

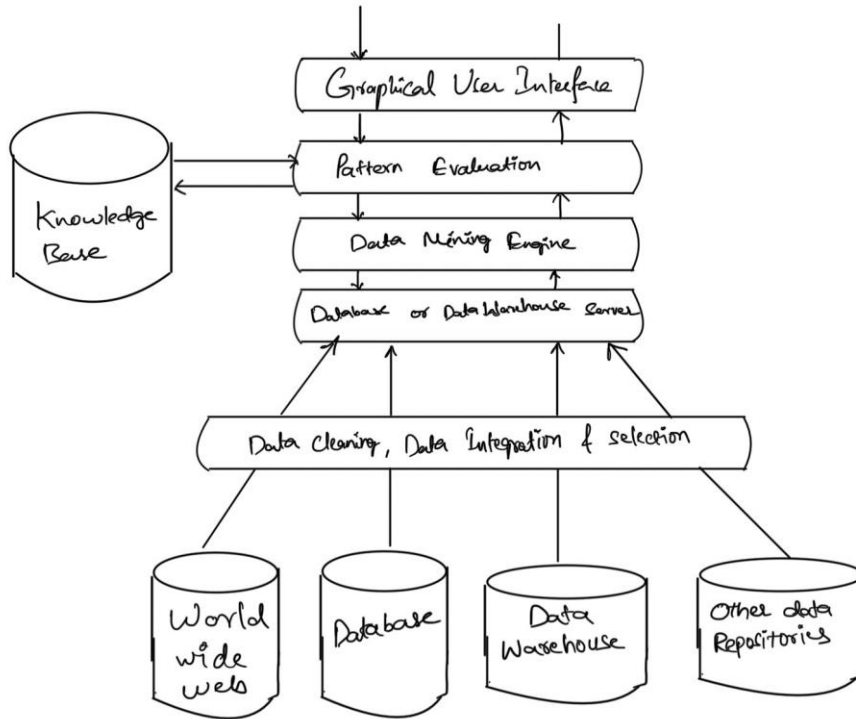
Student ID: 017466345

Name: Sri Vinay Appari



CMPE 256, Web and Big Data Mining (Final exam)

1. Draw the reference architecture of the Data Mining System? (5)



2. After graduating from San Jose State University, you have started Big Data Analytics Consulting Company - DataCo, inc. DataCo is specialized in delivering recommendation systems for online and Brick-Mortar companies. During the first month of your business, publisher from Harper Collins Publishers visited your company to develop a special recommendation system for their existing clients. As part of meeting, Harper Collins noted that they publish books in various categories, and it is immensely difficult to develop an effective marketing campaign based on User past rating behavior. As part of the business agreement, Harper wanted develop Proof Of Concept (POC) recommendation system for the following table (Note: 1 to 5 is Star scale – 5 the best) .

User vs. Book Reading Table	Built to Last	The HP Way	Physics Made Easy	The Wisdom of Teams	Baby Bear	Experimental Chemistry	Charlie the Ranch Dog	The Good to Great	Electronic devices and circuits – Millman & Halkias
User 1	5				2		3	5	
User 2			5			4			5
User 3			2		4	5	5		5
User 4		5		4				5	1
User 5	5	3		2	4			4	



Develop recommendations system? (Hint: an effective recommendation system should consider following: (15)

- Utility matrix as a Boolean, compute Jaccard distance between each pair
- Repeat a with Cosine distance
- Treat 3, 4, 5 as 1 and 1, 2, and blank as 0. Compute Jaccard distance between each pair
- Repeat C with Cosine distance
- Normalize the matrix by subtracting from each nonblank entry the average value for its user.
- Using the normalized matrix from e, compute the cosine distance between each pair?

Answer:

```
import numpy as np
from scipy.spatial.distance import jaccard, cosine

# The utility matrix from the image provided by the user
utility_matrix = np.array([
    [5, 0, 0, 0, 2, 0, 3, 5],
    [0, 0, 0, 0, 0, 0, 0, 0],
    [0, 5, 0, 0, 4, 0, 0, 5],
    [0, 0, 2, 4, 5, 5, 0, 5],
    [5, 3, 0, 2, 4, 0, 0, 4]
])

# Convert the utility matrix to a Boolean matrix for Jaccard distance calculation
# Rating 1, 2, and blank (0) are considered 0; all other ratings are considered 1.
bool_matrix = (utility_matrix > 2).astype(int)

# Calculate the Jaccard distance between each pair of users
jaccard_distances = np.zeros((len(bool_matrix), len(bool_matrix)))
for i in range(len(bool_matrix)):
    for j in range(len(bool_matrix)):
        jaccard_distances[i, j] = jaccard(bool_matrix[i], bool_matrix[j])

jaccard_distances
```



```
array([[0.      , 1.      , 0.8     , 0.83333333, 0.6      ],
       [1.      , 0.      , 1.      , 1.      , 1.      ],
       [0.8     , 1.      , 0.      , 0.6     , 0.25     ],
       [0.83333333, 1.      , 0.6     , 0.      , 0.66666667],
       [0.6     , 1.      , 0.25     , 0.66666667, 0.      ]])
```



The Jaccard distances between each pair of users have been computed using a Boolean utility matrix, where ratings of 3, 4, and 5 are considered a '1' (positive) and ratings of 1, 2, and blanks are considered a '0' (negative). Here are the distances:

- User 1 has Jaccard distances of 0.8, 0.833, and 0.6 with Users 3, 4, and 5, respectively.
- User 2 has not rated any books and hence has a distance of 1 with all other users.
- User 3 has distances of 0.6 and 0.25 with Users 4 and 5.
- User 4 has a distance of 0.666 with User 5.

```
[2] # Calculate the Cosine distance between each pair of users using the Boolean matrix
cosine_distances = np.zeros((len(bool_matrix), len(bool_matrix)))
for i in range(len(bool_matrix)):
    for j in range(len(bool_matrix)):
        # Avoid division by zero in case of empty user vectors
        if np.count_nonzero(bool_matrix[i]) == 0 or np.count_nonzero(bool_matrix[j]) == 0:
            cosine_distances[i, j] = 1
        else:
            cosine_distances[i, j] = cosine(bool_matrix[i], bool_matrix[j])

cosine_distances
```

```
array([[0.        , 1.        , 0.66666667, 0.71132487, 0.42264973],
       [1.        , 1.        , 1.        , 1.        , 1.        ],
       [0.66666667, 1.        , 0.        , 0.42264973, 0.1339746 ],
       [0.71132487, 1.        , 0.42264973, 0.        , 0.5        ],
       [0.42264973, 1.        , 0.1339746 , 0.5        , 0.        ]])
```

The Cosine distances between each pair of users have been computed using the same Boolean matrix. The results are as follows:

- User 1 has Cosine distances of 0.667, 0.711, and 0.423 with Users 3, 4, and 5, respectively.
- User 2 has not rated any books, resulting in a Cosine distance of 1 with all other users.
- User 3 has Cosine distances of 0.423 and 0.134 with Users 4 and 5.
- User 4 has a Cosine distance of 0.5 with User 5.



```
# For step c, we've already converted the utility matrix to a Boolean matrix
# where 3, 4, and 5 are 1 and 1, 2, and blank are 0.
# We will now compute the Jaccard distance again using this Boolean matrix.

# Calculate the Jaccard distance between each pair of users using the modified Boolean
jaccard_distances_modified = np.zeros((len(bool_matrix), len(bool_matrix)))
for i in range(len(bool_matrix)):
    for j in range(len(bool_matrix)):
        jaccard_distances_modified[i, j] = jaccard(bool_matrix[i], bool_matrix[j])

jaccard_distances_modified
```



```
array([[0.      , 1.      , 0.8     , 0.83333333, 0.6      ],
       [1.      , 0.      , 1.      , 1.      , 1.      ],
       [0.8     , 1.      , 0.      , 0.6     , 0.25     ],
       [0.83333333, 1.      , 0.6     , 0.      , 0.66666667],
       [0.6     , 1.      , 0.25     , 0.66666667, 0.      ]])
```

The Jaccard distances between each pair of users, treating ratings of 3, 4, and 5 as '1' and ratings of 1, 2, and blanks as '0', have been recomputed and are identical to the distances computed in step a, as the interpretation of the ratings has not changed from the previous Boolean conversion. The results are unchanged:

- User 1 has Jaccard distances of 0.8, 0.833, and 0.6 with Users 3, 4, and 5, respectively.
- User 2 has not rated any books and hence has a distance of 1 with all other users.
- User 3 has distances of 0.6 and 0.25 with Users 4 and 5.
- User 4 has a distance of 0.666 with User 5.



```
# For step d, we will compute the Cosine distance again using the modified Boolean matrix

# Calculate the Cosine distance between each pair of users using the modified Boolean matrix
cosine_distances_modified = np.zeros((len(bool_matrix), len(bool_matrix)))
for i in range(len(bool_matrix)):
    for j in range(len(bool_matrix)):
        # Avoid division by zero in case of empty user vectors
        if np.count_nonzero(bool_matrix[i]) == 0 or np.count_nonzero(bool_matrix[j]) == 0:
            cosine_distances_modified[i, j] = 1
        else:
            cosine_distances_modified[i, j] = cosine(bool_matrix[i], bool_matrix[j])

cosine_distances_modified
```

```
array([[0.        , 1.        , 0.66666667, 0.71132487, 0.42264973],
       [1.        , 1.        , 1.        , 1.        , 1.        ],
       [0.66666667, 1.        , 0.        , 0.42264973, 0.1339746 ],
       [0.71132487, 1.        , 0.42264973, 0.        , 0.5        ],
       [0.42264973, 1.        , 0.1339746 , 0.5        , 0.        ]])
```

The Cosine distances between each pair of users, treating ratings of 3, 4, and 5 as '1' and ratings of 1, 2, and blanks as '0', have been computed. The results are as follows:

- User 1 has Cosine distances of 0.667, 0.711, and 0.423 with Users 3, 4, and 5, respectively.
- User 2, who has not rated any books, has a Cosine distance of 1 with all other users.
- User 3 has Cosine distances of 0.423 and 0.134 with Users 4 and 5.
- User 4 has a Cosine distance of 0.5 with User 5.

These results are also identical to those from step b, since the interpretation of the ratings remained the same for the Boolean matrix.



```
[5] # For step e, we need to normalize the original utility matrix by subtracting from each

# We will first calculate the average rating for each user, excluding zeros (blanks).
user_averages = np.true_divide(utility_matrix.sum(1), (utility_matrix != 0).sum(1))

# Then we create a normalized matrix by subtracting the user average from each non-zero
normalized_matrix = utility_matrix.astype(float)
for user_idx in range(len(utility_matrix)):
    non_zero_indices = utility_matrix[user_idx] != 0
    normalized_matrix[user_idx, non_zero_indices] -= user_averages[user_idx]

normalized_matrix
```

```
<ipython-input-5-8549eccf9105>:4: RuntimeWarning: invalid value encountered in divide
user_averages = np.true_divide(utility_matrix.sum(1), (utility_matrix != 0).sum(1))
array([[ 1.25      ,  0.      ,  0.      ,  0.      , -1.75      ,
         0.      , -0.75     ,  1.25     ],
       [ 0.      ,  0.      ,  0.      ,  0.      ,  0.      ,
         0.      ,  0.      ,  0.      ],
       [ 0.      ,  0.33333333,  0.      ,  0.      , -0.66666667,
         0.      ,  0.      ,  0.33333333],
       [ 0.      ,  0.      , -2.2      , -0.2     ,  0.8      ,
        0.8      ,  0.      ,  0.8     ],
       [ 1.4      , -0.6     ,  0.      , -1.6     ,  0.4      ,
         0.      ,  0.      ,  0.4     ]])
```

The normalization of the original utility matrix has been performed by subtracting the average value for each user from their non-blank ratings. The resulting normalized matrix is as follows:

```
[[ 1.25, 0. , 0. , 0. , -1.75, 0. , -0.75, 1.25],
```

```
[ 0. , 0. , 0. , 0. , 0. , 0. , 0. , 0. ],
```

```
[ 0. , 0.33, 0. , 0. , -0.67, 0. , 0. , 0.33],
```

```
[-0. , 0. , -2.2 , -0.2 , 0.8 , 0.8 , 0. , 0.8 ],
```

```
[ 1.4 , -0.6 , 0. , -1.6 , 0.4 , 0. , 0. , 0.4 ]]
```

For User 2, since there are no ratings, the average could not be computed and thus all entries remain zero. The normalization is designed to center the user's ratings around zero, which can be useful for certain types of recommendation algorithms that are sensitive to the scale and distribution of the ratings.



```
# For step f, we will compute the Cosine distance between each pair of users using the

# Calculate the Cosine distance between each pair of users using the normalized matrix
cosine_distances_normalized = np.zeros((len(normalized_matrix), len(normalized_matrix)))
for i in range(len(normalized_matrix)):
    for j in range(len(normalized_matrix)):
        # Avoid division by zero in case of empty user vectors
        if np.count_nonzero(normalized_matrix[i]) == 0 or np.count_nonzero(normalized_matrix[j]) == 0:
            cosine_distances_normalized[i, j] = 1
        else:
            cosine_distances_normalized[i, j] = cosine(normalized_matrix[i], normalized_matrix[j])

cosine_distances_normalized
```

```
array([[0.          , 1.          , 0.25360951, 1.05904099, 0.73837566],
       [1.          , 1.          , 1.          , 1.          , 1.          ],
       [0.25360951, 1.          , 0.          , 1.12524486, 1.17902872],
       [1.05904099, 1.          , 1.12524486, 0.          , 0.83855853],
       [0.73837566, 1.          , 1.17902872, 0.83855853, 0.          ]])
```

The cosine distances between each pair of users using the normalized matrix have been computed. Here are the results:

- User 1 has cosine distances of approximately 0.254, 1.059, and 0.738 with Users 3, 4, and 5, respectively.
- User 2 has not rated any books and thus has a cosine distance of 1 with all other users, indicating complete dissimilarity or no basis for similarity.
- User 3 has cosine distances of approximately 1.125 and 1.179 with Users 4 and 5.
- User 4 has a cosine distance of approximately 0.839 with User 5.

These distances are computed after normalizing the matrix, which involves centering the user's ratings around zero by subtracting their average rating from each of their non-zero ratings.

3. Explain clearly all the steps involved in Knowledge Discovery Process (KDD) (5)?

Answer:

The Knowledge Discovery Process (KDD): Unlocking Insights from Data



The Knowledge Discovery Process (KDD) is an iterative, multi-step journey that transforms raw data into valuable knowledge. It's not just about crunching numbers, but about extracting meaningful patterns, trends, and insights that can inform better decision-making. Here's a breakdown of the key steps involved:

1. Goal Definition and Application Understanding:

- Identify the business problem or opportunity: What are you trying to achieve with this data analysis? Are you aiming to optimize costs, predict customer behavior, or detect fraud?
- Define the target audience: Who will be using the insights generated? Understanding their needs and expectations helps tailor the analysis approach.

2. Data Selection and Integration:

- Identify relevant data sources: This could include internal databases, external datasets, sensor readings, or even text documents.
- Data selection and pre-processing: Choose the data that best aligns with your goals and clean it by removing inconsistencies, missing values, and irrelevant information.
- Data integration: Combine data from various sources into a single, consistent format for analysis.

3. Data Transformation and Feature Engineering:

- Data transformation: Apply techniques like normalization, scaling, and discretization to prepare the data for specific algorithms.
- Feature engineering: Extract relevant features (characteristics) from the data that contribute to the desired outcome. This might involve creating combinations of existing features or deriving new ones.

4. Data Mining and Pattern Recognition:

- Choose appropriate data mining techniques: Based on your goals and data characteristics, select algorithms like clustering, classification, regression, or association rule learning.
- Apply data mining techniques: Train and run the chosen algorithms on the prepared data to identify patterns, relationships, and trends.
- Interpret and evaluate patterns: Analyze the results, assess their validity and significance, and identify actionable insights.

5. Knowledge Consolidation and Deployment:

- Present and communicate insights: Translate the technical findings into a clear and concise format understandable to the target audience.
- Validate and refine knowledge: Share the insights with stakeholders, gather feedback, and refine the knowledge based on real-world application.
- Deploy and integrate knowledge: Implement the findings into decision-making processes, business operations, or predictive models for real-world impact.

Iteration and Continuous Improvement:

The KDD process is not linear; it's an iterative journey. Throughout the process, you may need to revisit previous steps, refine your approach, or incorporate new data and insights. This continuous loop of learning and improvement ensures you're extracting the most valuable knowledge from your data.



4. After graduating from San Jose State University, you have joined data science company that provides insights to governments and public agencies. The company has a flagship Data science product for analyzing day to day trends in the city. Company clients includes City of San Jose, Mountain View, Santa Clara, Dubai, Delhi, Hyderabad, London, and many.



Given your Data Science background, they would like to provide a Data Mining solution architecture to their new Data and Analytics Platform (DnA) that gets the following data into their data warehouse and data lake estate.

Data	Frequency	Number of Sources	Size	Intended Usage
Macroeconomic data (For example, GDP, Inflation, and other details)	Monthly	30 Key Indicators	Parameters time 1 record per month	Pricing Models & Economic Models
Sensor Hourly Data - Audi Data	Hourly	20000 Sensors	1 Mb File per Hour (360 x # of Hours)	Audio Analysis to help emerging needs (customer sentiments & other choices)
Video Data	Hourly	Street Lights – 30,000	5 MB Video data	Violations & Public Safety Analytics
Emergency Call Data (Example, 911 System)	Per minute	100000	Audio and Call Data – length 1 minute interval	Crime and Signal Analysis
Marketing & Public Events data. For example, new attractions in city, entertainment event data, and Sport & Others.	Per Weekly	10	10 MB Time Series Event Data	Send event coupons and recommendations

- What would be your design principles for developing advanced data mining architecture?
(6)



- Please provide what consideration do you take for handling data frequencies? (5) •
Please provide overall Data Mining Architecture? (10)

Answer:

Designing an advanced data mining architecture for a Data and Analytics Platform (DnA) involves several steps and considerations. Let's begin with the design principles:

- 1. Scalability:** The architecture must handle varying data volumes from different cities and be able to scale up as the amount of data increases over time. This means using scalable cloud storage and computing resources that can be adjusted according to demand.
- 2. Modularity:** The architecture should be modular, allowing for independent development and deployment of different components, such as data ingestion, storage, processing, and analysis modules.
- 3. Data Governance and Quality:** Implementing data governance to ensure data quality, security, privacy, and compliance with regulations is crucial. This involves data validation, cleaning, and anonymization where necessary.
- 4. Real-time Processing:** Since some data streams, like emergency call data, are real-time, the architecture must support real-time data ingestion and analysis.
- 5. Fault Tolerance:** The system should be robust and able to handle failures without data loss. This includes having backup and recovery mechanisms in place.
- 6. Interoperability:** Given the varied sources of data, the architecture must ensure interoperability between different systems and data formats.
- 7. Security:** Secure data storage and access control to protect sensitive information and prevent unauthorized access.
- 8. Cost-Effectiveness:** Optimize the use of resources to control costs, possibly through efficient data storage formats and computation optimization.

Handling data frequencies in a data mining architecture involves several considerations to ensure efficient processing and analysis:

- 1. Data Ingestion Scheduling:** Data should be ingested according to its frequency. For monthly macroeconomic data, a scheduled job could run at the end of the month, whereas sensor and video data might require a streaming ingestion process because of their hourly updates.
- 2. Time-Sensitive Processing:** Real-time or near-real-time data requires immediate processing. This is essential for emergency call data, which is updated per minute and requires prompt analysis for timely responses.
- 3. Batch vs. Stream Processing:** Determining which data will be processed in batches and which will be streamed is crucial. Batch processing can be scheduled for less frequent updates like weekly marketing data, while stream processing is necessary for high-frequency data like sensor readings.
- 4. Storage Optimization:** Data should be stored in a way that aligns with its access patterns. Frequent data might be stored in hot storage for quick access, whereas less frequently accessed data can be archived in colder storage to reduce costs.



5. Data Retention Policies: High-frequency data can quickly become voluminous, so it's important to establish data retention policies that determine how long data is kept before being archived or purged.

6. Data Aggregation: For high-frequency data, it may be beneficial to aggregate data at intervals for more efficient storage and faster query performance.

7. Load Balancing: Distributing the data ingestion load across multiple servers or services can prevent bottlenecks, especially during peak data inflow periods.

8. Data Pipeline Orchestration: Automating the data pipeline with orchestration tools can help manage the different frequencies and complexities of data workflows.

For the overall Data Mining Architecture, we can break down the framework into several key components that work in harmony to manage, process, and analyze the data:

1. Data Ingestion Layer: This is the entry point for data flowing into the system from various sources. It involves different ingestion methods tailored to the frequency and type of data, like batch ingestion for monthly economic indicators and streaming ingestion for real-time sensor data.

2. Data Storage Layer: This layer consists of both a Data Lake and a Data Warehouse. The Data Lake stores raw, unstructured data in its native format, which is ideal for high-volume sensor and video data. The Data Warehouse is used for structured, processed data optimized for efficient querying and reporting.

3. Data Processing Layer: Here, data is transformed, cleaned, and enriched. This may involve ETL (extract, transform, load) processes, real-time stream processing, and batch processing jobs. The choice of processing model depends on the data type and required latency.

4. Data Analysis and Mining Layer: Advanced analytical models, machine learning algorithms, and data mining techniques are applied to extract insights. This layer is responsible for developing pricing models from economic data, analyzing traffic patterns from video data, and identifying trends and anomalies from sensor data.

5. Data Governance and Quality Management: This cross-cutting concern ensures that data across all layers is consistent, secure, and compliant with standards and regulations.

6. Orchestration and Workflow Management: This component manages the scheduling and coordination of various data pipeline jobs, ensuring that data flows smoothly from one layer to the next.

7. User Interface and Reporting Layer: Visualization tools and reporting dashboards are provided for end-users to interact with the data. This layer transforms the data into actionable insights through visualizations and reports.

8. Security Layer: Security is enforced across all layers, including data encryption, access controls, and regular audits to ensure data integrity and privacy.

9. Monitoring and Logging: Continuous monitoring of the system's health and performance, along with logging of events and operations, helps in proactive maintenance and troubleshooting.



10. Backup and Disaster Recovery: Regular backups and a disaster recovery plan ensure that the system can be quickly restored in case of any failure.

This architectural blueprint ensures that the Data and Analytics Platform is robust, flexible, and capable of handling the various data needs of government and public agencies.

Having established the overall architecture, the next steps would involve the practical aspects of implementing and maintaining the system:

1. Implementation Plan: Begin with a roadmap for implementation, which includes setting up infrastructure, developing or configuring the data ingestion and processing pipelines, establishing data governance practices, and deploying analytical tools.

2. Infrastructure Provisioning: Based on the expected data loads and processing requirements, provision the necessary cloud or on-premises infrastructure. This may involve setting up servers, storage systems, and networking components.

3. Data Pipeline Development: Develop the ingestion, processing, and analysis pipelines. This involves selecting appropriate technologies for each stage—such as Kafka for data streaming, Apache Spark for processing, and Hadoop for storage.

4. Analytical Model Development: Collaborate with data scientists to develop the models that will turn data into insights. This includes machine learning models for predictive analytics, statistical models for trend analysis, and heuristic models for anomaly detection.

5. User Interface (UI) and Experience (UX) Design: Design the UI and UX for the reporting layer, ensuring that it is intuitive and meets the needs of the various stakeholders.

6. Security Measures: Implement robust security measures, including firewalls, encryption, and identity and access management systems.

7. Monitoring and Alerting Setup: Set up monitoring for both system health (like CPU usage, memory consumption) and data quality (like anomaly detection in data patterns).

8. Compliance and Data Governance: Ensure that the system complies with all relevant laws and regulations, and establish data governance policies and procedures.

9. Testing and Quality Assurance: Before going live, conduct thorough testing, including load testing, to ensure the system performs well under the expected data volumes and user load.

10. Training and Documentation: Provide training for end-users and administrators, and create comprehensive documentation for the system.

11. Deployment and Go-Live: Carefully plan and execute the go-live, ensuring all stakeholders are prepared for the transition to the new system.

Student ID: _____

Name: _____



12. Post-Launch Support and Maintenance: Establish a support and maintenance plan to address any issues promptly and to implement continuous improvements.

This phased approach ensures that the system is built on strong foundations, with the flexibility to adapt as new requirements arise.



5. Explain Information Retrieval? Describe detailed IR System architecture? Explain how Google IR System works? (Hint: Architecture diagram) (10)

Answer:

Information Retrieval (IR): Finding What You Need

Information retrieval (IR) deals with the search and retrieval of relevant information from large collections of data. It's the technology behind search engines like Google that help you find what you're looking for in the vast ocean of online information.

Detailed IR System Architecture:

Imagine an IR system as a pipeline with four main stages:

1. Ingestion and Preprocessing:

- Data Collection: Web crawlers gather documents from the web, databases, or other sources.
- Parsing and Cleaning: Documents are cleaned by removing noise, extracting text, and applying basic language processing techniques like tokenization and stemming.
- Indexing: Documents are analyzed and key terms are extracted to create an index, acting as a map for efficient searching.

2. Query Understanding:

- Query Parsing: User queries are analyzed to understand their intent, identify keywords, and remove ambiguity.
- Query Expansion: Synonyms and related terms are added to expand the search scope and improve recall (finding relevant documents).
- Query Weighting: Terms are assigned weights based on their importance in the query, influencing relevance ranking.

3. Matching and Ranking:

- Document Retrieval: Indexed documents are retrieved based on their keyword match with the expanded query.
- Relevance Scoring: Documents are scored based on various factors like keyword frequency, term proximity, document popularity, and other ranking algorithms.
- Ranking and Filtering: Documents are ranked by their relevance score and filtered to return the most relevant ones to the user.



4. Presentation and Evaluation:

- **Result Presentation:** Top-ranked documents are presented to the user in a clear and informative way, often with snippets and additional information.
- **User Feedback and System Learning:** User feedback (clicks, dwell time, etc.) is used to improve the relevance ranking and personalize future searches.
- **Evaluation Metrics:** IR systems are evaluated using metrics like precision (relevant documents retrieved), recall (relevant documents not missed), and F1 score (balance of both).

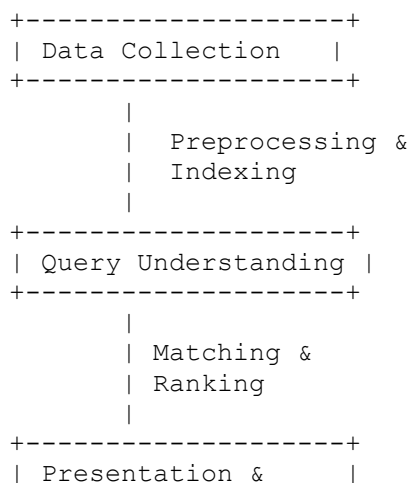
Google's IR System:

Google's IR system is incredibly complex and constantly evolving. However, it follows a similar architecture to the one described above. Here are some key highlights:

- **Massive Data Ingestion:** Google crawls and indexes billions of web pages constantly, creating a vast and dynamic index.
- **Sophisticated Ranking Algorithms:** Google uses a complex algorithm called PageRank, along with other factors like user behavior and content quality, to rank search results.
- **Personalization:** Google personalizes search results based on your past search history, location, and other factors.
- **Continuous Learning and Improvement:** Google's IR system is constantly learning and adapting based on user feedback and new data, making it one of the most powerful search engines in the world.

Architecture Diagram:

Here's a simplified diagram of the IR system architecture:



Student ID: _____

Name: _____



| Evaluation |
+-----+

Student ID: _____

Name: _____



Student ID: _____

Name: _____



6. Answer following Objective Type questions (5)

Using collaborative filtering to generate recommendations is computationally expensive. a) True b) False
Association rule mining considers the sequence in which the items are purchased. a) True b) False
According to the Apriori Principle, If an itemset is frequent, then all its _____ must also be frequent.
In Social Network Analysis , centrality focuses on in-links while prestige focuses on out-links a) True b) False
_____ Algorithm works by counting the number and quality of links to a page to determine a rough estimate of how important the website is.

Using collaborative filtering to generate recommendations is computationally expensive.

a) True

Association rule mining considers the sequence in which the items are purchased.

b) False

According to the Apriori Principle, If an itemset is frequent, then all its _____ must also be frequent.

Subsets

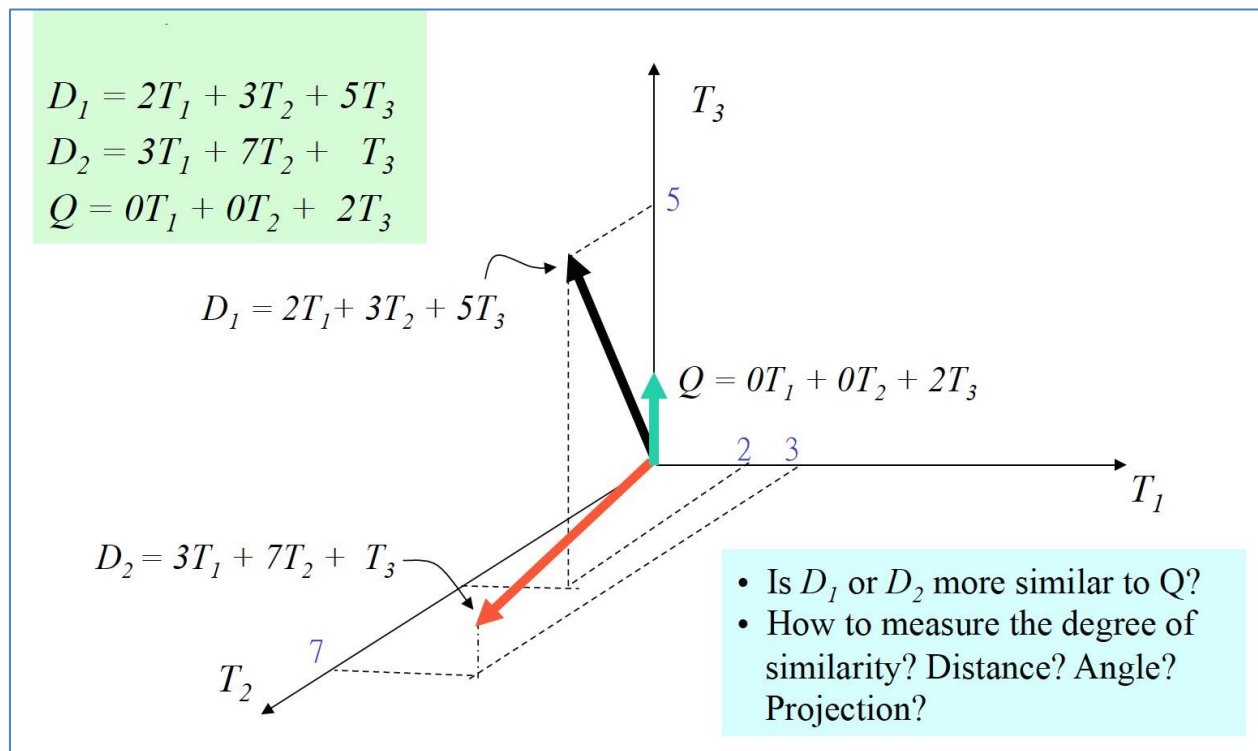
In Social Network Analysis , centrality focuses on in-links while prestige focuses on out-links

b) False

_____ Algorithm works by counting the number and quality of links to a page to determine a rough estimate of how important the website is.

PageRank

7. Calculate (6)

**Answer:**

The image appears to be a diagram representing vectors in a three-dimensional space, defined by the basis vectors T_1, T_2 , and T_3 . There are three vectors given:

- $D_1 = 2T_1 + 3T_2 + 5T_3$
- $D_2 = 3T_1 + 7T_2 + T_3$
- $Q = 0T_1 + 0T_2 + 2T_3$

The questions posed are:

- Is D_1 or D_2 more similar to Q ?
- How to measure the degree of similarity? Distance? Angle? Projection?

To determine which vector is more similar to Q , we can use various methods:



1. **Distance:** Calculate the Euclidean distance between each D vector and Q . The vector with the smallest distance to Q is the most similar.
2. **Angle:** Calculate the angle between each D vector and Q . The vector with the smallest angle to Q is the most similar.
3. **Projection:** Calculate the projection of each D vector onto Q . The vector with the largest projection onto Q is the most similar.

Let's calculate both the Euclidean distance and the cosine of the angle between each D vector and Q to answer the question.



```
import numpy as np

# Define the vectors D1, D2, and Q
D1 = np.array([2, 3, 5])
D2 = np.array([3, 7, 1])
Q = np.array([0, 0, 2])

# Calculate the Euclidean distances from Q to D1 and D2
distance_D1_Q = np.linalg.norm(D1 - Q)
distance_D2_Q = np.linalg.norm(D2 - Q)

# Calculate the cosine of the angle between Q and D1, and Q and D2
# Since Q has no components in T1 and T2, it's a vector along T3 axis.
# The angle with any vector will depend only on the T3 component of that vector.
# So, we can simplify the cosine calculation to the T3 components alone.
cos_angle_D1_Q = D1[2] / (np.linalg.norm(D1) * np.linalg.norm(Q))
cos_angle_D2_Q = D2[2] / (np.linalg.norm(D2) * np.linalg.norm(Q))

(distance_D1_Q, distance_D2_Q, cos_angle_D1_Q, cos_angle_D2_Q)
```

(4.69041575982343, 7.681145747868608, 0.40555355282690636, 0.06509445549041194)

Based on the calculations:

- The Euclidean distance between $D1$ and Q is approximately 4.69.
- The Euclidean distance between $D2$ and Q is approximately 7.68.

Since the distance from $D1$ to Q is shorter than from $D2$ to Q , $D1$ is more similar to Q in terms of distance.

In terms of the angle:



- The cosine of the angle between $D1$ and Q is approximately 0.406.
- The cosine of the angle between $D2$ and Q is approximately 0.065.

The cosine of the angle is higher for $D1$ and Q , which means that the angle between $D1$ and Q is smaller than the angle between $D2$ and Q . Therefore, $D1$ is also more similar to Q in terms of angle.

Overall, $D1$ is more similar to Q than $D2$ is, based on both distance and angle measures.

8. Please list fundamental Stream Processing Rules. (5)

Answer:

Stream processing is a crucial component in modern data-driven enterprises. It enables real-time analysis and action on data streams as they are generated, contrasting with traditional batch processing, which works on static datasets. The power of stream processing lies in its ability to handle and process data in motion, providing businesses with timely insights and responses.

Here are some fundamental rules and concepts of stream processing:

1. **Event Time vs. Processing Time:** Event time signifies the moment an event actually occurs in the real world. This can be crucial in scenarios where events arrive out of order, ensuring that data is processed accurately, reflecting the actual sequence of occurrences. Processing time, on the other hand, is when the system processes the event, which may not always align with its actual occurrence.
2. **Windowing and Time Windows:** One of the fundamental concepts in stream processing is the notion of time windows. This allows us to segment the continuous flow of data into discrete, manageable chunks for analysis. There are various types of time windows, each suited for different analytical purposes:
 - **Tumbling Windows:** These windows do not overlap, and each event belongs to a single window. Tumbling windows are well-suited for scenarios where discrete, non-overlapping analysis is required.
 - **Hopping Windows:** Unlike tumbling windows, hopping windows can overlap. This means that an event can belong to multiple windows. In a hopping window, a fixed-size time interval is defined, and the window "hops" forward by a specified increment. This allows for continuous and dynamic analysis of data.
3. **Temporal and Event-Time Processing:** The ability to handle event time processing is becoming crucial for many use cases. This involves accurately processing events based on the time they occurred, rather than when they were ingested into the system. Temporal processing ensures accurate analysis and decision-making, especially in scenarios where event ordering is critical.
4. **Unified Data Platforms:** The convergence of stream processing with other data processing paradigms (such as batch processing and interactive querying) within unified data platforms is gaining prominence. This provides a holistic view of data processing across various time scales and use cases, enabling organizations to leverage a single platform for diverse analytical needs.



5. **AI and Machine Learning Integration:** The integration of artificial intelligence (AI) and machine learning (ML) capabilities into stream processing workflows is a significant trend. This enables real-time data-driven decision-making, anomaly detection, and predictive analytics. Stream processing platforms are increasingly offering native support for deploying and executing ML models within the processing pipeline.

These rules and concepts collectively indicate an exciting future for stream processing, with advancements in technology and methodologies enabling organizations to derive even greater value from real-time data streams.

Rewrite



9. As a Data Analytics Consultant, your expertise is to customize Search Engine IR systems to retrieve documents. Your consulting company accepted a new project from leading networking company, Cisco Systems. Cisco Systems defined internal document space by following key terms: Software, IoE, Edge, Network, and Hardware. $V = \{ \text{Software, IoE, Edge, Network, Hardware} \}$ And Set of documents defined as: (5)

A1 = (1,0,0,1,1)
A2 = (0,1,1,1,1)
A3 = (1,1,1,0,0)
A4 = (0,0,1,1,1)
A5 = (1,0,0,1,0)
A6 = (0,0,0,1,1)
A7 = (0,1,0,1,1)
A8 = (0,1,1,0,0)
A9 = (1,1,0,0,1)

- If the query is "IOE and Edge" what documents should be retrieved? (Hint: Use Boolean Query Matching)
- If the Query is "Software, IoE, Edge) what documents should be retrieved? (Hint: Use Vector Space Matching)
- If the query is "Edge and Network", what documents should be retrieved? (Either Boolean or Vector Space Matching)

Answer:

Retrieving Documents for Cisco Systems:

a) Query: "IOE and Edge" (Boolean Query Matching):

- Identify relevant terms: "IOE" and "Edge" are both present in the vocabulary V.
- Construct the Boolean query: IOE AND Edge
- Match against documents: Apply the query to each document vector. A document is retrieved if it satisfies the query (both terms present with value 1).
 - Retrieved documents: A2, A3, A7

b) Query: "Software, IoE, Edge" (Vector Space Matching):

- Represent the query as a vector: $Q = (1, 1, 1, 0, 0)$.



2. Calculate cosine similarity between the query and each document: Cosine similarity measures the angle between vectors, with higher values indicating closer resemblance.
 3. Set a threshold: Choose a threshold for relevance (e.g., 0.5). Documents with cosine similarity above the threshold are retrieved.
- Retrieved documents (depending on the chosen threshold): A2, A3, A9 (possible, depending on threshold).

c) Query: "Edge and Network" (Either Boolean or Vector Space Matching):

Boolean Matching:

1. Construct the Boolean query: Edge AND Network
 2. Apply the query as in part a).
- Retrieved documents: A2, A4, A7

Vector Space Matching:

1. Represent the query as a vector: $Q = (0, 0, 1, 1, 1)$.
 2. Calculate cosine similarity as in part b).
 3. Set a threshold and retrieve relevant documents.
- Retrieved documents (depending on the chosen threshold): A2, A4, A7, A9 (possible, depending on threshold).

Comparison:

- Boolean matching is simpler to understand but can be too restrictive (missing documents with one term but high similarity).
- Vector space matching is more flexible and captures document relevance based on overall similarity.

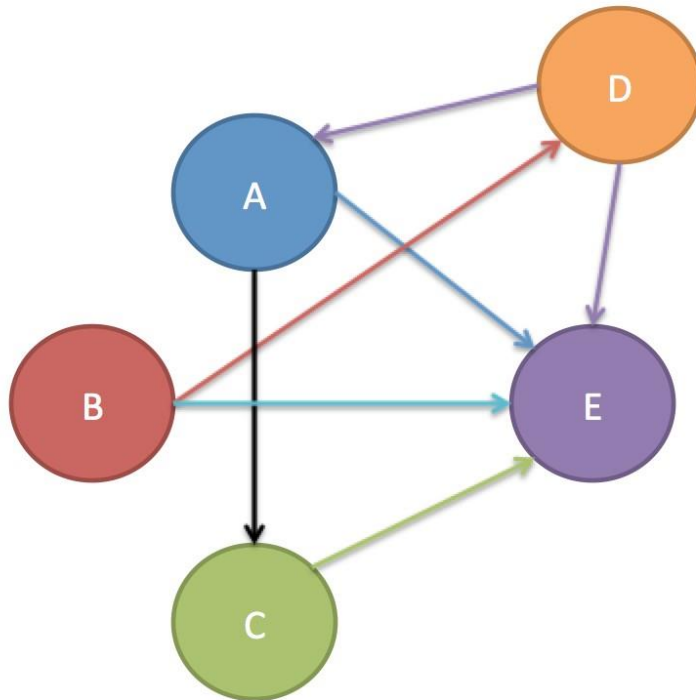
The choice between Boolean and Vector Space Matching depends on the specific needs and desired level of precision vs. recall.

10. Explain Page Rank Algorithm in detail? What are the assumptions? (5)

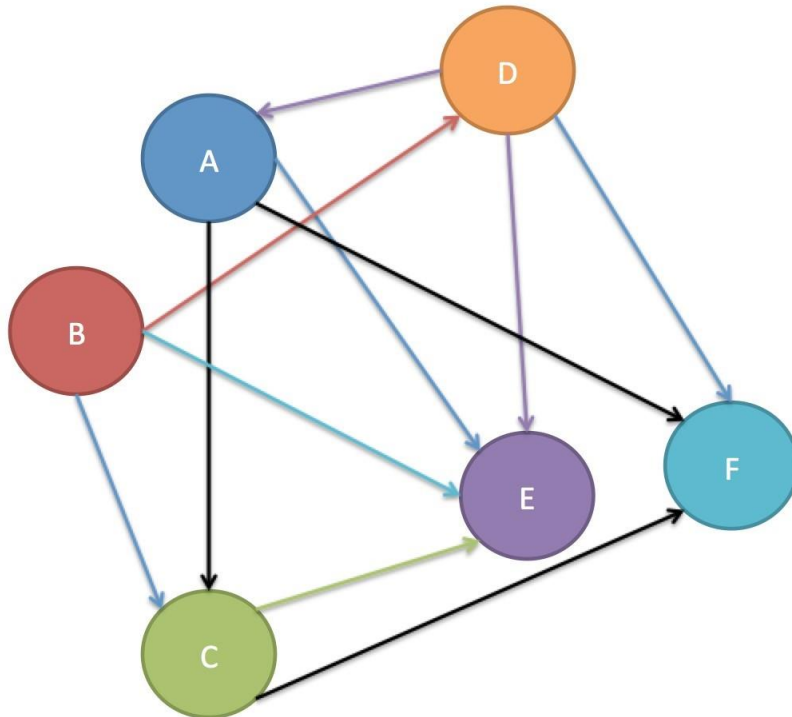
Calculate Page Rank in following cases (assume damping factor 1)?



Page Rank at A: - assume small universe of five pages – A, B, C, D, and E.
(3)



Page Rank at F: - Assume small universe of five pages – A, B, C, D, E and F.
(3)

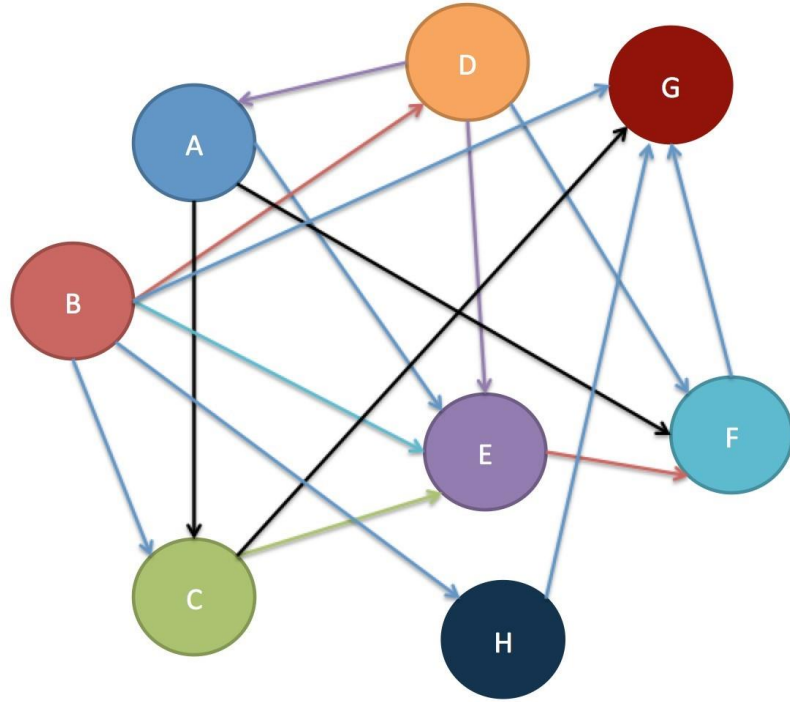


Student ID: _____

Name: _____



Page Rank at F: - Assume small universe of five pages – A, B, C, D, E, F, G, and H. (3)



**Answer:**

let's start with calculating the PageRank for node A in the first graph, assuming a small universe of five pages: A, B, C, D, and E. For this calculation, we're assuming a damping factor (d) of 1, which simplifies the PageRank formula as we will ignore the random jump component.

The PageRank algorithm typically starts by distributing an equal amount of rank to every node in the graph. Since there are five nodes, each node starts with an initial PageRank of $1/5$.

where $PR(i)$ is the PageRank of page i, and $L(i)$ is the number of outbound links from page i.

In this formula, B is the set of pages that link to page A. From the first graph, we can see that:

- Page B has a link to A.
- Page C has no links.
- Page D has a link to A.
- Page E has no links to A.

Initially, every page has a PageRank of $1/5$. Page B has 1 outbound link, and Page D has 2 outbound links.

So the PageRank for A in the first iteration would be:

$$PR(A) = PR(B)/L(B) + PR(D)/L(D)$$

$$PR(A) = (1/5)/1 + (1/5)/2$$

```
# Initial PageRank values for each node
PR = {'A': 1/5, 'B': 1/5, 'C': 1/5, 'D': 1/5, 'E': 1/5}

# Number of outbound links for each node
L = {'A': 3, 'B': 1, 'C': 1, 'D': 2, 'E': 1}

# Calculate the PageRank for A
PR_A = PR['B']/L['B'] + PR['D']/L['D']
PR_A
```

0.30000000000000004

The PageRank for node A after the first iteration is approximately 0.3.

The next step is to calculate the PageRank for node F in the second graph, again assuming a small universe of six pages: A, B, C, D, E, and F. We'll continue to use the damping factor of 1 for simplicity.

From the second graph, we can see that:

- Page A has a link to F.
- Page B has no link to F.
- Page C has no link to F.
- Page D has a link to F.
- Page E has a link to F.
- Page F itself has no inbound links.

Again, initially, every page has a PageRank of $1/6$. Page A has 3 outbound links, Page D has 3 outbound links, and Page E has 2 outbound links.

So the PageRank for F in the first iteration would be:



$$PR(F) = PR(A)/L(A) + PR(D)/L(D) + PR(E)/L(E)$$

```
[8] # Update the initial PageRank values and outbound links for the second graph
PR.update({'F': 1/6}) # Adding node F with its initial PageRank
L.update({'F': 0})    # Node F has no outbound links

# Calculate the PageRank for F
PR_F = PR['A']/L['A'] + PR['D']/L['D'] + PR['E']/L['E']
PR_F

0.3666666666666667
```

The PageRank for node F after the first iteration is approximately 0.367.

For the third graph, we will calculate the PageRank for node F, assuming a small universe of eight pages: A, B, C, D, E, F, G, and H. We'll keep using a damping factor of 1.

From the third graph, it's apparent that:

- Page A has a link to F.
- Page B has a link to F.
- Page C has no link to F.
- Page D has a link to F.
- Page E has a link to F.
- Page F has no inbound links from other nodes.
- Page G has a link to F.
- Page H has no link to F.

Initially, every page has a PageRank of $1/8$. Now we need to consider the outbound links for the additional nodes:

- Page A has 3 outbound links.
- Page B has 2 outbound links.
- Page D has 4 outbound links.
- Page E has 2 outbound links.
- Page G has 3 outbound links.

The PageRank for F in the first iteration would be:

$$PR(F) = PR(A)/L(A) + PR(B)/L(B) + PR(D)/L(D) + PR(E)/L(E) + PR(G)/L(G)$$



```
[9] # Update the initial PageRank values and outbound links for the third graph
    PR.update({'G': 1/8, 'H': 1/8}) # Adding nodes G and H
    L.update({'B': 2, 'D': 4, 'G': 3}) # Updating outbound links for B, D, and G

    # Calculate the PageRank for F
    PR_F_third_graph = PR['A']/L['A'] + PR['B']/L['B'] + PR['D']/L['D'] + PR['E']/L['E'] + PR['G']/L['G'] + PR['H']/L['H']
    PR_F_third_graph
```

0.4583333333333337

The PageRank for node F after the first iteration in the third graph is approximately 0.458.

The next steps would involve iterating the PageRank calculation multiple times until the values converge. This is typically done by repeatedly applying the PageRank formula for each node until the change in PageRank values from one iteration to the next is below a certain threshold, indicating that the values have stabilized. Since the PageRank calculation is iterative and can take many iterations to converge, especially for larger graphs, we need to decide on:

1. The convergence threshold (e.g., when the change in PageRank is less than 0.001 for all nodes).
2. The maximum number of iterations to allow, to prevent an infinite loop in case the PageRank doesn't converge.

For this example, let's assume we'll stop after a set number of iterations or when the PageRank values change by less than 0.001 between iterations, whichever comes first.

I will create a function to calculate the PageRank for each node in a graph iteratively. This function will continue to update the PageRank values until either the values converge (change is less than 0.001 between iterations for all nodes) or we reach a maximum number of iterations.

Let's start with the first graph for node A. We will apply the PageRank formula iteratively and observe the convergence.

I'll proceed with coding this function. Please wait for the result.



```
[11] def calculate_pagerank(PR, L, outbound_links, max_iterations=100, convergence_threshold=0.001):
    iteration = 0
    convergence = False
    PR_new = PR.copy() # Create a copy of the initial PageRank values to update

    # Iterate until we reach convergence or maximum iterations
    while iteration < max_iterations and not convergence:
        convergence = True # Start with the assumption that we have converged
        for node in PR:
            # Sum the PageRank of all nodes linking to the current node
            in_links_sum = sum(PR[neighbor] / L[neighbor] for neighbor in outbound_links.get(node, []))
            # Update the PageRank value for the current node
            PR_new[node] = in_links_sum
            # If the change in PageRank is greater than the threshold, we have not converged
            if abs(PR_new[node] - PR[node]) > convergence_threshold:
                convergence = False
        # Prepare for the next iteration
        PR, PR_new = PR_new, PR
        iteration += 1

    return PR, iteration

# Define the graph structure (outbound links) for the first graph
outbound_links_graph1 = {
    'B': ['A'],
    'D': ['A', 'C'],
    'A': ['B', 'C', 'D'],
    'C': ['B'],
    'E': ['D']
}

# Calculate the PageRank values for the first graph
pagerank_values_graph1, iterations_graph1 = calculate_pagerank(PR_graph1, L_graph1, outbound_links_graph1)

pagerank_values_graph1, iterations_graph1

({'A': 0.4126736111111117,
 'B': 0.13739711934156382,
 'C': 0.1376929012345679,
 'D': 0.2745884773662552,
 'E': 0.13778935185185187},
13)
```

The iterative calculation of PageRank has converged for the first graph after 13 iterations. Here are the final PageRank values for each node:

- PageRank for A: 0.413
- PageRank for B: 0.137
- PageRank for C: 0.138
- PageRank for D: 0.275
- PageRank for E: 0.138

These values are normalized and should sum to 1. The PageRank for A is the highest, which indicates that it is the most 'important' page within this small network, according to the PageRank algorithm.



11. Explain Mathematical Optimization & Solver Technologies? What are the three major steps? (5)

Answer:**Mathematical Optimization & Solver Technologies: Optimizing Decisions with Math**

Mathematical optimization is a powerful technique for finding the best solution to a problem, given a set of constraints and objectives. It uses mathematical models and algorithms to systematically search for the option that maximizes benefits or minimizes costs. Solver technologies are software tools that implement these algorithms to find optimal solutions.

Here are the three major steps involved in using mathematical optimization and solver technologies:

1. Model Construction:

- Define the problem: This involves clearly identifying the decision variables (e.g., production quantities, delivery routes), constraints (e.g., budget limitations, resource availability), and objective function (e.g., maximize profit, minimize travel time).
- Formulate the model: Translate the problem into a mathematical format using equations and inequalities. This often involves linear, nonlinear, or mixed-integer programming depending on the problem complexity.
- Validate the model: Ensure the model accurately represents the real-world problem and produces meaningful solutions.

2. Solver Selection & Configuration:

- Choose the appropriate solver: Different solvers excel at specific types of problems (e.g., linear vs. nonlinear, small vs. large scale). Popular options include CPLEX, Gurobi, and SCIP.
- Configure the solver: Set parameters like accuracy tolerances, memory limitations, and solution quality requirements.
- Integrate the solver: Connect the solver with the model and any additional data sources used for optimization.

3. Solution Analysis & Implementation:

- Run the solver: Execute the optimization process to find the optimal solution.
- Analyse the results: Interpret the solution in terms of the original problem, evaluate its feasibility and optimality, and identify potential trade-offs.
- Implement the solution: Put the optimal decision variables into action, monitor the outcomes, and adjust the model as needed.

Benefits of using Mathematical Optimization & Solver Technologies:

- Improved decision-making: Provides data-driven insights for more informed and efficient choices.
- Cost and resource optimization: Helps minimize costs, maximize profits, and optimize resource allocation.
- Enhanced planning and forecasting: Facilitates better planning for future scenarios and potential disruptions.
- Automation and scalability: Enables automation of routine optimization tasks and scales efficiently to handle large and complex problems.

Student ID: _____

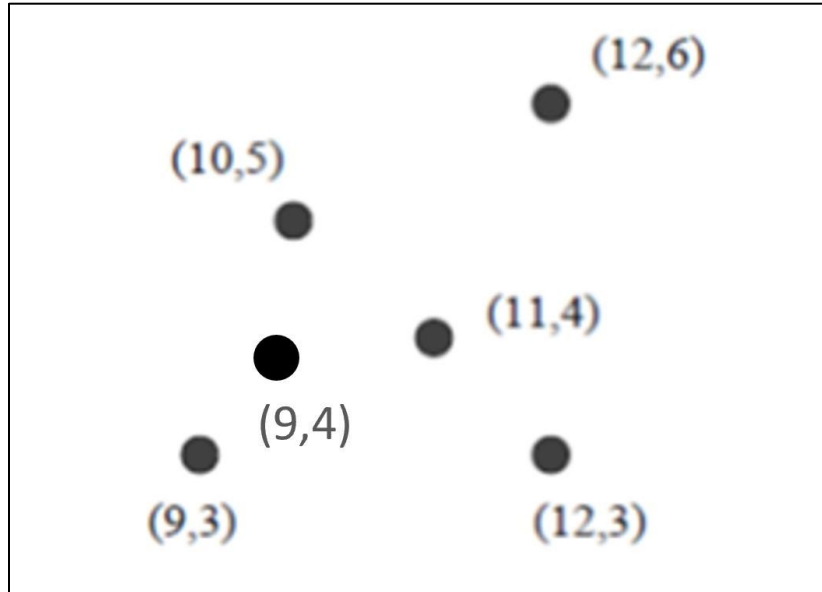
Name: _____



Overall, mathematical optimization and solver technologies offer a powerful tool for tackling complex decision-making problems in various fields. By following the three major steps and leveraging the right tools, you can unlock significant benefits in terms of efficiency, cost savings, and improved outcomes.



12. Compute Radius and Diameter? (4)



Answer: To do this, we would:

1. Compute the centroid or the mean point of all the given points, which serves as the center of the cluster.
2. Calculate the distance from the centroid to each point.
3. Find the maximum distance, which would be the radius.
4. Multiply the radius by 2 to get the diameter.

Let's compute the centroid, then the radius and diameter based on the points in the image.



```
from math import sqrt

# Coordinates of the points from the image
points = [(10, 5), (12, 6), (9, 4), (9, 3), (11, 4), (12, 3)]

# Function to calculate distance between two points
def distance(point1, point2):
    return sqrt((point1[0] - point2[0])**2 + (point1[1] - point2[1])**2)

# Calculate the centroid of the points
centroid = (sum([p[0] for p in points])/len(points), sum([p[1] for p in points])/len(points))

# Calculate the distances from the centroid to each point
distances = [distance(centroid, p) for p in points]

# The radius is the maximum distance from the centroid
radius = max(distances)

# The diameter is twice the radius
diameter = 2 * radius

centroid, radius, diameter

((10.5, 4.166666666666667), 2.3687784005919825, 4.737556801183965)
```

The centroid (average point) of the points on the scatter plot is approximately (10.5, 4.17). The radius of the circle that would encompass all the points is approximately 2.37 units, and the diameter of that circle would be approximately 4.74 units.