be far larger.

To compensate for the effect of document length, the standard way of quantifying the similarity between two documents d1 and d2 is to compute the cosine similarity of their vector representations V \(^{(d1)}\) and V \(^{(d2)}\).
\[ \text{sim}(d_1, d_2) = \frac{\langle V(d_1), V(d_2) \rangle}{|V(d_1)||V(d_2)|} \]

where

- The numerator represents the dot product (also known as the inner product) of the vectors \( V(d_1) \) and \( V(d_2) \)
- The denominator is the product of their Euclidean lengths. The dot product of two vectors \( V(d_1) \) and \( V(d_2) \) is defined as \[ \sqrt{\sum_{i=1}^{m} (v(d_1)_i \cdot v(d_2)_i)}. \]

Example: consider term vector presenting bag of words of three documents along with frequency as presented in the table below.

<table>
<thead>
<tr>
<th>Term</th>
<th>D1</th>
<th>D2</th>
<th>D3</th>
</tr>
</thead>
<tbody>
<tr>
<td>car</td>
<td>27</td>
<td>4</td>
<td>24</td>
</tr>
<tr>
<td>auto</td>
<td>3</td>
<td>33</td>
<td>0</td>
</tr>
<tr>
<td>insurance</td>
<td>0</td>
<td>33</td>
<td>29</td>
</tr>
<tr>
<td>best</td>
<td>14</td>
<td>0</td>
<td>17</td>
</tr>
</tbody>
</table>

Table 1-3: Term vectors for three documents

\[ \text{sim}(d_1, d_2) = \frac{\langle V(d_1), V(d_2) \rangle}{|V(d_1)||V(d_2)|} \]

\[ V(d_1) \cdot V(d_2) = (27) \cdot (4) + (3) \cdot (33) + (0) \cdot (33) + (14) \cdot (0) = 207 \]

\[ |V(d_1)| = \sqrt{(27)^2 + (3)^2 + (0)^2 + (14)^2} = 30.5614135799 \]

\[ |V(d_2)| = \sqrt{(4)^2 + (33)^2 + (33)^2 + (0)^2} = 46.8401537145 \]

\[ \text{sim}(d_1, d_2) = \frac{207}{(30.5614135799 \cdot 46.8401537145)} \]

Similarity, the similarity between documents is computed as

\[ \text{sim}(d_1, d_3) = 0.70189 \]

\[ \text{sim}(d_2, d_3) = 0.544277 \]

The three similarity values shows that document \( d_1 \) is more similar to \( d_3 \) and \( d_2 \) and \( d_3 \) are similar as well.

**Queries as vectors**

Similar to documents, a user query, \( q \), is represented as vectors also called query vector, \( V(q) \).

Each dimension of \( V(q) \) corresponds to a separate term. If a term occurs in the query its value in the vector is non-zero, for instance 1, otherwise 0.
Cosine similarity can be used to compute the similarity between each document \( d_i \) of the document collection and the query. Documents having higher similarity value are considered as relevant result for the query.

\[
sim(d_i, q) = \frac{(V(d_i) \cdot V(q))}{|V(d_i)||V(q)|}
\]

Example: considering the document vectors presented in Table 1-3 and the user query search for the best car insurance.

Error! Reference source not found. shows the query and term vectors. It also shows the TF-IDF value of terms in corresponding document.

<table>
<thead>
<tr>
<th>Term</th>
<th>Query</th>
<th>Document Frequency</th>
<th>TF-IDF Document</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Q</td>
<td>D1</td>
<td>D2</td>
</tr>
<tr>
<td>Car</td>
<td>1</td>
<td>27</td>
<td>4</td>
</tr>
<tr>
<td>Auto</td>
<td>0</td>
<td>3</td>
<td>33</td>
</tr>
<tr>
<td>Insurance</td>
<td>1</td>
<td>0</td>
<td>33</td>
</tr>
<tr>
<td>Best</td>
<td>1</td>
<td>14</td>
<td>0</td>
</tr>
</tbody>
</table>

Table 1-4: Query and Term vectors for three documents

<table>
<thead>
<tr>
<th>9</th>
<th>10</th>
<th>11</th>
<th>12</th>
<th>13</th>
<th>14</th>
<th>15</th>
</tr>
</thead>
<tbody>
<tr>
<td>10-Apr</td>
<td>Arrives</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>11-Apr</td>
<td>DCS</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>12-Apr</td>
<td>DIS</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>13-Apr</td>
<td>State Minister, ME, Zelalem Assefa</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>14-Apr</td>
<td>Bahir-Dar University</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

The dot product of the query vector is computed and presented in Table 1-6.

<table>
<thead>
<tr>
<th>Q.D1</th>
<th>41</th>
</tr>
</thead>
<tbody>
<tr>
<td>Q.D2</td>
<td>37</td>
</tr>
<tr>
<td>Q.D3</td>
<td>70</td>
</tr>
</tbody>
</table>

Table 1-5: Dot product of Q and each document
The magnitude of each vector is computed and presented in Table 1-6.

<table>
<thead>
<tr>
<th></th>
<th>9</th>
<th>10</th>
<th>11</th>
<th>12</th>
<th>13</th>
<th>14</th>
<th>15</th>
</tr>
</thead>
<tbody>
<tr>
<td>10-Apr</td>
<td>Arrives</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>11-Apr</td>
<td>DCS</td>
<td>DIS</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>12-Apr</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>13-Apr</td>
<td>State Minister, ME, Zelalem Assefa</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>14-Apr</td>
<td>Bahir-Dar University</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

The cosine similarity between the query and each document is computed as

$$\text{Sim}(Q, D) = \frac{(Q \cdot D)}{|Q||D|}$$

- $$\text{Sim}(Q, D_1) = \frac{41}{(1.73 \cdot 30.6)} = 0.77$$
- $$\text{Sim}(Q, D_2) = \frac{37}{(1.73 \cdot 46.8)} = 0.46$$
- $$\text{Sim}(Q, D_3) = \frac{70}{(1.73 \cdot 41.2)} = 0.98$$

Thus the query Q is very similar to D3 followed by D1 and less similar to D2.

**Summary**

Vector space model is used to represent text documents in multidimensional space in which keywords extracted from documents are used as dimension. The vector space is used to represent query and retrieve list of documents very similar to the query using cosine similarity method.
Assessment

1. Given the query “digital cameras” and the document “digital cameras and video cameras”.

2. What is the vector similarity score between the query and the document if the term weighting is using Boolean weighting scheme and disregarding the use of common words stopwords?

3. What is the vector similarity score between the query and the document if the term weighting is term frequency and considering all words?

4. Given two documents “The dog eat the boy” and “the boy eat the dog”, compute their corresponding cosine similarity.

5. What are the main drawback of vector based similarity measure, discuss with example?

Unit Summary

In this unit basic notion of Information retrieval is presented. In particular, the two popular text representation models – Boolean and vector space. The Boolean model present texts and query using Boolean or logical expression whereas vector space model represents documents/query using its key terms in multi-dimensional space. The weight of each dimension is computed using binary value, term frequency or term frequency and inverse document frequency. The notion of vector space model is used to compute the similarity between documents or between documents and query using cosine similarity approach.
Unit 2: Indexing and Query Processing

Introduction

This unit describes indexing techniques in particular inverted files. Indexing facilitates searching of text documents using keywords. In this using keyword-based searching in relation to inverted index is discussed. In addition, the query operations and main techniques needed to perform information retrieval on text databases will be disused.

Objectives

Upon completion of this unit you should be able to:

- Explain indexing process.
- Construct inverted index.
- Evaluate the different text query processing approaches.

Key Terms

**Inverted index** Inverted index is a word-oriented mechanism for indexing a text collection to speed up the searching task.

**Vocabulary** It is the set of different words in the text.

**Tokenization** It is the process of chopping down a document into pieces or tokens.

**Stop word** Stop words are words which are filtered out before or after processing of natural language data. These words occur almost in all documents and will not be capable to differentiate a document.
Inverted Index/Files

Introduction

Inverted index is the basic data structure in the design of data retrieval system such as search engines. It allows efficient retrieval. Each document is uniquely identified with unique id – docID. As a document is collection of words, you may think of using fixed size array to model a document. However, this is not ideal as the addition of a new word will change dimension of posting.

An inverted file (or inverted index) is a word-oriented mechanism for indexing a text collection in order to speed up the searching task. The inverted file structure is composed of two elements: the vocabulary and the occurrences. The vocabulary is the set of all different words in the text. For each such word a list of all the text positions where the word appears is stored. The set of all those lists is called the ‘occurrences’ and stored in the posting lists. These positions can refers to words or characters. The word positons are relevant in answering phrase and proximity queries.

<table>
<thead>
<tr>
<th>1</th>
<th>6</th>
<th>9</th>
<th>1</th>
<th>17</th>
<th>19</th>
<th>14</th>
<th>28</th>
<th>33</th>
<th>40</th>
<th>46</th>
<th>50</th>
<th>55</th>
<th>60</th>
</tr>
</thead>
<tbody>
<tr>
<td>This is a text. A text has many Words. Words are made from letters.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Text

<table>
<thead>
<tr>
<th>Vocabulary</th>
<th>Occurrences</th>
</tr>
</thead>
<tbody>
<tr>
<td>letters</td>
<td>60...</td>
</tr>
<tr>
<td>made</td>
<td>50...</td>
</tr>
<tr>
<td>many</td>
<td>28...</td>
</tr>
<tr>
<td>text</td>
<td>11, 19, ...</td>
</tr>
<tr>
<td>words</td>
<td>33, 40, ...</td>
</tr>
</tbody>
</table>

*Figure 2-1: A sample text and its corresponding Inverted index lists*

The space required for the vocabulary is relatively small. The size will be further reduced if stop words are removed and stemming is applied to normalize words. However, the occurrence demands much more space. As each word appearing in the text is referenced once in that structure, the extra space is O(n). To reduce space requirements, block addressing is used.
The text is divided into blocks, and the occurrences points to the blocks where the word appears. Using block addressing, the pointers are smaller because there are fewer blocks than positions and also all the occurrences of a word inside a single block are collapsed to one reference.

<table>
<thead>
<tr>
<th>Block 1</th>
<th>Block 2</th>
<th>Block 3</th>
<th>Block 4</th>
</tr>
</thead>
<tbody>
<tr>
<td>This is a text.</td>
<td>A text</td>
<td>Words. Words are made from letters.</td>
<td></td>
</tr>
<tr>
<td>has many</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Vocabulary/Dictionary** | **Occurrences/Postings**
---|---
letters | 4...
made | 3...
many | 2...
text | 1, 2, ...
words | 3, ...

*Figure 2-2: The sample text split into four blocks and its corresponding Inverted index lists*

Figure 2-2 shows inverted index with block based occurrence. Such index file reduces size but it is inefficient for a proximity query.

**Activity Details**

Building and maintaining an inverted index on a text of n characters need O(n) time. Omitting stop-words reduce the space overhead of the occurrence.

**a. Building an Index**

The following are the major steps in building index.

1. Collect the documents to be indexed
2. Tokenize the text, turning each document into a list of tokens so as to build dictionary and postings
3. Do linguistic preprocessing, producing a list of normalized tokens also called indexing terms
4. Index the documents that each term occurs in by creating an inverted index, consisting of a dictionary and postings

5. Within a document collection, we assume that each document has a unique serial number, known as the document identifier (docID).

**b. Tokenization**

Tokenization is the task of chopping a document up into pieces, called tokens, perhaps at the same time throwing away certain characters, such as punctuation. Here is an example of tokenization:

Input: Friends, Romans, Countrymen, lend me your ears;

Output: Friends Romans Countrymen lend me your ears

A token is a sequence of characters also called term representing a semantic unit from a document. A type is the class of all tokens containing the same character sequence. A term is included in the IR system’s dictionary. The Dictionary contains distinct set of index terms. These index terms are strongly related to the tokens in the document.

One of the major question in the tokenization phase is identifying the correct tokens? In some cases chopping text using whitespace and throw away punctuation characters may not be enough. The existence of apostrophe for possession and contractions is a case that need attention.

For o’Neill, which of the following is the desired token?

- neill
- oneill
- o’neill
- o neill
- o neill.

And for aren’t, which of the following is the desired token?:

- aren’t
- arent
- are
- n’t
- aren t
A simple strategy is to split on all non-alphanumeric characters, but o neill seems okay, aren't looks unacceptable. The Boolean query neill match with three tokens. However, using a query o’neill matches only one case unless preprocessing to the query is done. Notice that the use of whitespace character or hyphen can also split what should be regarded as a single token. For instance San Francisco, Addis Ababa, Los Angeles, plug-in, Hewlett-Packard) each represent a token. Splitting using whitespace splits what should be considered as one token into more tokens (i.e., San Francisco becomes San Francisco).

In some cases there is a need to consider complex concepts or technical concepts, organization and product names which are multiword or phrases. In such cases the tokenizer shouldn’t chop the text looking for whitespace character rather use the phrase as is. Recently web search engines are supporting phrase queries either using double quotes as delimiter or without the quotes.

To handle the hyphen case, hyphen shouldn’t be used as whitespace character and encourage users to formulate queries using hyphen.

**Linguistic processing**

The tokens generated from the tokenization process may exist in different forms and some of the tokens may exist most frequently hence there is a need to normalize the tokens or exclude it from inverted index construction.

**Dropping common terms: stop words**

Stop words are extremely common words that appear in almost all documents. Such words are considered to be of little value in selecting documents matching and are excluded from the vocabulary entirely (i.e., searching a document using the and by doesn’t seems very useful).

The common strategy for determining a stop list is to sort the terms by collection frequency and then to take the most frequent terms as stop list. The use of a stop-list reduces the number of postings that a system has to store.

Notice that in phrase based matching, excluding stop word has impact in the query result. For instance in the phrase query “President of the United States” which contains two stop-words of is more precise than President and “United States”.

**Normalization**

Map text and query term to the same form. It is likely that a word may appear in different forms. For instance, U.S.A and USA are referring to the same thing but represented differently. The normalization is done to make words to be similar.
Stemming and lemmatization

It is likely that a word may have different forms of a root to match. For instance, for grammatical reasons, documents are going to use different forms of a word, such as organize, organizes, and organizing. Additionally, there are families of derivationally related words with similar meanings, such as democracy, democratic, and democratization. In many situations, it seems as if it would be useful for a search for one of these words to return documents that contain another word in the set. The stemming refers to a process that chops off the ends of words so as to remove derivational affixes.

For instance

The result of mapping the text:

the boy's cars are different colors ⇒ the boy car be differ color

Notice that these two texts are not the same.

c. Indexer

Indexer is the module that is responsible to construct the inverted index. Constructing the inverted index following the following steps

- Extract the tokens of each document in the order it exists and put it in data structure having (Term, DocID)
- Sort the above data structure by term. If a term exists more than once in the data structure further sort the list by DocID. This step is core process in indexing.
- Merge multiple term entries in a single document and store its document frequency
- Split content of the data structure into two: Dictionary and Postings

The Dictionary

The dictionary is the central data structure that is used to manage the set of terms found in a text collection. It provides a mapping from the set of index terms to the locations of their postings list.

Postings Lists

The actual index data is stored in the index’s postings list. Each term’s postings list contains information about the term’s occurrences in the collection. A term’s postings list may contain docID, frequency, positional or schema-independent information. The postings data always constitute the vast majority of all the data in the index. Hence, it is too large to be stored in main memory and have to be kept on disk.
For instance given a corpus containing following two documents

Doc1: I did enact Julius Caesar I was killed I’ the Capitol; Brutus Killed me.

Doc2: So let it be with Caesar. The noble Brutus hath told you Caesar was ambitious

Applying a tokenization process, language processing to change words into small letters, stopwords removal, stemming. The following figure shows results after each steps

<table>
<thead>
<tr>
<th>Term</th>
<th>DocID</th>
<th>Term</th>
<th>DocID</th>
</tr>
</thead>
<tbody>
<tr>
<td>I</td>
<td>1</td>
<td>ambitious</td>
<td>2</td>
</tr>
<tr>
<td>did</td>
<td>1</td>
<td>be</td>
<td>2</td>
</tr>
<tr>
<td>enact</td>
<td>1</td>
<td>brutus</td>
<td>1</td>
</tr>
<tr>
<td>Julius</td>
<td>1</td>
<td>brutus</td>
<td>2</td>
</tr>
<tr>
<td>Caesar</td>
<td>1</td>
<td>caesar</td>
<td>1</td>
</tr>
<tr>
<td>I</td>
<td>1</td>
<td>caesar</td>
<td>2</td>
</tr>
<tr>
<td>was</td>
<td>1</td>
<td>caesar</td>
<td>2</td>
</tr>
<tr>
<td>killed</td>
<td>1</td>
<td>capitol</td>
<td>1</td>
</tr>
<tr>
<td>i’</td>
<td>1</td>
<td>did</td>
<td>1</td>
</tr>
<tr>
<td>the</td>
<td>1</td>
<td>enact</td>
<td>1</td>
</tr>
<tr>
<td>Capitol</td>
<td>1</td>
<td>hath</td>
<td>2</td>
</tr>
<tr>
<td>Brutus</td>
<td>1</td>
<td>I</td>
<td>1</td>
</tr>
<tr>
<td>Killed</td>
<td>1</td>
<td>I</td>
<td>1</td>
</tr>
<tr>
<td>me</td>
<td>1</td>
<td>i’</td>
<td>1</td>
</tr>
<tr>
<td>So</td>
<td>2</td>
<td>it</td>
<td>2</td>
</tr>
<tr>
<td>let</td>
<td>2</td>
<td>julius</td>
<td>1</td>
</tr>
<tr>
<td>it</td>
<td>2</td>
<td>killed</td>
<td>1</td>
</tr>
<tr>
<td>be</td>
<td>2</td>
<td>killed</td>
<td>1</td>
</tr>
<tr>
<td>with</td>
<td>2</td>
<td>let</td>
<td>2</td>
</tr>
<tr>
<td>Caesar.</td>
<td>2</td>
<td>me</td>
<td>1</td>
</tr>
<tr>
<td>The</td>
<td>2</td>
<td>noble</td>
<td>2</td>
</tr>
<tr>
<td>noble</td>
<td>2</td>
<td>so</td>
<td>2</td>
</tr>
<tr>
<td>Brutus</td>
<td>2</td>
<td>the</td>
<td>1</td>
</tr>
<tr>
<td>hath</td>
<td>2</td>
<td>the</td>
<td>2</td>
</tr>
</tbody>
</table>
### Term | Doc-Frequency | Term | Doc-List
---|---|---|---
ambitious | 1 | ambitious | 2
be | 1 | be | 2
brutus | 2 | brutus | 1, 2
caesar | 2 | caesar | 1, 2
capitol | 1 | capitol | 1
did | 1 | did | 1
enact | 1 | enact | 1
hath | 2 | hath | 2
I | 1 | I | 1
i’ | 1 | i’ | 1
it | 1 | it | 2
julius | 1 | julius | 1
killed | 1 | killed | 1
let | 1 | let | 2
me | 1 | me | 1
noble | 1 | noble | 2
so | 1 | so | 2
the | 2 | the | 1, 2
told | 1 | told | 2
was | 2 | was | 1
with | 1 | with | 2
Even if the above inverted index is efficient to retrieve word based query it is inefficient to respond to phrase base queries. Biword and position index are capable to model phrases of a document.

**Biword indexes**

It is an approach to handle phrases and it considers pair of consecutive terms in a documents as a phrase. The biword model treats these biwords as a vocabulary term. A phrase is broken down into pair of consecutive terms.

For example, the text Stanford university palo alto would generate the biwords:

- Stanford university
- University palo
- Palo alto

To each biword, store the associated term weight computed using any of the techniques discussed in the previous unit.

However, using biword index returns false negative for single word query and such query shall be interpreted as scanning the whole index for biword containing the term.

**Positional indexes**

It is the most commonly employed solution for phrase indexing. In position index, for each term in the vocabulary it stores postings of the form

docID, frequency: <position1, position2, ...>.

Where, positioni , is a token index in the document frequency: term frequency in the document to, 230:

- 1, 6: <7, 18, 33, 72, 86, 231>;
- 2, 5: <1, 17, 74, 222, 250>;
- 4, 5: <8, 16, 190, 429, 433>;
- 5, 2: <363, 367>;
- 7, 3: <13, 23, 191>; . . . >,
be, 178:

\[
\begin{align*}
1, 2: & <17, 25>; \\
4, 5: & <17, 191, 291, 430, 434>; \\
5, 3: & <14, 19, 101>; \ldots
\end{align*}
\]

Figure 2-4: Position index example. The word to has a document frequency of 230 and occurs 6 times in document 1 at positions 7, 18, 33, etc.

To process a phrase query, access the inverted index entries for each distinct term. Then first examine intersecting postings (i.e., documents containing both terms) and then check the positions of appearance in the document are compatible with the phrase query.

For example: considering the postings for to and be in Figure 2-4 and the query “to be or not to be”. The query terms access are: to, be, or, not. First look for documents that contain both to and be (i.e. intersecting postings lists for to and be). Then look for places in the list where there is an occurrence of be with a token index one higher than a position of to, and then look for another occurrence of each word with token index 4 higher than the first occurrence.

From the above listing the possible match is:

to: 4: <16, 190, 429, 433>;
be: 4: <17, 191, 430, 434>;

Text Query Processing

Introduction

In the previous activity the basic data structure to represent textual data has been discussed in detail. A query is the formulation of a user information need. In its simplest form, a query is composed of keywords and the documents containing such keywords are searched for. In this activity, query processing using the created inverted index will be explained. The created data structure determines the kind of query to be processed. Using inverted list we can process word based queries.

Activity Details

Assume that inverted index has been constructed following the steps explained in previous activity.

The user query can be:

1. **Single-Word Queries**

A word is the most elementary query element. Text document is assumed to be long sequences of words. The identification of word is done applying the tokenization process. The result of word queries is the set of documents containing at least one of the words of the query.
Boolean

Boolean queries are queries where the search terms are connected to each other using Boolean connectors AND, OR and NOT. Usually parenthesis can be used to group search terms. For instance the query: Brutus AND Caesar is a simple Boolean query.

NOT: A NOT operator combined with search terms using AND operator is used to eliminate all documents that contain the NOT term.

OR: The query terms1 OR term2 OR … OR Term n is processed to retrieve the document list for all of the terms and combining them by a union operation (i.e., a document is in the final result set if it is contains at least one of the terms)

AND: The query term1 AND term2 AND … AND term n returns document list that contains all of the terms and combining them with an intersection operation.

To answer such Boolean query, the following steps will be followed

1. Find query elements, query terms, in the Dictionary
2. Retrieve postings for each query term
3. Manipulate postings according to the operators under consideration

Find the intersection of the postings if the terms are connected with AND

Find the union of the postings if the query terms are connected with OR

For the query Brutus AND Caesar

- Locate Brutus in the Dictionary and retrieve its postings
- Locate Caesar in the Dictionary and retrieve its corresponding postings
- Find the intersection of these two postings and return the result as the result of the query

Algorithm: Merge/ Intersection of two postings

Inputs: Postings1, Postings2: List // containing posting list of two query terms
Output: Result: List

Let I = 0; J= 0;

While (I < Postings1.Length AND J < Postings2.Length)
    If (Postings1[I] == Postings2[J])
        Result.Add(Postings1[I])
        I ++ // move the pointer forward
        J++
Else if (Postings1[I] < Postings2[j])
    I++
Else
    J++
END IF
END LOOP
Return Result

The merge algorithm returns the result of merging two postings. It compares docID of each posting; if equal the docID is added to result; if the docID of first is less than the second, it document pointer moves forward in the postings otherwise the document pointer in the second moves forward. This process continue until document pointer in one of the two postings is beyond the size of the list.

Brutus 2, 4, 8, 16, 32, 64, 128
Caesar 1, 2, 3, 5, 8, 13, 21, 34

Dictionary Postings

Applying the merging algorithm to the posting lists of Brutus and Caesar, document ID 2 and 8 are returned as result.

Brutus AND Caesar 2, 8

**Assessment**

1. Draw the inverted index that would be built for the following document collection
   - Doc 1 new home sales top forecasts
   - Doc 2 home sales rise in july
   - Doc 3 increase in home sales in july
   - Doc 4 july new home sales rise

2. State the main steps in inverted index construction

3. State drawbacks of inverted index

4. State the difference type of queries that can be applied to text database.

5. State the major steps that has to be taken to answer Boolean Query

6. What is the time complexity of query engine that uses Boolean query execution mod
Unit 3: Document Clustering

Introduction

The objective of clustering is to partition an unstructured set of objects into groups called clusters. The objects in the same cluster are relatively similar with each other and are dissimilar to objects from other clusters. Clustering has been used in a number of application areas and there are number of clustering algorithms.

Independent of the different clustering algorithms all have two things in common

- Object representation
- A similarity computation method

A clustering algorithm finds a partition of a set of objects that fulfills some criterions based on the similarity measure. The similarity measure is defined on a representation.

The input to a clustering algorithm can be viewed as a graph consisting of vertices corresponding to the objects and edges corresponding to the similarities.

Clustering is an unsupervised learning method. The result is based solely on the similarity between the objects and the clustering algorithm.

In text clustering the objects are texts or documents. These could be grouped in many ways. We are primarily interested in clustering them based on content. In this unit, we will use vector space model in representing content and hierarchical and portioning based clustering algorithms to cluster documents.

Objectives

Upon completion of this unit you should be able to:

- Remember the difference between clustering and classification
- Explain clustering and different types of clustering
- Apply the K-means clustering algorithm in real problem solving
- Use the Hierarchical clustering algorithm
Key Terms

**Clustering:** A group of the same or similar elements gathered or occurring to closely together.

**Classification:** The systematic grouping of objects according to the similarity between their corresponding content and place them in one of the predefined classes or taxonomy.

**Hierarchical clustering:** It is a method of cluster analysis which seeks to build a hierarchy of clusters following either bottom-up (agglomerative) or top-down (Divisive) approach.

**Partitioning based clustering:** It is a method that decomposes a data set into a set of k disjoint clusters.

Learning Activities

1. **Text Similarity**

   Clustering of textual documents demand computing the similarity between their corresponding contents or objects. Computing the similarity between textual content depends on the text representation methods. In this sub-section, two different similarity measures are presented.

   **Vector similarity**

   As explained in unit 1, a text document can be represented in an n-dimensional vector space model. Each dimension of the vector space is a distinct term extracted from the text corpus and the weight of the term is computed as TF, IDF, TF-IDF or Boolean value. The similarity between a pair of textual documents d1 and d2, represented with vectors is computed as the cosine of angle separating the two texts. It is computed using the equation shown.

   \[
   \cosin(d_1,d_2) = \frac{(V \cdot(d_1) \cdot V \cdot(d_2))}{|V \cdot(d_1)||V \cdot(d_2)|}
   \]

   **Jaccard index**

   Given pair of documents represented using bag of words approach, the similarity between them can be computed using Jaccard Index. The Jaccard index, also known as the Jaccard similarity coefficient, is a statistic used for comparing the similarity and diversity of sample sets. The Jaccard coefficient measures the similarity between sets and is defined as the amount of common information (i.e., size of the intersection) divided by the total amount of information (i.e., size of the union of the sample sets). It is formalized as follows:

   \[
   J(A,B) = \frac{|A \cap B|}{|A \cup B|}
   \]

   Notice that: If A and B are both empty, we define \( J(A,B) = 0 \). Thus \( 0 \leq J(A,B) \leq 1 \).
Levenshtein distance

The Levenshtein distance also called edit string is a string metrics for measuring the distance/difference between two sequences. Given two texts A and B, the Levenshtein distance between A and B is the minimum number of edit script (i.e. insertions, deletions or substitutions) required to change A to B or vice versa.

For instance, the Levenshtein distance between “kitten” and “sitting” is 3. The minimal edit script that transforms the former into the latter is:

- kitten → sitten (substitution of “s” for “k”)
- sitten → sittin (substitution of “i” for “e”)
- sittin → sitting (insertion of “g” at the end).

Edit distance has a non-negative cost that rises due to the need to change one to another. The following are basic behavior/characteristics of edit distance:

- \( \text{EditDistance}(A, A) = 0 \), since each string can be trivially transformed to itself using exactly zero operations.
- \( \text{EditDistance}(A, B) > 0 \) when \( A \neq B \), since this would require at least one operation at non-zero cost.
- \( \text{EditDistance}(A, B) = \text{EditDistance}(B, A) \) by equality of the cost of each operation and its inverse.
- Triangle inequality: \( \text{EditDistance}(A, C) \leq \text{EditDistance}(A, B) + \text{EditDistance}(B, C) \)

2. Partitioning Algorithms/ K-means

One of the most common partitioning algorithms is K-means. The algorithm partition the data into \( k \) similar clusters. The algorithm is presented in Figure 3-1.

The initial partitions can be constructed by picking \( k \) objects at random and let them define \( k \) clusters. The K-Means algorithm requires a number of clusters \( K \) as input. That is, one has to guess the appropriate number. These randomly selected objects will be used as the cluster centroid.

Each cluster has its own cluster centroid that contains features representing the cluster. Each text in the text corpus will belongs to one of the \( k \) clusters it is highly similar to. The similarity between a text and cluster is computed using any of the text similarity measures presented in the previous activity.
1. Construct an initial partition consisting of k clusters.

2. For each text \( t \) in the text corpus
   a. Add the text \( t \) in the cluster \( c_j \) in which it is highly similar with
   b. Recomputed the cluster centroid of \( c_i \) after \( t \) is add to it

3. Next

**How the K-Mean Clustering algorithm works?**

Once the cluster of the text is identified, a new centroid that represent members of the cluster will be recomputed in similar fashion. In most cases, a centroid is computed as the mean, median vector or use of few objects representing the clusters. The time complexity of the K-Means algorithm is \( O(knl) \), where \( k \) is the number of clusters, \( n \) the number of objects and \( l \) the number of iterations. In each iteration the cluster centroids and the \( kn \) similarities between all objects and all clusters must be computed.
Hierarchical clustering

Hierarchical algorithms build a cluster hierarchy; clusters are composed of clusters that are composed of clusters. This may be either all the way from single documents up to the whole text set or any part of this complete structure. There are two natural ways of constructing such a hierarchy: bottom-up and top-down also called agglomerative and divisive.

Agglomerative clustering methods starts with individual clusters and iteratively combine similar clusters so as to build bigger clusters as shown in Figure 3-2 hierarchical clustering algorithm.

Start with all documents forming their own cluster.

Until there is only one bigger cluster

Among the current clusters, determine two most similar clusters Ci and Cj

Merge Ci and Cj (i.e. Ci U Cj).

The similarity between clusters can be computed using single-link, complete-link, group average, and distance between Centroid. Among this methods, single-link and complete links are the most popular ones. The single-link method defines the similarity between two clusters as the similarity between the two most similar objects of the two clusters (i.e. similarity is defined as the distance between two closest members). This may result in elongated, locally similar clusters due to chaining effect.

The complete-link method defines the similarity between two clusters as the similarity between the two most dissimilar objects of the two clusters (i.e., the distance between two furthest members).

As cluster A and C; e and b; h and m; are close to each other with similarity values of 0.2 and hence the clusters are combined. The clustering containing e and b is merged with t to get bigger cluster (e, b, t), followed by forming cluster containing a, c, e, b, t; a, c, e, b, t, h, m and finally the bigger cluster containing all elements a, c, e, b, t, h, m, t. The time complexity of the agglomerative algorithms are at least O(n^2) as they all need to compute the similarity between all objects to find the pair of objects that are most similar. In the divisive algorithm is the reverse of agglomerative clustering method. It starts with the bigger cluster and decompose it iteratively into smaller clusters. The division continue until individual clusters are generated.
Assessment

1. When do you say two texts represented in n-dimensional vector space model are similar?

2. Given two documents, d1: d2: compute their similarity using Cosine, Jaccard

3. Given two text: T1 and T1, what EditDistance(T1 and T2)?

4. Discuss the nation of similarity and distance

5. Download Reuters-21578.

6. Keep only documents that are in the classes crude, interest, and grain. Discard documents that are members of more than one of these three classes. Compute a

7. Single-link clustering

8. Complete-link clustering of the documents

9. K-mean cluster, k- is fixed to be 6

Unit Summary

In this unit, notions related to text document clustering are presented. In particular text similarity methods such as cosine similarity, Jaccard approach are presented. Furthermore the similarity methods are used in creating clusters. The two popular clustering approaches – hierarchical clustering and portioning based algorithms are presented and discussed with examples. The hierarchical clustering approach group similar documents using top-down (divisive approach) that consider all documents to be in the initial cluster and divide them one step by step until a cluster related to individual documents are generated. In bottom up approach, individual documents are considered as initial clusters and combined to form merged clusters. This process continue until one cluster is created.
Unit 4: Web Search Engine

Introduction

The web is showing explosive growth due to decentralized content publishing with essentially no central control of authorship. This turned out to be the biggest challenge for web search engine in the process of indexing and retrieving content.

Web contents are authored in various natural language, having thousands of dialects. These demand different forms of natural processing tools – stemming, parsing, and other kinds of linguistic operations. More generally web contents are heterogeneous both in structure and content – variations in grammar and style.

In addition, the web contained truth, lies, contradictions and suppositions on a grand scale. This give raise to the question: which web pages does one trust? Answering such question demands link analysis and page ranking.

The pages in web, can be either static or dynamic. Static web pages are those whose content does not vary from one request for that page to the next. Whereas dynamic pages are typically mechanically generated by an application server in response to a query to a database.

The static Web consists of HTML pages together the hyperlinks between them to form a directed graph in which each page is a node and each hyperlink is a directed edge connecting nodes (page).

Figure 4-1: A sample small web graph having six pages labelled A through F.
The anchor tag <a> is used to define the link between pages and store anchor text. Users look for information extracted from such pages or may want to navigate through the pages. Thus in this unit, taxonomy of Web searching, search engine architecture and web directory is presented.

**Objectives**

Upon completion of this unit you should be able to:

- Describe the different components of Web search systems
- Explain the architecture of a Search Engine and along with its components
- Differentiate Search engines and Web Directories
- Understand Web Information Retrieval

**Key Terms**

- **Web** also called WWW is a system of interlinked hypertext documents that are accessed via the Internet.
- **A Web Search Engine** is a software system that is designed to search for information on the World Wide Web.
- **Search tools** are computer software used to search data (as text or a database) for specified information; also a site on the World Wide Web that uses such software to locate key words in other sites.
- **Directories** are not a search engine and does not display lists of web pages based on keywords; instead, it lists web sites by category and subcategory. Most web directory entries are also not found by web crawlers but by humans

**Learning Activities**

1. **Taxonomy of Web Search**

Classic IR is inherently predicated on users searching for information. In the web context the “need behind the query” is often not informational in nature. Searching in a Web can be classified into three based on the type of information to be searched.
Navigational. The immediate intent is to reach a particular site.

Informational. The intent is to acquire some information assumed to be present on one or more web pages.

Transactional. The intent is to perform some web-mediated activity.

**Navigational queries.** The purpose of such queries is to reach a particular site that the user has in mind, either because they visited it in the past or because they assume that such a site exists. Some examples are

- African Union. Probably target www.au.int/
- Ethiopian Airlines. Probable target www.ethiopianairlines.com/
- Compaq. Probable target http://www.compaq.com
- Masai mara. Probable target www.maasaimara.com/

This type of search is sometimes referred as “known item” search in classical IR, but is mostly used in the evaluation of various systems.

**Informational queries.** The purpose of such queries is to find information assumed to be available on the web in a static form in the target document is not created in response to the user query. The informational queries are closest to classic IR.

**Transactional queries.** The purpose of such queries is to reach a site where further interaction will happen. This interaction constitutes the transaction defining these queries. The main categories for such queries are shopping, finding various web-mediated services, downloading various type of file (images, songs, etc), accessing certain data-bases (e.g. Yellow Pages type data), finding servers (e.g. for gaming) etc.

It is very hard to evaluate the result of transactional queries in terms of classic IR. Binary judgment might be the only option to use such as classifying the result as appropriate, non-appropriate. However most external factors important for users (e.g. price of goods, speed of service, quality of pictures, etc) are usually unavailable to generic search engines.

2. **Search Engines**

A web search engine is a tool that allows users to find information on the world wide web. The content to be search may have different format, syntax, semantic, content organization. The search engine provide a consistent interface that allows users to search in constantly updating data structure. One main difference between standard IR systems and the Web is that, in the Web, all queries must be answered without accessing the text (that is, only the indices are available). Otherwise, that would require either storing locally a copy of the Web pages (too expensive) or accessing remote pages through the network at query time (too slow). Thus this has difference in indexing, searching algorithm and the query language.
3. Search Engine Architecture

Most search engines use a centralized crawler-indexer architecture as shown in Figure 4-2 containing crawler, indexer, Query Engine and Interface. Crawlers also called robots, spiders, wanderers, walkers or knowbots are programs (software agents) that traverse the Web sending new or updated pages to a main server where they are indexed. A crawler does not actually move to and run on remote machines, rather the crawler runs on a local system and sends requests to remote Web servers.

The crawler component may start crawl the Web with a set of URLs and from there extract other URLs which are followed recursively in a breadth-first or depth-first fashion. For that reason, each engines allow users to submit top Web sites that will be added to the URL set. A variation is to start with a set of popular URLs as such sites can have information of frequently requested pages. Both cases work well for one crawler, but it is difficult to coordinate several crawlers to avoid visiting the same page more than once. In general the crawler component contains and perform the following activities.

a. It contains URLs yet to be fetched in the current crawl (in the case of continuous crawling, a URL may have been fetched previously but is back in the frontier for re-fetching).

b. A DNS resolution to determine the web server from which to fetch the page specified by a URL.

c. Uses the http protocol to retrieve the web page at a URL.

d. Parse the page and extracts the text and set of links from a fetched web page.

e. Eliminate duplicates if an extracted link is already crawled or has recently been fetched.

The indexer creates inverted indexes so that online systems and users can search. The index is used in a centralized fashion to answer queries submitted from different places in the Web.
The user interface component is responsible to formulate search query and present answer using search interface and answer interface respectively. The query interface may allow basic and complex query. The basic search query interface is a box where one or more words can be typed and it passes it the query engine component. The advanced query interface, Figure 4-3 shows sample from Google, allows formulating query with all keywords, phrase based query, range based search, and narrow the search result using language specific search, region, last update etc. The query engine performs the searching using the index created by the indexer.

![Advanced Google searching interface](image)

Figure 4-3: Advanced Google searching interface

The result of searching is send back to the user and presented in answer interface. The answer is usually consists of top ranked Web pages. Each entry in the list includes some information about the document it represents – URL, title, the date when the page was indexed, snippet of the page.

4. **Web Directories**

In addition to search engines, there are number of Web tools which are based on browsing and searching. Web Directories are Web tools that provide browsing and searching capability and the results are much focused and more relevant.
A web directory is a directory on the WWW specializes in linking to other web sites and categorizing those links. Web Directory is not a search engine and does not display lists of web pages based on search query; rather it lists web sites by category and subcategory. Most web directory entries are also not found by web crawlers but by humans. Directories are hierarchical taxonomies that classify human knowledge. The categorization is usually based on the whole web site rather than one page or a set of keywords, and sites are often limited to inclusion in only a few categories. Web directories often allow site owners to submit their site for inclusion, and have editors review submissions for fitness. Web directories are also called catalogs, yellow pages, or subject directories. For instance, the World Wide Web Virtual Library (http://www.vlib.org/), Yahoo!(www.yahoo.com), Snap( http://www.snap.com,)LookSmart (http://www.looksmart.com), Open Directory (http://dmoz.org/),Lycos Subjects (http://www.lycos.com/),NewHoo( www.newhoo.com )and etc.

The main advantage of this technique is that if we find what we are looking for, the answer will be useful in most cases. On the other hand, the main disadvantage is that the classification is not specialized enough and that not all Web pages are classified.

**Assessment**

1. What are the main components of a typical search engine?
2. What are the main differences between search engine and web directories?
3. State the advantage of metasearch engines?
4. What are the main tasks performed during web crawling?
5. What makes indexing a web page different from indexing simple text document? Discuss similarity and differences.
6. Assume that you are assigned to the design of a typical search engine associated to your local language. State the different components that shall be in the system and justify the necessity of the component/

**Unit Summary**

In this unit, web search engine is presented in detail. Classification of web searching into three classes based on the type of search as navigational, informational and transactional is presented. A typical search engine has web crawler, indexer, query engine and user interface as main components. These components are presented.
Unit 5: Evaluation in Information Retrieval

Introduction

Before implementing an information retrieval system, an evaluation of the system is usually carried out. The type of evaluation to be carried out depends on the objective of the system. The first type of evaluation to be considered is functional analysis in which the specified functionalities are tested one by one. The functional analysis include error checking. After the functional analysis, the performance of the system will be evaluated.

The most common measures of system performance are time and space. The shorter the response time, the smaller the space used, the better the system is considered to be. There is an inherent tradeoff between space complexity and time complexity which frequently allows trading one for the other.

In a system designed for providing data retrieval, the response time and the space required are usually the metrics of most interest and the ones normally adopted for evaluating the system. Thus, the interest is in looking the performance of the index structure, the interaction with the operating system, the delays in communication channels and the overheads introduced by the many software layers. Such forms of evaluation are considered as performance evaluation.

In a system designed for providing information retrieval, other metrics, besides time and space, are also of interest. As the user query request is inherently vague, the retrieved documents are not exact answers and have to be ranked according to their relevance to the query. Such relevance ranking introduces a component which is not present in data retrieval systems and which plays a central role in information retrieval. Thus, information retrieval systems require the evaluation of how precise is the answer set. This type of evaluation is referred to as retrieval performance evaluation.

In this unit discussion on retrieval performance evaluation for information retrieval system will be made. Such an evaluation is usually based on a test reference collection and on an evaluation measure. The test reference collection consists of a collection of documents, a set of example information requests, and a set of relevant documents (provided by specialists) for each example information request. i.e., given a retrieval strategy S, the evaluation measure quantifies (for each example information request) the similarity between the set of documents retrieved by S and the set of relevant documents provided by the specialists. This provides an estimation of the goodness of the retrieval strategy S.
Objectives

Upon completion of this unit you should be able to:

- Understand the importance evaluation in IR
- Explain how to measure the retrieval performance of IR system
- State how to evaluate the precision, recall, f-score and accuracy of IR system

Key Terms

Recall is the fraction of the documents that are relevant to the query that are successfully retrieved.

Precision is the fraction of retrieved instances of documents that are relevant.

Harmonic Mean is a measure that associate equal weight to recall and precision in measure the retrieval efficiency of an information retrieval system.

Learning activities

1. Recall and Precision

Consider an example information request I (of a test reference collection) and its set R of relevant documents. Let IRI be the number of documents in this set. Assume that a given retrieval strategy (which is being evaluated) processes the information request I and generates a document answer set A. Let IAI be the number of documents in this set. Further, let IRaI be the number of documents in the intersection of the sets R and A as shown in Figure 5-1.

The recall and precision measures are defined as follows.

Recall is the fraction of the relevant documents (the set R) which has been retrieved i.e.,

Recall=IRaI/IRI

Precision is the fraction of the retrieved documents (the set A) which is relevant i.e.,

Precision=IRaI/IAI
Recall and precision, as defined above, assume that all the documents in the answer set $A$ have been examined (or seen). However, the user is not usually presented with all the documents in the answer set $A$ at once. Instead, the documents in $A$ are first sorted according to a degree of relevance (i.e., a ranking is generated). The user then examines this ranked list starting from the top document. In this situation, the recall and precision measures vary as the user proceeds with her/his examination of the answer set $A$.

Thus, proper evaluation requires plotting a precision versus recall curve.

For instance, for a given query $q$, assume that a set $R_q$ contains the relevant documents for $q$ has been defined in the eye of specialist and represented as:

$$R_q = \{d_3, d_5, d_9, d_{25}, d_{39}, d_{44}, d_{71}, d_{89}, d_{123}\}$$

Let us assume that the information retrieval algorithm returns the ranked document set $A$.

1. $d_{123}$  
2. $d_{54}$  
3. $d_{56}$  
4. $d_6$  
5. $d_8$  
6. $d_9$  
7. $d_{12}$  
8. $d_{88}$  
9. $d_{511}$  
10. $d_{45}$  
11. $d_{129}$  
12. $d_{250}$  
13. $d_{113}$  
14. $d_{3}$  
15. $d_{3}$
The documents that are relevant to the query q are marked with a bullets after the document number.

Computing the recall and precision on the ranked list and relevant documents of A from the top document is shown in Table 5-1.

<table>
<thead>
<tr>
<th></th>
<th>Precision</th>
<th>Recall</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>d123•</td>
<td>100%</td>
</tr>
<tr>
<td>2</td>
<td>d54</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>d56•</td>
<td>67%</td>
</tr>
<tr>
<td>4</td>
<td>d6</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>d8</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>d9•</td>
<td>50%</td>
</tr>
<tr>
<td>7</td>
<td>d511</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>d129</td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>d125</td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>d25•</td>
<td>40%</td>
</tr>
<tr>
<td>11</td>
<td>d88</td>
<td></td>
</tr>
<tr>
<td>12</td>
<td>d45</td>
<td></td>
</tr>
<tr>
<td>13</td>
<td>d250</td>
<td></td>
</tr>
<tr>
<td>14</td>
<td>d113</td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>d3•</td>
<td>50%</td>
</tr>
</tbody>
</table>

Table 5-1: Precision and Recall value of an Example

Table 5-1 shows Precision and recall value computed for the relevant documents identified of the information retrieval algorithm. The computation is done progressively. Precision value of the top element is 1 (100%) and this document corresponds to 1/10 (i.e. recall value of 10%) of all relevant documents of Rq. Second the document d56 which is ranked as the third element is the next in the relevant list. This document has precision value of 66% (i.e., 2/3 two documents out of three are relevant) and recall value of 20% (two of the ten relevant have been seen). The resulting Precision value along with the recall is shown in Figure 5-2.
Normally, the precision and recall value of information retrieval system will be defined for different test queries and average of the precision at given recall level is computed as follows:

\[ p(r) = \frac{\sum_{i=1}^{N_q} (p_i(r))}{N_q} \]

Where:

- \( p(r) \) is the average precision at the recall level \( r \)
- \( N_q \) is the number of queries used and
- \( p_i(r) \) is the precision at recall level of \( r \) for the \( i \)th query

Average precision versus recall figures are now a standard evaluation strategy for information retrieval systems and are used extensively in the information retrieval literature. They are useful because they allow us to evaluate quantitatively both the quality of the overall answer set and the breadth of the retrieval algorithm.

**F-score**

Even if recall and precision are used extensively to evaluate the retrieval performance of retrieval algorithms, they are criticized for the following reasons

- Proper estimation of maximum recall for a query requires detail knowledge of all the documents in the collection.
- Recall and precision are related measures which capture different aspects of the set of retrieved documents. In many situations, the use of a single measure which combines recall and precision could be more appropriate.
- Recall and Precision are easy to define when a linear ordering of the retrieved documents is enforced.
Harmonic mean also called F score is a single measure which combines recall and precision. It is defined as:

\[
F(j) = \frac{2 \times r(j) \times p(j)}{r(j) + P(j)}
\]

Where:

- \( p(j) \) is the precision for the \( j \)-th document in the rank
- \( r(j) \) is the recall for the \( j \)-th document in the ranking
- \( F(j) \) is the harmonic mean of \( r(j) \) and \( o(j) \)

The harmonic mean is a value between 0 and 1 inclusive. It assumes a maximum value when both the precision and recall values are 1.

For instance, Table 5-2 shows the harmonic mean of the relevant documents shown in Table 5-1.

<table>
<thead>
<tr>
<th></th>
<th>Precision</th>
<th>Recall</th>
<th>Harmonic mean</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. d123</td>
<td>100%</td>
<td>10%</td>
<td>0.18</td>
</tr>
<tr>
<td>2. d56</td>
<td>67%</td>
<td>20%</td>
<td>0.31</td>
</tr>
<tr>
<td>3. d9</td>
<td>50%</td>
<td>30%</td>
<td>0.38</td>
</tr>
<tr>
<td>4. d25</td>
<td>40%</td>
<td>40%</td>
<td>0.4</td>
</tr>
<tr>
<td>5. d3</td>
<td>50%</td>
<td>50%</td>
<td>0.5</td>
</tr>
</tbody>
</table>

*Table 5-2: Harmonic mean value of relevant documents*

Considering Table 5-2 document d3 is more relevant as its f-score is higher than all the others.

2. Precision Recall graph

Precision, recall, and the F measure are set-based measures. They are computed using unordered sets of documents. In evaluating ranked retrieval results which are related to search engines these measures aren’t enough and should be extended.

In a ranked retrieval context, appropriate sets of retrieved documents are naturally given by the top \( k \) retrieved documents. For each such set, precision and recall values can be plotted to give a precision-recall curve.
Typically, Precision and Recall are inversely related, i.e. as Precision increases, recall falls and vice-versa. A balance between these two needs to be achieved by the IR system, and to achieve this and to compare performance, the precision-recall curves come in handy.

Precision-recall curves have a distinctive saw-tooth shape: if the \((k+1)\)th document retrieved is non-relevant then recall is the same as for the top \(k\) documents, but precision has dropped. If it is relevant, then both precision and recall increase, and the curve jags up and to the right. It is often useful to remove these jiggles and the standard way to do this is with an interpolated precision.

Suppose, we have a database with 100 documents, out of which 60 are relevant to a particular keyword base query. If an IR system that uses algorithm 1 returns a total of 50 documents, out of which 40 are relevant, the precision for this system is \(40/50 = 0.8\) and the recall is \(40/60 = 0.66\).

Assume that another system (that uses algorithm 2) returns only 10 documents and at least 9 of them are relevant. This would increase the precision to 0.9 \((9/10)\) but decrease its recall to 0.15 \((9/.60)\).

Thus, the above two IR's systems need to be analyzed and compared. One would choose the appropriate system depending on the need (high precision, or more data with false positives allowed).

This trade-off between precision and recall can be observed using the precision-recall curve, and an appropriate balance between the two obtained.

![Figure 5-3: Comparison in Precision Recall Space extracted from the work of Jesse and Mark](image)
Assessment

1. Compare and contrast recall and precision

2. What are the key challenges stated by recall and precision in measure the retrieval performance of IR system?

3. What are the drawback of harmonic mean in the context of typical search engine? Discuss your answer with example
Unit Summary

In this unit, different methods in evaluating the performance of IR system are presented. The most known approach in evaluation based on a test reference collection and on an evaluation measure is used. The test reference collection consists of a collection of documents, a set of example information requests, and a set of relevant documents (provided by specialists) for each example information request. The goodness of the result of retrieval is measured using recall, precision and f-score methods.

1

Jesse Davis, Mark Goadrich The Relationship Between Precision-Recall and ROC Curves http://pages.cs.wisc.edu/~jdavis/davisgoadrichcamera2.pdf
Module Summary

In this module notions related information retrieval is presented. The module starts by revising concepts in principles of database. A database is a repository that store data. This repository needs to be modelled in nice way to represent real world entities and relationship in between. In particular, E-R modelling concept that represents a database conceptually and normalization are recapped. Normalization steps are taken to minimize the existence of redundancy and the steps mainly results to set of related relations.

Unit 1 presents basic notion of Information retrieval focusing mainly on the two popular text representation models – Boolean and vector space. The Boolean model presents texts and query using Boolean or logical expression whereas vector space model represents documents/query using its key terms in multi-dimensional space. The weight of each dimension in vector space model is computed using binary value, term frequency or term frequency and inverse document frequency. The notion of vector space model is used to compute the similarity between documents or between documents and query using cosine similarity approach.

Unit 2 presents processing textual query which involves changing the user query into set of keywords, matching it against the text corpora represented in the system. The query can be a single word, Boolean expression, phrase based etc. The document retrieval is different from structured database query as the result of the query is not exact match.

In unit 3, notions related to text document clustering are presented. In particular text similarity methods such as cosine similarity, Jacaard approach are presented. Furthermore the similarity methods are used in creating clusters. The two popular clustering approaches – hierarchical clustering and portioning based algorithms are presented and discussed with examples. The hierarchical clustering approach group similar documents using top-down (divisive approach) that consider all documents to be in the initial cluster and divide them step by step until individual documents form a cluster by themselves. In bottom up approach, individual documents are considered as initial clusters and combined to form merged clusters. This process continue until one bigger cluster that contains all documents is created.

In unit 5, web search engine is presented in detail. Methodologically web searching approaches are classified into three classes based on the type of search as navigational, informational and transactional. A typical search engine has web crawler, indexer, query engine and user interface as main components.

In unit 6, different methods in evaluating the performance of IR system are presented. The most know approach in evaluation is based on a test reference collection and on an evaluation measure. The test reference collection consists of a collection of documents, a set of example information requests, and a set of relevant documents (provided by specialists) for each example information request. The goodness of the result of retrieval is measured using recall, precision and f-score methods.
## Module Assessment

1. What is E-R diagram? And why do we need it?
2. Define the second normal form.
3. What do we need formalization in relational database design?
4. State the two popular text representation models in IR
5. What is similarity?
6. What is inverted index?
7. What are the main reasons to build index in document query/retrieval?
8. Discuss the single link clustering approach
9. What are the pre-conditions in the k-means clustering? Discuss the clustering approach.
10. What are the main differences between web directories and search engine?
11. State the different components of a typical search engine.
The African Virtual University Headquarters
Cape Office Park
Ring Road Kilimani
PO Box 25405-00603
Nairobi, Kenya
Tel: +254 20 2528333
contact@avu.org
oer@avu.org

The African Virtual University Regional Office in Dakar
Université Virtuelle Africaine
Bureau Régional de l’Afrique de l’Ouest
Sicap Liberté VI Extension
Villa No.8 VDN
B.P. 50609 Dakar, Sénégal
Tel: +221 338670324
bureau regional@avu.org