Week 1 & 2 - Introduction to Information Retrieval

Information Retrieval

What is information Retrieval (IR)? Information retrieval involves finding material – typically documents (e.g., emails, twitter posts, articles, etc) that is unstructured and satisfies specific information needs within large collections typically stored on computers e.g., web search engines, digital libraries and recommender systems

Key Concepts

Handling large quantities of unstructured information

- **Challenge:** Managing and retrieving relevant information from vast amounts of unstructured data is complex.
- **Example:** Finding 10 articles that match specific political content is not straightforward; it involves nuanced understanding rather than binary distinctions

Classical Information Retrieval

- **Objective:** Given a query document, identify and retrieve relevant documents from a collection
- **Approach:** Traditionally focused on matching documents based on relevance to the given query

Week 2 Introduction

The field of information retrieval (IR) has evolved as a natural extension of data retrieval, designed to handle vast amount of unstructured amount of unstructured information available today

Key differences between data retrieval and information retrieval:

- **Data Retrieval:** works with well structured collections, where each item is atomic (indivisible) and has a clearly defined meaning. Queries in data retrieval are typically precise and expressed in languages like SQL
- **Information Retrieval:** focuses on retrieving semi-structured or unstructured documents such as those written in natural language. Unlike data retrieval, the content in IR is often ambiguous and lacks clear structure.

Main characteristics of information retrieval

- **Ambiguity in queries:** instead of dealing with well defines queries, IR systems must interpret the meaning of the user queries which are often vague or ambiguous
- **Ranking & Relevance:** Rather than returning a fixed set of results an IR system ranks documents based on how well they meet the user's information needs, attempting to assess the semantic content of each document

Information Filtering

Information Filtering refers to systems designed to identify and provide relevant information to users based on their ongoing or long term needs. Although information retrieval (IR) share many similarities, such as similar representations, mathematical modes and comparison techniques, but they differ in two key areas.

- **Nature of the information need**

- In information retrieval (IR), the user's information need is usually a one-time specific query (e.g., search for a document or a fact)
- In information filtering, the information need is typically continuous or long-term. The system is constantly monitoring new information to identify what is relevant to the user over time (e.g., news updates, email filtering or recommendation systems)

- **Nature of the document collection:**
  - In Ir, the collection is usually static or fixed. The system searched through an existing set of documents to find relevant ones.
  - In information filtering, the document collection is viewed as a dynamic stream of incoming information. The system makes relevance decision in real-time as new documents or pieces of information flow in, often without access to the entire collection at once (e.g., filtering social media feeds or email messages)

User Role

In traditional information retrieval and library systems, the user role was pretty well defined and understood in that the user formulated a query, viewed the results and possibly offered feedback to the system. These systems were often centralized, with the collection of documents or information stored in one place, and the user's actions were relatively limited to searching and feedback. In **modern IR systems**, however, user behaviour has **evolved significantly**, especially with the rise of the hyper text paradigm (e.g., the web). Now, users often mix querying with browsing by following links, navigating through different sources and interacting with it in a more dynamic way. In doing so the user's information need often evolves and changes during the interaction. This presents new challenges for IR researchers as systems must now accommodate these shifting needs and support a more interactive and exploratory search process

IR System Architecture

At a high level, an IR system consists of several key components.

- **Document set/collection:** This is the collection of documents the system will search through (e.g., web pages, articles, tweets)
- **Queries:** These present users information needs, typically entered as search terms or natural language questions.
- **Pre-processing Components:** These are techniques like stemming or tokenization used to prepare the text in both the documents and the query for further processing

In most cases, the documents and the query are initially in natural language. They're both processed into an internal representation (e.g., vectors, terms, weights), which is suitable for comparison using the chosen algorithm. The exact format of this representation and the comparison methods will depend on the IR model used (e.g., vector model, or boolean model)

The comparison algorithm calculates an estimate of relevance (or similarly) for each document in relation to the query determining how likely the document is to satisfy the user's information need. This allows the system to rank documents based on their relevance to the query. The most relevant (top-ranked) documents one presented to the user. Finally, the user feedback module allows the user to provide feedback on the returned results. This feedback is used to improve the query by: incorporating new terms or re-weighting existing terms. The goal is to refine the query to better align with the user's information need, thereby the quality of the search results.

Pre-processing Overview

The document pre-processing phase involves applying a well known set of techniques to the document collection to convert it to a more structured format that can be efficiently searched an indexed and is more suitable for the task at hand. The idea is to make the data clearer, more consistent and therefore suitable for retrieval tasks

Stop word removal

This is the process whereby some extremely common words, which appear to be of little value in helping select document's matching a user's need are excluded from the vocabulary entirely **(these are called stop words)**. The general strategy for determining a stop list is target the terms by collection frequency and then take the most frequent terms, often hand filtered for their semantic content relative to the domain of the documents being indexed, as the stop list. The members of this list are then disregarded during indexing. Using a stop list can sometimes backfire, as certain queries like "let it be" or "to be or not to be" made up entirely of stop words, could be stripped of their meaning leading to the loss of important phrases in search results.

Example: "The quick brown fox jumps over the lazy dog"
After stop word removal: "quick brown fox jumps lazy dog"

Stemming & Lemmatization

For grammatical reasons, documents are going to use different forms of a word, such as organize, organizes, organizing etc. Additionally, there are families of derivationally related words with similar meanings, such as democratic, democracy and democratization. It would be most optimal for a search of one of these words to return documents that contain another word in this set. The goal of both stemming and lemmatization is to reduce these types of words to their root form. **However these two slighlty differ**

- **Stemming:** usually refers to a crude heuristic process that chops off the ends of words in the hope of achieving this goal correctly *most of the time* (this can be **incorrect** in cases such as **skies** and **skiing** which would both be cut to ski and lead to incorrect associations or irrelevant search results)

  Example: Running -> Run & Ran -> Ran **(Note no change)**

- **Lemmatization:** usually refers to doing things properly with the use of a vocabulary or morpholigic analysis of words, normally aiming to remove inflectional endings only to return the base form or dictionary form of a word which is better known as a lemma

  Example: Running -> Run & Ran -> Run **(Note a change)**

Tokenization

Tokenization is the process of breaking a character sequence or document into smaller units called tokens, often removing characters in a document, a **type** is the class of all tokens with the same sequence, and a **term** is a normalized type included in the IR system's dictionary

Example: "Friends, Romans, countrymen, lend me your ears"
After Tokenization: "| Friends | Romans | Countrymen | Lend | Me | Your | Ears |"

Tokenization involves decisions like handling apostrophes (e.g., "O'Neill" is "O Neill" or "Oneill" or "Neill" etc) and contractions (e.g., "aren't" is it "arent" or "aren t" etc). The chosen strategy impacts how queries match the indexed text, as splitting on non-alphanumeric characters could lead to variations in the tokens produced, affecting search results.

IR Models

Various information retrieval models have been proposed over time, and they can generally be categorized into 3 main types: **Boolean Models, Vector Based Models, Probabilistic Models.**

An IR model is essentially a framework used to represent documents and queries and to asses how relevant a document is to a user's query

It can be understood as a tuple consisting of the following components:

- **D:** The logical representations of documents (how documents are structured or represented in the system)
- **Q:** The logical representation of the user's information needs (how the user's query is represented)
- **F:** The framework or method used to model the relationships between the representations of documents and queries. This can involve different mathematical approaches depending on the modal (e.g. set theory, vectors or probability)
- **R(q_i, d_j):** The ranking function that generates a ranking of documents based on their estimated relevance to the query q_i. this is how the system decides which datasets are most relevant to the user's query

In most models, documents and queries are represented by a set of index terms (e.g., keywords) denoted as $t_1, t_2, \ldots, t_n$. A weight $w_{i,j}$ is assigned to each term t_i in the document d_j, indicating its importance within the document. Similarly queries are also often treated as a set of terms, each with its own weight representing its significance within the query

Boolean Model

The **Boolean Model** is based on the simplistic concept that if a document contains a term or set of terms that satisfy a query **(a boolean expression)**, then the document is considered relevant to that query. It relies on set theory and Boolean algebra. while it is a simple and commonly used model, its advantages – such as simplicity and clear formalism – can often be overshadowed by its limitations.

These limitations include:

- Users often Struggle to formulate Boolean expressions correctly due to the subtle difference in natural language usage and the Boolean algebraic meanings of operations like **AND** or **OR**. which can behave differently than their typical usage in everyday language.
- The model struggles to handle natural language complexities such as **synonymy** (where different words have similar meanings) and **polysemy** (where one word can have multiple meanings), making it less effective at capturing a users true intent of their query
- The model treats all matching terms equally, ignoring the frequency of terms within a document, which means they **cannot distinguish** between documents where a term is highly relevant (appears more frequently) and those where it appears only once)
- Documents are considered either relevant or non-relevant, there is no potential for partial matching; no real ranking allowed – this can lead to exclusion of useful documents.
- Terms in a document are considered independent of each other, this ignores the relationship between terms (such as phrases or contextual meaning), which can lead to less accurate results. For example if you search "**artificial intelligence**" the model will retrieve documents that contain both "**artificial**" and "**intelligence**" anywhere in the text, even if they are far apart and unrelated. This means that a document discussing say **emotional intelligence** and **artificial plants** could be considered equal to a document discussing **artificial intelligence**

Vector Space Model

The **Vector Space Model** attempts to improve upon the **Boolean Model** by removing the limitation of binary weights for index terms. In the vector space model, terms can have a non-binary value for both queries and documents.

These term weights are used in comparison between documents and queries. We can sort the documents based on their degree of similarity and return a ranked list to the user.

Each term in both documents and queries will have an associated weight. Hence we can represent documents and queries as n-dimensional vectors, where each dimension corresponds to a specific term in a system's vocabulary.

$$\vec{d_j} = (w_{1,j}, w_{2,j}, \ldots, w_{n,j})$$
$$\vec{q} = (w_{1,q}, w_{2,q}, \ldots, w_{n,q})$$

$\vec{d_j}$ represents the vector for the documents i, with each $w_{i,j}$ being the weight of term i in the document.
$\vec{q}$ represents the vector for a query, with each $w_{i,q}$ being the vector weight of term i in the query.

We can calculate the similarity between a document and a query by calculating the **similarity between vector representations**.

$$\textbf{Formula 1:}\ \text{sim}(d_j, q) = \frac{\vec{d_j} \cdot \vec{q}}{|\vec{d_j}||\vec{q}|}$$

$$\Rightarrow \textbf{Formula 2:}\ \text{sim}(d_j, q) = \frac{\sum_{i=1}^{n} w_{i,j} w_{i,q}}{\sqrt{\sum_{i=1}^{n} w_{i,j}^2}\sqrt{\sum_{i=1}^{n} w_{i,q}^2}}$$

Formula 1

Calculates the cosine between 2 vectors:

- d_j**:** Represents the document vector,
- q**:** represents the query vector
- **The dot product** $\vec{d_j} \cdot \vec{q}$ **:** calculates the sum of the product of corresponding terms (features) from the document and the query
- **The magnitudes** $|\vec{d_j}||\vec{q}|$**:** Represent the lengths of the document and query vectors, respectively
- The cosine similarity value will range from -1 to 1:
  - **1:** means the document and query are identical in terms f the vector space (perfect match)
  - **0:** means they are completely unrelated (orthogonal)
  - **-1:** would indicate an inverse relationship (not common in typical IR applications, as term weights are usually non-negative)

Formula 2

Expanded cosine similarity (Component form)

- $\sum_{i=1}^{n} w_{i,j} w_{i,q}$**:** Represents the documents dot product. It sums the product of weights $w_{i,j}$ (the weight of term i in document j) and $w_{i,q}$ (the weight of term i in the query q) for all terms i in the vocabulary

- $\sqrt{\sum_{i=1}^{n} w_{i,j}^2}\sqrt{\sum_{i=1}^{n} w_{i,q}^2}$**:** Represents the magnitudes of the document and query vectors, respectively. The normalization ensures that the length of the vectors is accounted for, providing a more accurate similarity measure.

Why it matters in IR

- **Similarity measure:** cosine similarity gives a numeric value (ranging from 0 to 1 for non negative vectors) that measures how similar a document is to a query.
- **Ranking documents:** Documents with a higher cosine similarity to the query are considered more relevant and are ranked higher in search results.

Weighting Scheme

In IR, the similarity between a query and a document is influenced by two main factors:

- The number of **co-occurring terms** between the document and the query.
- The **weights** assigned to these terms, both in the document and in the query.

Assigning the correct weights to terms is crucial for determining relevance. Salton and Fox proposed two key heuristics for term weighting.

- **Term Frequency (tf):** This measures how frequently a term appears in a document, a term that occurs in a document likely has a higher performance in describing that document
- **Inverse Document Frequency (idf):** This considers how common or rare a term is across the entire document collection. A term that appears in many documents (e.g., "the") does little to distinguish between them, so its weight is reduced.

Instead of using **collection frequency (cf)** – the total number of times a term appears in documents **document frequency (df)** is more commonly used. Df measure how many individual documents contain th term, and is a better way to gauge the term's ability to distinguish between documents.

For example: terms like "insurance" and "try" might have a similar collection frequency, but "insurance" is more informative because it appears in fewer documents leading to a higher idf value

The **td-idf** weighting scheme combines these two heuristics to assign a composite weight to each term in each document.

$$w_{i,j} = f_{i,j} \times \log\left(\frac{N}{n_i}\right)$$

Where:

- $f_{i,j}$ is some function of the frequency of term t_i in document d_j
- N is the number of documents in the collection
- n_i is the number of documents in the collection that contain term t_i

This means:

- **High tf-idf:** A term receives a high weight if it appears in a few documents **(providing strong discriminating power for those documents)**
- **Low tf-idf:** A term receives a lower weight if it occurs infrequently in a document or appears in many documents, **as it offers less relevance**
- **Zero tf-idf:** If a term appears in nearly all documents, its tf-idf weight is close to zero because **it provides no distinguishing power**

Common & Rare Words

- **Common words:** (e.g., "the," "is") are prevalent across most documents but are not informative for distinguishing between them. They can be effectively removed from the lexicon without losing much useful information. This concept is the basis for stop word removal, which we implement in TF-IDF as well

- **Rare words:** on the other hand, occur infrequently and may not even appear in many queries. Even when they do appear, they can have ambiguous or negligible importance. For example, rare words might have meanings specific to certain contexts, making them unreliable for general document discrimination.