

Bronco ID: |0|1|7|3|2|4|7|6|2|

Last Name: Lebid

First Name: Yurii

Question 1: [10 points – 5.0 points each]. Despite recent advances in the field, the primary focus of Information Retrieval remains on text documents. Based on this observation, answer the following questions:

(a) Why is querying a relational database table easier than querying text documents?

1. Querying relational databases is easier mainly because the data in the database is represented in a structured form with well-defined attributes (fields) and formats. This allows for exact matching and evaluation of predicates, such as SQL queries for numeric balances. Text files, on the other hand, are mostly unstructured and context-driven; "matching" requires equivalent and comparative language, where the same message can be conveyed in different ways, and exact word assignment is often inadequate, and so information retrieval focuses on ranking and relevance rather than equivalence.
2. Another significant problem is - vocabulary mismatch. When people search for information, they usually enter a short and vague query, but what they really want (their entire information need) is much larger and more specific. This is a mismatch between what they mean and what they really do. For instance:

"car" vs "automobile", "cheap" vs. "affordable"

What constitutes as a "good match" in text searching is much harder to determine than in databases. In a database, fields are well-defined, and because of that, the search is exact and structured. In text, meaning is flexible, ambiguous, and conveyed in different ways.

(b) Explain how text has traditionally been used by Information Retrieval researchers to represent and compare multimedia documents, and how this scenario is currently evolving.

Back in time, multimedia components such as images, videos, and audio were normally compared and represented using text-based metadata, which includes captions, tags, context, and file names. This was mainly due to the complexity involved in comparing the original content of the multimedia files. However, with deep learning algorithms, it is now possible to compare multimedia files directly by comparing images based on learned representations or embeddings.

Question 2: Explain the scope of each of the following search application types and give one practical example for each.

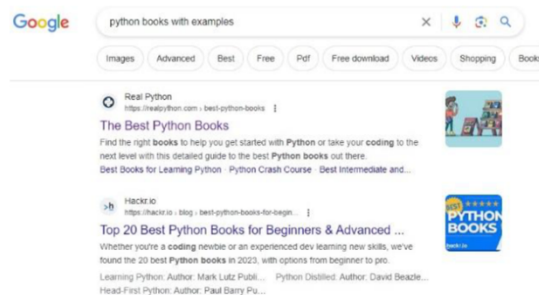
- a. Web search engine.
- b. Vertical search engine.
- c. Enterprise search engine.
- d. Desktop search engine.

Search type	Scope of search	Access type	Example of usage
Web search engine	Broad public web pages and web documents of any different kind of information	Public	Google (searching for news)
Vertical search engine	A specialized subset of content such as scholarly papers, images, jobs, products	Usually public (can be mixed)	Google Scholar (looking for scholarly verticals), ResearchGate (finding science publications)
Enterprise search engine	An organization's internal/private repositories (documentations, contacts information, shared datasets)	Private	Internal company search over SharePoint/Confluence (looking for documentation about specific project)
Desktop search engine	A single user's local machine content (files, emails, metadata, sometimes indexed app content)	Private on personal device	Windows Search or MacOS Spotlight (searching for documents on local driver)

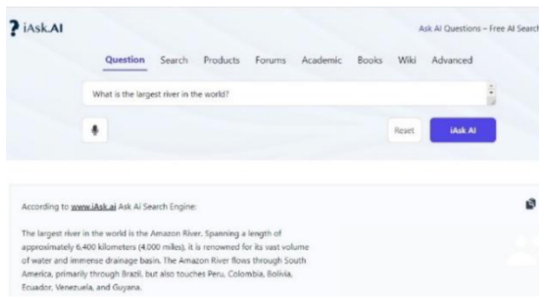
Question 3: Identify and explain the Information Retrieval tasks illustrated below.



(a) Assigning tags: house, grass, outdoor”
– means that it is classification (assigning labels to an item/document).



(b) Typical search-engine results page for a typed query, which is ad-hoc search (ranked retrieval of documents for an arbitrary text query).



(c) Question-style input (“What is the largest star in the world?”) intended to return a direct response, which is question answering.



(d) News feed (or Notifications) of recommended stories. This is filtering (selecting/pushing items likely relevant to a user/profile from a stream of new documents, as a personalized feed).

Question 4: Considering the results of a given IR system, use the interpretations - TP

(True Positive): relevant retrieved, FP (False Positive): irrelevant retrieved, TN (True Negative): irrelevant not retrieved, and FN (False Negative): relevant not retrieved, to classify the situations below.

- a) You query ChatGPT “Mission to Moon”, and the machine answers “Apollo 8 succeed”.
- b) You query IBM Watson “PhD in Computer Science”, and the machine does not include “Cal Poly Pomona”.
- c) You query IBM Watson “Elections in France”, and the machine answers “Trump and Biden dominate”.
- d) You query ChatGPT “Theme parks in CA”, and the machine does not include “Disneyland”

(a) “Mission to Moon” returns “Apollo 8 succeed”

Retrieved/Returned?	↓	Yes
Relevant? (assumption)	↓	Yes (It is a moon mission which succeeded)
Classification:		TP

(b) “PhD in Computer Science” does **not** include “Cal Poly Pomona”

Retrieved/Returned?	↓	No
Relevant? (assumption)	↓	No (CPP is not relevant to that query)
Classification:		TN

(c) “Elections in France” returns “Trump and Biden dominate”

Retrieved/Returned?	↓	Yes
Relevant? (assumption)	↓	No
Classification:		FP

(d) “Theme parks in CA” does **not** include “Disneyland”

Retrieved/Returned?	↓	No
Relevant? (assumption)	↓	Yes
Classification:		FN

Question 5: Draw the inverted index that would be built for the following document collection.

Requirement: apply surface level normalization and lemmatization to generate terms.

Doc1: New homes sale increases. Doc2: Home sale rising in July.

Doc3: Increasing home sales in july. Doc4: July new home sales rise

Normalized words + lemmatized terms per document:

Doc1: New homes sale increases.

{new, home, sale, increase}

Doc2: Home sale rising in July.

{home, sale, rise, july}

Doc3: Increasing home sales in july.

{increase, home, sale, in, july}

Doc4: July new home sales rise.

{july, new, home, sale, rise}

Inverted index (term goes to postings list with term frequency tf):

Term	Postings (doc:tf)
home	(1:1), (2:1), (3:1), (4:1)
increase	(1:1), (3:1)
july	(2:1), (3:1), (4:1)
new	(1:1), (4:1)
rise	(2:1), (4:1)
sale	(1:1), (2:1), (3:1), (4:1)

Question 6: Draw the document–term matrix using binary representation for the corpus below.

Requirement: no text processing is to be performed; order terms alphabetically.

Doc1: breakthrough drug for schizophrenia Doc2: new schizophrenia drug

Doc3: new approach for treatment of schizophrenia Doc4: new hopes for schizophrenia patients

Alphabetical vocabulary (unique tokens):

approach, breakthrough, drug, for, hopes, new, of, patients, schizophrenia, treatment

Binary document–term matrix:

Doc.	approach	breakthrough	drug	for	hopes	new	of	patients	schizophrenia	treatment
Doc1	0	1	1	1	0	0	0	0	1	0
Doc2	0	0	1	0	0	1	0	0	1	0
Doc3	1	0	0	1	0	1	1	0	1	1
Doc3	0	0	0	1	1	1	0	1	1	0

Question 7: Score of the documents below with respect to the query q = “I love dogs”.

Requirements: perform tokenization, surface-level normalization, stopping (pronouns, conjunctions, and articles), and stemming; construct the term vocabulary using word unigrams followed by word bigrams, with terms ordered alphabetically within each group; represent queries and documents using binary (0/1) term weights; rank the documents according to their scores.

$\text{Score}(d) = \sum q_i \cdot d_i$, where q_i and d_i are query and document term weights for term i .

Documents:

d_1 = “I love a dog and a cat.”

d_2 = “She loves her cat and dogs.”

d_3 = “They love their cat.”

Pipeline result after normalization + stopping pronouns/articles/conjunctions + Porter-style stemming):

$d_1 \rightarrow$ tokens: [love, dog, cat]

$d_2 \rightarrow$ tokens: [love, cat, dog]

$d_3 \rightarrow$ tokens: [love, cat]

q = “I love dogs” \rightarrow tokens: [love, dog]

Grouped vocabulary construction:

Unigrams (alphabetical): {cat,dog,love}

Bigrams: {cat dog,dog cat,love cat,love dog}

Binary vectors over [cat,dog,love | cat dog,dog cat,love cat,love dog]:

Query $q = [0,1,1 \mid 0,0,0,1]$

$d_1 = [1,1,1 \mid 0,1,0,1]$

$d_2 = [1,1,1 \mid 1,0,1,0]$

$d_3 = [1,0,1 \mid 0,0,1,0]$

Scores for dot product $\sum_i q_i d_i$:

Score for D1: 3, Score for D2: 2, Score for D3: 1

Ranking highest score first: D1 > D2 > D3

Question 8: Complete the Python program (search_engine.py) that will programmatically
Solve Question 7. Add the link to an online repository as the answer to this question.

Link to Github: <https://github.com/yuriilebid/CS5180.git>

```
Original documents:
d1: I love a dog and a cat.
d2: She loves her cat and dogs.
d3: They love their cat.

Vocabulary: ['cat', 'cat dog', 'dog', 'dog cat', 'love', 'love cat', 'love dog']

Scores:
d1: 3
d2: 2
d3: 1

Ranking (doc_id, score, document):
d1      3      I love a dog and a cat.
d2      2      She loves her cat and dogs.
d3      1      They love their cat.
```