Problem 1 (What comes first). Consider a repository of 1000 documents of which 28 are relevant to a user query. There are two search engines A and B. Search engine A returns 10 documents and search engine B returns 20 documents. An expert says that 7 of the 10 documents returned by A are relevant while 5 of the 20 returned by B are relevant.

(a) Calculate the precision, recall and accuracy metrics of the search engines.

(b) Which search engine is the winner in terms of its i) accuracy and ii) recall?

(c) Accuracy is not the best metric for IR evaluation. Briefly justify this statement based on the previous results.

(d) Let the order of relevant documents in the corpus be 1 (most relevant), 2, ..., 28 (least relevant). We mark the documents retrieved by seach engines A and B, by relevance and \( \hat{R} \) if not relevant and get the following results.

\[
\begin{array}{c|c|c|c|c|c|c|c|c}
A & R = 0 & R = 1 & B & R = 0 & R = 1 \\
\hline
R = 0 & 969 & 21 & R = 0 & 957 & 23 \\
R = 1 & 3 & 7 & R = 1 & 15 & 5 \\
\end{array}
\]

Table 1: Contingency table for the two search engines.

Solution.

(a) The metrics can be obtained from the contingency tables for the two search engines, where the actual relevant documents are labelled \( R = 1 \) and the documents the search engine tags as relevant is labelled as \( \hat{R} = 1 \).

Precision is the fraction of relevant documents over all the documents that the search engine retrieves. So,

\[
Precision_A = \frac{7}{10} = 70 
\]

\[
Precision_B = \frac{5}{20} = 25 
\]

Recall is the fraction of the relevant retrieved documents over all relevant documents in the corpus. So,

\[
Recall_A = \frac{7}{28} = 25 
\]

\[
Recall_B = \frac{5}{28} = 17.8 
\]

Accuracy is the fraction of the sum of relevant retrieved documents and non-relevant unretrieved documents over the entire corpus. So,

\[
Accuracy_A = \frac{7 + 969}{1000} = 97.6 
\]

\[
Accuracy_B = \frac{5 + 957}{1000} = 96.2 
\]
(b) Search engine A in both cases.

(c) In case of information retrieval problems, almost always, the data classes are skewed with the majority of documents being irrelevant. As in this example, out of a 1000 document corpus, only 28 are relevant. As a result, while a system may have high accuracy, it may be suffering from high false positives.

Considering the metrics of search engines A and B, we see that they have comparable accuracies. However, a user using system B gets more false positive results (15) than user A (3) which is not desirable in a good retrieval system. Comparing the recall and precision scores we see that A is a far better performer than B. Recall and precision scores evaluate important aspects of IR - how many relevant documents could be retrieved and what percentage was false positives.

(d) Here we see that the top five results in A have 3 relevant documents whereas B returns the top 5 relevant documents in almost the same order of relevance. Hence, intuitively, B performs better. (Later we will learn about metrics like precision@k to evaluate systems more strictly.)

**Problem 2** (Et tu, Brute?). Postings or inverted list increase the efficiency since instead of scanning entire documents word for word, we look up for the query terms and return the linked list of document IDs that contain the term in constant time. Assume that you have created an inverted index on the Shakespeare’s Collected Works. The terms Brutus, Ceasar and Calpurnia have posting lists of sizes $x$, $y$ and $z$, respectively.

a What is the time complexity for the querying the keywords "Brutus" AND "Calpurnia"? When is this achieved?

b What is the time complexity for querying "Brutus" OR "Calpurnia"?

c Make the following queries on Google (keeping uppercases intact) and report the number of estimated search results.

(a) Caesar
(b) Brutus
(c) Caesar AND Brutus
(d) Caesar OR Brutus
(e) Brutus AND Caesar

(i) Do the numbers in the first four queries follow Boolean logic? Briefly justify.

(ii) Do the numbers in the third and fifth query follow Boolean logic? Briefly discuss, with an example, why word order should affect search results.

**Solution.**

a The time complexity for querying is linear in the size of the postings list of both keywords, i.e., $O(x + z)$. It is achieved only when the postings list is arranged in increasing order on the document IDs. As a result, the intersection of the two posting lists (the desired result for this case) is obtained linearly traversing each list. In practice, we can avoid complete traversal of the longer list if we reach a document ID in the longer list greater than or equal to the last document ID in the shorter list.

b Here the time complexity is a tight bound over the length of the postings lists, $\Theta(x + z)$. Since, unlike AND queries, we always have to traverse the entire list.

c The order of the estimated search results are:
a Caesar: $598 \times 10^6$

b Brutus: $29 \times 10^6$

c Caesar AND Brutus: $3.6 \times 10^6$

d Caesar OR Brutus: $548 \times 10^6$

e Brutus AND Caesar: $3.8 \times 10^6$

(i) The number of search results on the conjunction query (Caesar AND Brutus) does satisfy boolean logic: the number of documents containing both terms cannot be more than the minimum of the document frequencies of the search keywords.

The number of search results on the disjunction query (Caesar OR Brutus) does not satisfy Boolean logic. The number of relevant documents is at least equal to the maximum of the document frequencies of the search keywords and upper bounded by the sum of the document frequencies of the search keywords.

Here document frequency is another term for the length of postings list of a keyword.

(ii) Both queries individually follow Boolean logic (the results of the AND queries are less than the single term query results). Also, following Boolean logic, order of conjunctions should not affect the final result since AND operation is commutative. However we get more results in (c) an order of $10^5$ than in (e).

Though not apparent in this query, word order can change the meaning of the search queries. Consider the two search queries sky blue and blue sky where the user might have different intents. While the first refers to a color shade in general, the second query refers to the color of the sky.

**Problem 3** (To grep or not to grep). The season of Triwizard Tournament is back and there is a stir in both Wizarding and Muggle worlds. With the challenge of the unprecedented tasks impending on the Champions, they need all hands on deck. As a Muggle powered by the knowledge of IR, you are hired as a consultant to the Hogwarts School to help its champions complete their tasks.

Consider the following users and their queries. Your job is to choose a method of retrieval from the ones taught in the lecture (grep, boolean, tf-idf, ..), to be used for finding relevant documents. Give one line justifying your choice. Assume that all data exists only as text documents. Any component of the IR pipeline necessary for the final retrieval must be mentioned.

(a) **User:** Harry Potter (a competing champion in the Triwizard tournament)

   **Background:** Harry wants to know all instances of deaths that might have occurred in the previous tournaments. As a champion he has access to the documents recording the events of past tournaments by year and Harry would want to retrieve all documents containing the the following keywords.

   **Query:** "Triwizard Cup" and "death"

(b) **User:** Hermione Granger (A bright witch who is also Harry’s best friend and supporter)

   **Background:** Hermione knows that the first task involves dragons and wants to read up on all charms and curses that can be used on dragons. The Hogwarts library has millions of documents and thousands on dragons alone. Scanning each document serially would take days. She has but a few hours and some important keywords.

   **Query:** "charm", "curse", "dragon"

(c) **User:** Ronald Weasly (Harry’s best friend)

   **Background:** Ron finds out the four dragon species that have been brought in for the first task. He knows that the library has an index of the dragon species and wants documents with exact mentions of any of the dragon species.

   **Query:** "Hungarian Horntail" or "Swedish Short-Snout" or "Welsh Green" or "Chinese Fireball"
User: Albus Dumbledore (headmaster of Hogwarts)

Background: Professor Dumbledore suspects foul play by organizers, since Harry being an underage wizard should never have been selected as a Champion. He wants to look into any past instances of involvement of organizers in manipulating results in the tournament.

Query: "Rigging the Goblet of fire in Triwizard tournament"

Solution.

(a) Both grepping and boolean retrieval make sense here. Since the record collection is limited to tournament events it must not be very huge and at most one for each year.

If boolean retrieval is used the required pipeline components include a pre-processing stage to compute an incidence matrix (each row is a document vector of the length of the dictionary and each column is 0/1 depending on whether the dictionary word is present in the document) and query processing to convert a query to a vector of the length of the dictionary, where only the columns representing query word is set and the rest are 0.

Using keyword search on posting lists is also possible.

(b) Using tf-idf document and query vectors and ranking documents with respect to some similarity metrics (cosine similarity) will yield more desirable results.

Here again, we need pre-processing on the documents (to create a dictionary and tf-idf vectors of the documents), query processing to a tf-idf vector and an optional ranking on closely matching documents.

(c) A keyword search on postings list is sufficient in this case.

Since there already exists inverted index of keywords, the document pre-processing step can be used to ensure that the lists are sorted on the document IDs for more efficient linear time search.

(d) A vector space model using tf-idf or its variations is ideal for this kind of query. The documents can be ranked on similarity scores (similarity to the query).

The required pipeline steps include pre-processing steps for creating a dictionary and document vector, query processing to vectors and ranking.