APPROACHES TO THE INFERENCE PROBLEM

02 mar 2023

Different approaches

- employed for building an inference controller
    - machine learning techniques
        - to build a learner
            - automatically learns to recognize complex patterns
                - make intelligent decisions based
    - Aggregation problem
    - Association problem
    - Domain Restriction.
    - Statistical Reasoning

Aggregation problem

- $A \cup B \cup C \subseteq$ Secret
  - collection of data elements is Secret but the individual elements are Unclassified.
  - Suppose that a company wants to conduct a survey of its employees to assess their job satisfaction

- The situation involves a company conducting a survey of its employees to assess job satisfaction.
- The company designates the survey data as "Confidential" to keep the results of the survey confidential.
  - the individual responses of the employees are not considered confidential.
- This creates an aggregation problem where if someone were to aggregate the individual responses of multiple employees, they could potentially infer sensitive information about the company or its employees.
- Each individual response is considered unclassified, but when combined with other responses, they may reveal sensitive information.
  - Safeguards needed to mitigate the aggregation problem.
  - Aggregation methods that preserve employee anonymity.
  - Limiting access to survey data.
  - Using statistical techniques to protect against inference attacks

- Differential privacy and k-anonymity are examples of statistical techniques used to protect against inference attacks.
- Differential privacy involves adding random noise to the data to protect individual responses.
- K-anonymity is a method that ensures the protection of individual identities by grouping them into clusters or categories, while still enabling effective analysis of the data.

Association problem :Classify and manage sensitive medical information

- Doc_n_6 opm:wasDerivedFrom med:Doc_n_5) ∧ (med:Doc_n_6 opm:wasGeneratedBy med:HeartSurgery_n_1) ∧ (med:HeartSurgery_n_1 opm:WasControlled)By med:Surgeon_n_1) → (med:HeartSurgery_n_1 med:Classification Secret)

  - This is a logical rule in which the conclusion states that the "med:HeartSurgery_n_1" document is classified as "Secret".

The patient’s record that was generated by this operation should be considered secret as well

- (med:Doc_n_6 opm:wasDerivedFrom med:Doc_n_5) ∧ (med:Doc_n_6 opm:wasGeneratedBy med:HeartSurgery_n_1) ∧ (med:HeartSurgery_n_1 opm:WasControlled)By med:Surgeon_n_1) → (med:Doc_n_6 med:Classification Secret)

- A ∩ B ∩ C ⊆ Secret:Something that is in all three classes is private
- If at most one individual is in all three classes, then classify KB: ≤1R.(A ∩ B ∩ C)

A social network, let's consider the sets A, B, and C as follows:

- Set A includes users who have "private" profiles, which can only be viewed by their friends.
- Set B includes users who have posted "sensitive" content, such as personal information or confidential data.
- Set C includes users who have participated in "closed" groups, where only members can see the content
- According to the statement "$A \cap B \cap C \subseteq$ Secret
  - users who belong to all three sets have something that is considered private.

If at most one individual is in all three classes, then classify KB: ≤1R.(A ∩ B ∩ C)

- Set A includes users who have "private" profiles, which can only be viewed by their friends.
- Set B includes users who have posted "sensitive" content, such as personal information or confidential data.
- Set C includes users who have participated in "closed" groups, where only members can see the content.
  - The statement "≤1R.(A ∩ B ∩ C)" means that at most one individual is in all three sets.
  - The intersection of sets A, B, and C has a maximum of one user in common.
  - This implies that there is no overlap between any three users from sets A, B, and C.

- John has a private profile, Alice has posted sensitive content, and Bob has participated in a closed group.
- None of these users have any overlap in terms of their membership in sets A, B, and C.
- The statement "$\leq 1R.(A \cap B \cap C)$" is satisfied.
  - KB can be classified as "low risk" since there is a low probability of sensitive information being leaked through the intersection of sets A, B, and C.

Domain Restriction

- a range restriction on a qualified value for a property as Secret
  - a property whose value is something with nine digits is Sensitive.(SSN)
  - if something with nine digits is released, then we need to classify the KB
  - something with 16 digits is a credit card number.
- If at most one heart surgeon is on duty during a patient's visit
- <med:HeartSurgery_n_1><opm:WasControlledBy><_:b>

1. Subject: "med:HeartSurgery_n_1" represents the subject of the triple, which is the heart surgery procedure being discussed.
2. Predicate: "opm:WasControlledBy" represents the predicate of the triple, which is the relationship between the subject and the object.
3. Object: "<_:b>" represents the object of the triple, which is a blank node that refers to an unnamed or anonymous resource

Questions

- 1.If Exam_n_4 was derived from Exam_n_3 and was generated by Doctor_n_1, who was controlled by Nurse_n_1, then Exam_n_4 should be classified as confidential.
- 2.Lgical rule implies that if a student has taken Calculus and Physics courses, and Physics is a prerequisite for Calculus, then the student should be classified as "Advanced" in their academic standing.
- 3. A logical rule for assessing a patient's risk factor based on their blood pressure, BMI, and cholesterol level. If all three conditions are met, then the patient is classified as having a high risk factor

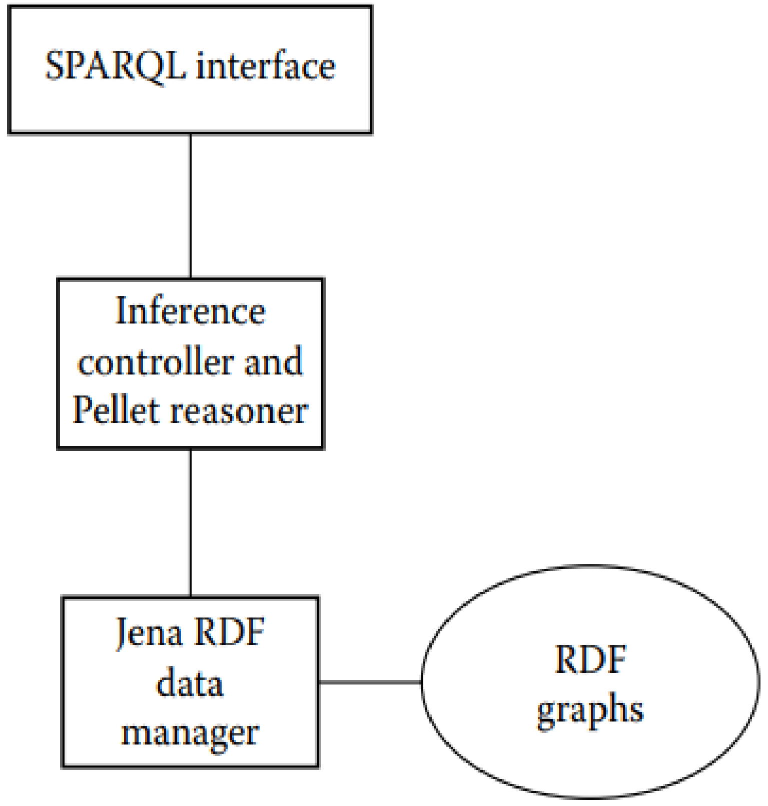
Implementation of an inference controller for provenance.

Unit II

- Inference controller for provenance implemented using a modular approach
- Modules can be extended or replaced, providing flexibility for users
- Plug-and-play approach used for implementation, utilizing open source products such as Jena and Pellet
- Policies and data represented in RDF and managed using Jena
- Reasoning based on Pellet, with additional inference strategies built
- Inference controller examines policies, rewrites queries, and retrieves authorized data when a user poses a query in SPARQL
- Functionality demonstrated through examples in a health-care application.

Implementation architecture.

•The system manages policies and data using RDF.
•Jena is used to manage the RDF data.
•Pellet is used as the reasoning engine.
•Additional inference strategies have been built on top of Pellet.
•When a user poses a query in SPARQL, the inference controller examines policies to determine authorized data.
•The query is then rewritten based on policies and queries the Jena RDF store.
•The system can reason about complex policies and data structures.
•Users can only access data they are authorized to see



Provenance in a Health-Care Domain

1. <u>**Need for provenance**</u>
2. Provenance is a critical component in the healthcare domain.
3. It enables communication and coordination among healthcare organizations and medical teams.
4. It provides an integrated view of treatment processes.
5. It facilitates the analysis of healthcare service performance.
6. It enables audits to assess that proper decisions were made and proper procedures were followed.
7. Provenance tracks and records the history of medical data.
8. It records who generated the data, when it was generated, and how it was modified.
9. Provenance helps ensure proper decisions and procedures for better patient outcome

Populating the Provenance Knowledge Base

1. The provenance knowledge base is updated using a set of generators.
2. Background generators extract background information available online.
3. A workflow generator produces synthetic provenance data that is not available online.
4. The workflow generator produces actual provenance data.
5. The provenance data has subsets that must be protected.
6. The provenance store is populated by extracting information related to a health-care domain.
7. The health-care domain records provenance and some of its data is available online

- **Provenance is critical for maintaining trustworthy patient data in electronic health record systems.**

- Provenance
  - essential for maintaining the integrity and security of electronic health records.
  - tracking the origin and history of patient data to ensure accuracy and trustworthiness.
  - helps healthcare professionals make better-informed decisions based on reliable data.
    - improved decision-making can lead to better patient outcomes
    - to analyze performance and carry out audits.
    - can identify areas for procedure and workflow improvements.

1.The hospital wants to track the provenance of patient data in their electronic health record system.
2.Background generators extract publicly available information about medical procedures, equipment, and other relevant entities.
3.A workflow generator produces synthetic provenance data based on the actual procedures followed by the hospital.
4.The synthetic provenance data includes information about who accessed the patient record, when it was accessed, and what procedures were performed.
5.The provenance store is populated by extracting information related to the health-care domain,
such as patient demographics, lab results, and treatment plans.
6.Subsets of data that must be protected, such as patient identifiers and sensitive medical information, are not released to unauthorized parties.
7.Provenance helps ensure that patient data is accurate, complete, and trustworthy.
8.Provenance can be used for ensuring the accuracy, completeness, and trustworthiness of patient data
9.By examining the provenance data, hospitals can identify areas where improvements can be made to their procedures and workflows.

Generating and Populating the Knowledge Base.

- The process described
  - creating a set of seeds,
    - consisting of a first name, last name, state, and city, to create queries against yellowpages.com
  - web crawler
    - to extract information from the websites in an automated manner
    - crawled pages are then stored in an appropriate format in a text file
    - The patient lists gathered from the first crawl are used to create queries for hospitals, doctors, and their specialties based on the patients' zip codes
      - the creation of object classes for entities related to healthcare, such as persons, hospitals, doctors, and nurses.
        - the information gathered from the web pages can be easily organized and utilized for further analysis and processing

Generating Workflows

- The system updates patient records and records provenance data.
- Provenance data contains information about the changes made to the patient's record.
- The system is designed to allow querying users to guess the patient's disease, medications, or tests associated with the record.
- The controller of the system must anticipate inferences involving the user's prior knowledge and the causal relationships among the provenance data objects.
- The system is designed to support complex queries and analysis of the data.
- If the provenance data contains confidential information, appropriate measures must be taken to protect the privacy and security of the data.

Properties of the Workflow.

- Let's say we have a workflow for a patient record that includes the following entities:
    - doctor (Agent A), a prescription medication (Entity B), and a lab test result (Entity C).
    - The OPM toolbox is used to generate the skeleton of the workflow, capturing the relationships between the entities
        - P: WasControlledBy, Used, WasDerivedFrom, WasGeneratedBy, and WasTriggeredBy.
    - initial workflow generated does not include any RDF triples related to the entities
        - add RDF triples that provide additional information
            - name, type, or other properties
            - other attributes are added to scale the size

INFERENCE AND PROVENANCE

01 MAR 2023

R DEVIKA /CSE/SoC

Problem of inference in data protection systems

- Traditional methods of protecting data
  - fail when dealing with data organized as directed graphs
- People can use data queries to figure out sensitive information even if the protection rules are in place
- Inference refers to people drawing conclusions without permission from those who control the data.
- Inference can occur using information from a knowledge base or from the user's own knowledge
- Therefore,
  - protecting data requires developing mechanisms that can effectively handle directed graphs and prevent inference from occurring

Inferred knowledge obtained from a knowledge base

1. The interpretation of a negative answer to a query can depend on whether the knowledge base uses a closed-world or an open-world assumption.
2. In a closed-world assumption, a negative answer typically indicates that the data is not present in the knowledge base.
3. In an open-world assumption, a negative answer does not necessarily mean that the data is not present but could be available elsewhere in the system
4. Example
   - The user queries a knowledge base for information on a patient's medical procedures.
   - A positive answer from the knowledge base indicates that the patient had the procedure.
   - A negative answer may be interpreted differently depending on the knowledge base's assumptions.
   - In a closed-world assumption, a negative response typically means the data is not present.
   - In an open-world assumption, a negative response could mean that the data is not present in the knowledge base but could be available elsewhere in the system.

Rules of inference

- used to infer a conclusion from a premise/statemet to create an argument
- A complete set of rules can infer any valid conclusion.
- A sound set of rules ensures no invalid conclusion is drawn.
- A sound and complete set of rules need not include every rule to arrive at a conclusion.
  - without including unnecessary or redundant rules.

Complete set of rules

- Premise 1: All humans are mortal.
- Premise 2: Socrates is a human.
- Rule of inference: If X is a human and all humans are mortal, then X is mortal.
- Conclusion: Therefore, Socrates is mortal.

Sound set of rules

- Premise 1: All dogs have wings.
- Premise 2: Fido is a dog.
- Rule of inference: If X is a dog and all dogs have wings, then X has wings.
- Invalid conclusion: Therefore, Fido has wings.

Sound set of rules (Cont’d)

- Premise 1: All birds can fly.
- Premise 2: Penguins are birds.
- Conclusion: Therefore, penguins can fly
- Rule of inference used in this example is not sound, and it leads to an invalid conclusion.

A forward-chaining mode

- Forward-chaining is a mode of inference used in artificial intelligence.
- It starts with the available data and uses inference rules to derive new information and draw conclusions.
- The new information is added to the knowledge base, and the process repeats until a goal is achieved or no more new information can be derived.
- Forward-chaining is used in expert systems, rule-based systems, and other AI applications to make decisions, solve problems, and provide recommendations.
- It is also known as data-driven reasoning or bottom-up reasoning

- Examples of forward-chaining applications include medical diagnosis, fraud detection, and supply chain optimization.
- One key advantage of forward-chaining is that it is incremental and can handle large amounts of data efficiently.
- However, it can also be computationally expensive and may generate many irrelevant conclusions.

A forward-chaining mode (Cont’d)

These are rules in a medical ontology that relate to surgery and control of medical procedures

1. (med:HeartSurgery opm:wasControlledBy ?X) → (X rdf:type med:Surgeon)

Rule 1 states that if a heart surgery is controlled by a person or entity represented by the variable ?X, then ?X must be of type Surgeon according to the medical ontology.

2. (med:Results_n_1 opm:Used med:Doc_n_6) ∧ (med:Results_n_1 opm:wasControlledBy ?X) → (X rdf:type med:Surgeon

Rule 2 states that if a certain result identified by "Results_n_1" uses a document represented by "Doc_n_6", and that result is controlled by a person or entity represented by the variable ?X, then ?X must be of type Surgeon according to the medical ontology.

If our knowledge base contains the following triples

1. <med:HeartSurgery_n_1> <opm:Used><med:Doc_n_5>
2. <med:Doc_n_6><opm:WasGeneratedBy><med:HeartSurgery_n_1>
3. <med:Doc_n_6><opm:wasDerivedFrom><med:Doc_n_5>
4. <med:Results_n_1><opm:Used><med:Doc_n_6>
5. <med:Results_n_1><opm:WasControlledBy><med:Surgeon_n_1>
6. <med:HeartSurgery_n_1><opm:WasControlledBy><med:Surgeon_n_1>

FACT FORM KNOWLEDGE BASE: this information can be used to track the involvement of different people and documents in the medical procedure, from generating the document to controlling the results

Backward chaining

- Backward chaining is a reasoning method used in artificial intelligence.
- It starts with a given expression, which is the consequent that we want to prove.
- The system then works backwards to determine all of the antecedents, which are the conditions that must be satisfied for the consequent to be true.
- Backward chaining is commonly used in expert systems and rule-based systems.
- It is often used in diagnostic applications, where the system needs to determine the root cause of a problem based on a set of symptom

Backward chaining (cont’d)

- Using backward chaining, we start with the consequent expression, which in this case is "User A is connected to User B”
- User A is friends with User C
- User C is friends with User D
- User D is friends with User E
- User E is friends with User F
- User F is friends with User G
- User G is friends with User H
- User H is friends with User B

Approaches to the Inference Problem

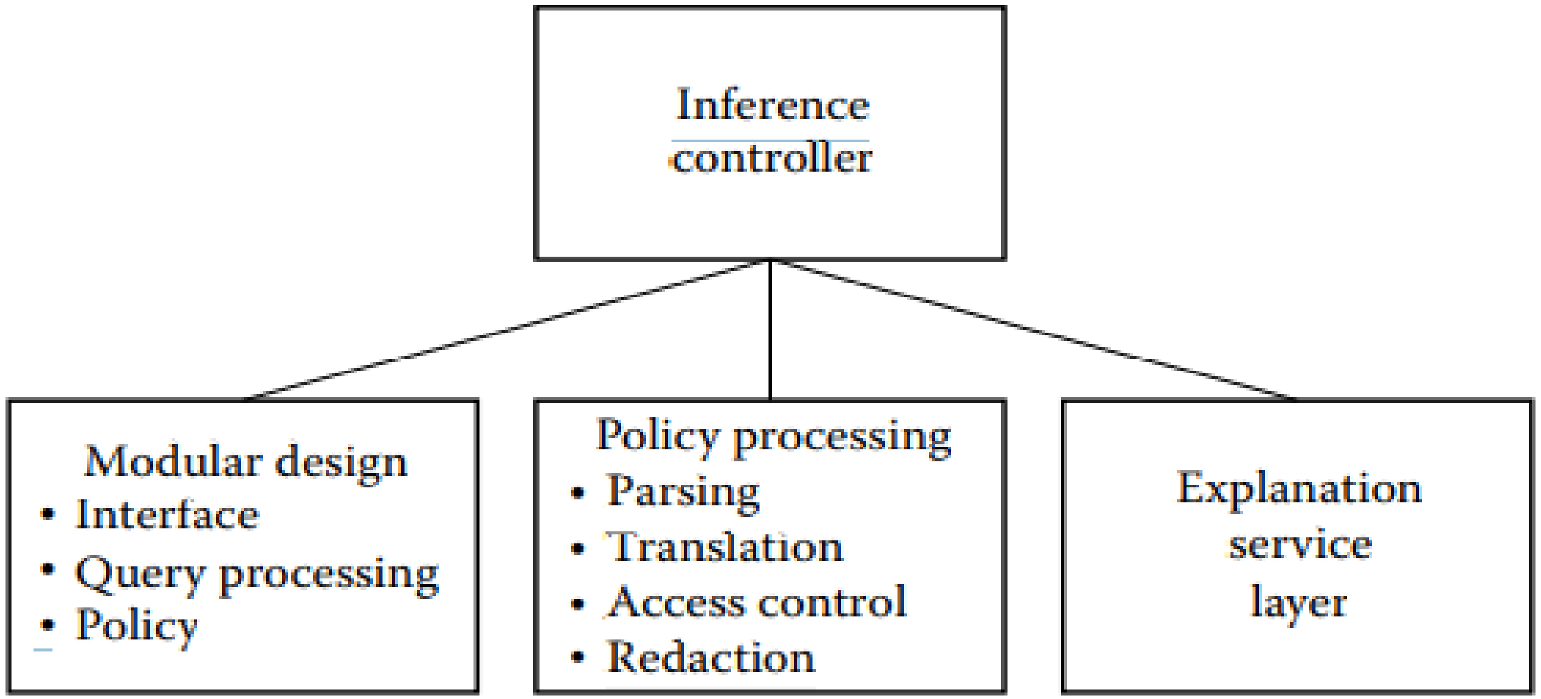
Inference Control for Social Media Using Semantic Web Technologies

R Devika APII/CSE/SoC

Introduction

- Social media platforms are popular but can be problematic due to the inference problem, where users can draw unauthorized conclusions from legitimate responses
- Inference controllers can prevent the inference problem and ensure data privacy and security
- Semantic web technologies, such as RDF graphs, can be used for provenance data and inference control for social media

Architecture of an inference controller



Inference:

- Inference involves drawing conclusions or making predictions based on available information or evidence.
- Social networks might use machine learning algorithms to infer users' interests, preferences, or demographic characteristics based on their behavior.
- For example, if a user frequently likes or comments on posts about veganism, the social network might infer that they are interested in plant-based diets

Controller

- The controller is a component or module of a system that is responsible for managing the flow of data or events between other components or modules.
- In the context of a social network, the data controller would be responsible for managing the collection, storage, and processing of users' personal data in compliance with data protection regulations.
- For example, the data controller might implement measures to secure users' data, such as encryption or access control mechanisms

Explanation service layer:

- The explanation service layer is a software layer that provides explanations or reasoning for the outputs or results produced by a system or application.
- In the context of a social network, the explanation service layer might provide users with explanations for why they see certain posts or ads in their feed, based on their interests, location, or other factors.
- For example, the explanation service layer might inform a user that they are seeing a particular post because it was liked by many of their friends

Modular design

- Modular design is an approach to system or software design that emphasizes the use of independent, interchangeable modules or components that can be easily combined or replaced to create new systems or applications.
- In the context of a social network, modular design might involve implementing different features or functions (such as messaging, groups, or events) as independent modules that can be combined or customized according to user needs.
- For example, a user might be able to customize their profile page by adding or removing different modules, such as a photo gallery or a list of favorite books

Policy processing

- Policy processing refers to the process of applying rules or policies to data or events in a system to enforce compliance with regulatory or organizational guidelines.
- In the context of a social network, policy processing might involve enforcing policies around hate speech, bullying, or nudity through content moderation tools and user reporting mechanisms.
- For example, the social network might use AI-powered content moderation tools to automatically flag posts or comments that contain offensive language or images

Interface

- The interface is a point of interaction between two or more components or systems, allowing them to exchange data or information
- In the context of a social network, the interface might include features for posting, commenting, liking, or sharing content, as well as for managing privacy settings or searching for other users
- For example, the social network might provide users with a user-friendly interface that allows them to easily navigate the platform and interact with other users

Query processing

- Query processing refers to the process of executing user queries on a database or other data storage system, retrieving the requested data and returning it to the user

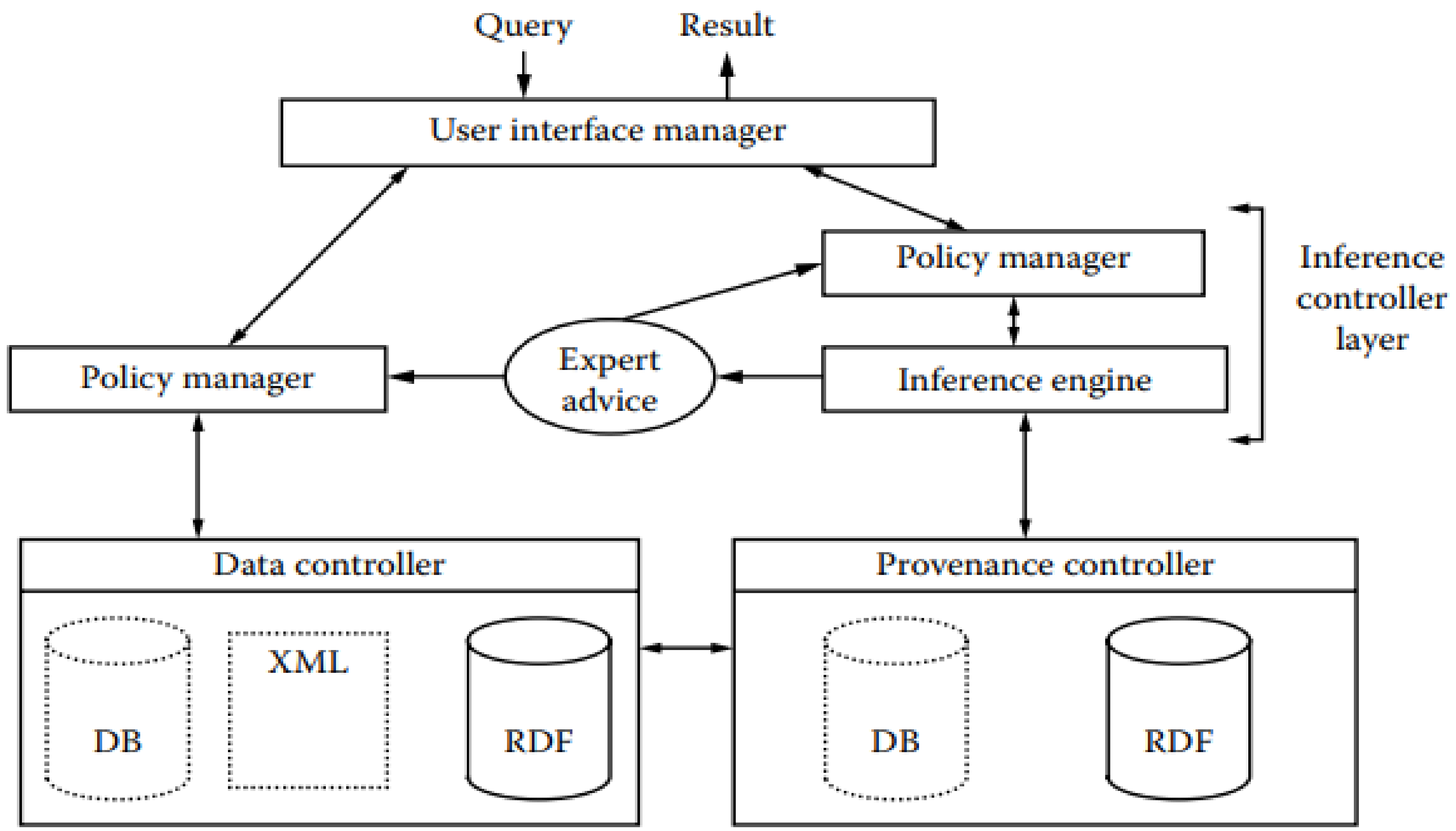
- In the context of a social network, query processing might involve retrieving information about users, such as their profiles, posts, or activity history

- For example, a user might use the social network's search function to find posts or comments that mention a particular topic or keyword

Policy

- A policy is a set of rules or guidelines that dictate how a system or organization operates, often in relation to security, privacy, or regulatory compliance
- In the context of a social network, the social network might have policies around data privacy, content moderation, or user behavior that are communicated to users through terms of service agreements or community guidelines
- For example, the social network might have policies around hate speech

System modules: Inference Control for Social Media



User Interface Manager

1. **Processing user requests**: The user interface manager is responsible for handling requests made by the user.
2. **Authenticating the user:** The user interface manager is responsible for verifying the identity of the user and ensuring that they have the appropriate permissions to access the system.
3. **Providing suitable responses:** The user interface manager must provide appropriate and relevant responses to the user's requests.
4. **Providing an abstraction layer**: The user interface manager provides an abstraction layer that allows users to interact with the system without needing to know the underlying technical details.
5. **Handling data and provenance queries**: The user interface manager can handle both data queries and provenance queries, depending on the nature of the user's request.
6. **Determining query evaluation:** The user interface manager determines whether a query should be evaluated against traditional data or provenance information

The policy manager

- need to ensure that only authorized users can view private user data, such as personal messages or photos.
- need to ensure that users' personal information is not shared with third parties without their consen
- need to evaluate and enforce policies that regulate user interactions on the platform.
- need to be able to evaluate user queries and associated query results, as well as be able to distinguish between different types of data (e.g. user profile data vs. user activity data)

Inference Engine

1. An inference engine is the central component of an inference controller.
2. The inference engine is responsible for executing various reasoning strategies supported by a particular reasoner.
3. It offers flexibility by allowing users to choose from among various reasoning tools for each reasoning task.
4. The inference engine enables improved efficiency as each strategy or combination of strategies can be executed on a separate processor.
5. The inference engine typically utilizes software programs capable of performing reasoning over different data representations.
6. These data representations can include relational data or RDF graph models.
7. The inference engine can perform reasoning tasks over the data representations using different reasoning strategies.
8. The use of an inference engine can result in more efficient and accurate inference in complex reasoning tasks.
9. The inference engine is an essential tool in various domains, including artificial intelligence, expert systems, and natural language processing

Data Controller

- The data controller is a suite of software programs
- It stores and manages access to data
- Data can be stored in various formats (relational database, XML files, RDF store)
- The controller accepts requests for information from the policy manager or inference engine layer
- The request is executed over stored data
- Results are returned back to the policy layer or inference engine layer
- Results are reevaluated based on a set of policies

Provenance Controller

- The provenance controller stores and manages provenance information associated with data items
- It can use a graph representation to store information in a logical graph structure
- The controller records ongoing activities associated with the data items in the data controller
- It takes a graph query as input and evaluates it over the provenance information
- The query evaluation returns a subgraph to the inference controller layer
- The subgraph is reexamined using a set of policies

APPLICATION TO SOCIAL MEDIA DATA

- Security policies in social media can be represented as RDF or in a language such as SWRL
- Inference controllers can be built on top of the access control architecture to handle unauthorized access via inference.
- Social media data is represented as RDF graphs and policies can be specified using combinations of SWRL, RDF, and OWL
- The policy engine examines the knowledge base and modifies queries posed against the RDF-based social media database
- The implementation of the inference controller for RDF-based social media data is similar to the discussion in previous sections

SECURITY RULE ENFORCEMENT

25 feb 23

Introduction

- Access control
  - protecting resources from unauthorized access
- Traditional access control mechanism
    - reference monitor evaluates access requests based on authorizations that grant or deny the request.
    - Role-Based Access Control
    - Mandatory Access Control
    - Discretionary Access Control
- SAKB ontology:
  - uses an SAKB ontology to store authorizations and prohibitions.
  - the SAKB ontology is used to represent knowledge about automobiles
    - hasMake, hasModel, and hasYear,
  - engines, and transmissions
    - has data properties such as hasCylinderCount, hasFuelType, hasHorsepower, and hasTorque
    - hasGearCount, hasTransmissionType, and hasTransmissionMode

Implementing the SAKB ontology

- The SAKB ontology is a framework for representing knowledge in the domain of software architecture
- Implementing the SAKB ontology involves several key steps
  - Identifying the domain of interest
  - Defining concepts and relationships
  - Creating a formal specification of the ontology
  - Integrating the ontology into your software application or knowledge management system
- Semantic Web language :OWL and RDF

how to INTEGRATE the SAKB ontology into a software application

- **Choose an API or library:**
  - Popular APIs for working with Semantic Web technologies include Jena, RDF4J, and Apache Fuseki.
- **Load the ontology:**
  - Reading the ontology file
  - Creating an in-memory representation of the ontology
- **Query the ontology**:
  - Retrieving all instances of a particular class
  - Finding all individuals related to a particular individual
  - Executing more complex queries using SPARQL or other query languages
- **Update the ontology**:
  - Modifying the in-memory representation of the ontology
  - Writing changes back to disk
  - Adding new individuals, modifying relationships between individuals, or adding new classes or properties to the ontology
- **Integrate with your application**:
  - Displaying ontology data to users
  - Using ontology data to make decisions within the application
  - Integrating the ontology with other software components

- SWRL rules
  - used to infer authorizations and prohibitions for access control and admin policies.
- Policy refinement
  - populate the SAKB with the inferred authorizations/prohibitions, by executing all the SWRL rules encoding security policie
- SAKB to retrieve the corresponding admin authorization
  - submitted SWRL can be evaluated
  - denies to the grantor the admin request

Admin Request Evaluation

- only a user with admin privileges can insert the SWRL into the system
  - if there is an admin authorization in the Semantic Automated Knowledge Base (SAKB) for the grantor
  - **Question:**Let's say we have an SAKB ontology that represents different types of software components and their dependencies. We want to use SWRL rules to infer which components are likely to cause performance issues if they are modified or removed from the system. We can define an SWRL rule like this
  - Component(?c) ∧ dependsOn(?c, ?d) ∧ hasProperty(?d, PerformanceCritical) → hasProperty(?c, PerformanceCritical)

- access control policy for the read privilege on a target object.
  - grantor's name
  - an access control or filtering policy encoded as an SWRL rule
  - <grantor, targetObject>:Read

- admin requests for access control authorizations in a knowledge base.
- framework verifies whether a particular user, Bob, is authorized to specify certain access control policies.
- SWRL (Semantic Web Rule Language) to represent the access control policies
- two examples of SWRL rules
- SWRL1 and SWRL2, which specify conditions that must be met in order for certain access control authorizations to be granted.

- SWRL1: Owns(Bob,?targetObject) ∧ Photo(?targetObject) ∧ Friend(Bob,?targetSubject) ⇒ Read(?targetSubject,?targetObject)
- if Bob owns the target object, the target object is a photo, and Bob is friends with the target subject, then the target subject is authorized to read the target object.
- To determine the result of the admin request, the framework has to verify the existence of :AdminRead instance in the SAKB.
- New_SWRL1:AdminRead(Bob,?targetObject) ∧ Owns(Bob,?targetObject) ∧ Photo(?targetObject) ∧ Friend(Bob,?targetSubject) ⇒ Read(?targetSubject, ?targetObject)

An administrator is submitting a filtering policy, and the system needs to determine whether the administrator is authorized to specify that policy.

- Goal
  - "PRead" instance (which is short for "permission read")
  - Bob must be the one who allowed the reading (this is the "Grantor" property)
  - "?controlled" must be the person who read it (this is the "Controlled" property)
  - "?targetObject" must be the video that was read (this is the "TargetObject" property).

Example

- If Person(?p) ∧ hasAge(?p,?age) ∧ swrlb:greaterThan(?age,18) ∧
- livesIn(?p,?city) ∧ isEmployed(?p) ⇒ EligibleForLoan(?p)

- This SWRL rule states that if a person is over 18 years old, lives in a certain city, and is employed, then they are eligible for a loan.

- We can modify this rule to make it more specific by adding additional conditions. For example, we might want to consider the person's credit score in determining their loan eligibility:

- If Person(?p) ∧ hasAge(?p,?age) ∧ swrlb:greaterThan(?age,18) ∧
- livesIn(?p,?city) ∧ isEmployed(?p) ∧ hasCreditScore(?p,?score) ∧
- swrlb:greaterThanOrEqual(?score, 700) ⇒ EligibleForLoan(?p)

- In this modified SWRL rule, we've added a condition that the person must have a credit score of at least 700 to be eligible for a loan. By adding this condition, we've made the rule more specific and increased the likelihood that only those who are most likely to repay the loan will be eligible.

- If a person is allergic to peanuts and eats food that contains peanuts, they may experience an allergic reaction. We can represent this relationship using an SWRL rule:

- AllergicTo(?person, Peanut) ∧ Contains(?food, Peanut) ∧
- EatenBy(?person, ?food) ⇒ AllergicReaction(?person)

AllergicTo(?person, Peanut) ∧ Contains(?food, Peanut) ∧
EatenBy(?person, ?food) ∧ Quantity(?food, ?amount) ∧ greaterThan(?amount, 10) ⇒
AllergicReaction(?person)

Access Request Evaluation

- The access request is represented as a triple (u, p, URI), where u is the user making the request, p is the privilege requested (e.g. read), and URI is the resource being accessed
- system needs to perform a check to make sure that the user is allowed to access the resource with the requested privilege

- Scenario: A company wants to enforce a security rule for their software system. The rule is that only users with an "Admin" role should be able to access certain sensitive data.

- Question: How can the SAKB ontology and SWRL rules be used to enforce this security rule?
- To implement this, the SAKB ontology can define a class for "Users" and a property "hasRole" to represent the roles of users. The ontology can also define a class for "SensitiveData" and a property "hasAccess" to represent access rights to sensitive data.

- The SWRL rule can be written as follows:

- If a user (?u) has a role (?r) of "Admin" and the sensitive data (?d) has access (?a) that requires an "Admin" role, then the user (?u) has access to the sensitive data (?d).

QUESTIONS:

1. You are the security architect of a company that uses a web application to store sensitive customer information. How can you use SAKB and SWRL to enforce security rules that prevent unauthorized access to this information?
2. A financial institution needs to comply with strict regulatory requirements for data privacy and security. How can SAKB and SWRL be used to enforce these requirements and ensure that customer data is protected?
3. An e-commerce company wants to implement a secure payment processing system that meets industry standards for security. How can SAKB and SWRL be used to enforce these standards and prevent fraud and other security threats?
4. A healthcare organization needs to ensure the privacy and security of patient data in compliance with HIPAA regulations. How can SAKB and SWRL be used to enforce these regulations and prevent unauthorized access to patient data?
5. A government agency needs to protect sensitive information related to national security. How can SAKB and SWRL be used to enforce security rules that prevent unauthorized access to this information, while still allowing authorized users to access the information they need?

Security Policy Specification

R Devika

AP II/CSE/SOC

OUTLINE

- Introduction
- Policy Evaluation
- Encoding Security Rules
- Adopting SWRL
- Representing Authorizations and Prohibition

Introduction

- security policies based on three main components
  - subject specification, object specification, and action specification
- subject specification
  - to identify the entity to which the security policy applies
    - e.g. (users or processes).
- object specification
  - to identify the resources to which the policy refers
    - (e.g. files, hardware resources, or relational tables).
- action specification
    - To specify the action
    - (e.g. read, write, execute or admin) that subjects can exercise on objects

Policy Evaluation

- by defining a set of authorizations
  - what rights each entity (e.g. users or processes) has over the resources being protected (e.g. files, hardware resources, or relational tables)
  - encoded as security rules
    - which consists of two parts - the antecedent and the consequent
    - The antecedent defines the conditions specified in the subject and object specifications of the policy,
    - the consequent represents the authorizations that result from these conditions.
    - Example 1:
      - Antecedent: If Subject A is a member of Group G and accesses File F
      - Consequent: Then Subject A is authorized to perform read and write actions on File F
    - Example 2:
      - Antecedent: If an employee (subject) tries to access a company document (object)
      - Consequent: Then the employee is authorized to read and write the document.

Encoding Security Rules:

- Antecedent: (User1 is a member of Group A) AND (Group A has access to Resource X)
- Consequent: User1 has read and write authorization on Resource X

Adopting SWRL (Semantic Web Rule Language)

- Antecedent: (User1 is a member of Group A) AND (Group A has access to Resource X)
  - as the body,
- Consequent: User1 has read and write authorization on Resource X
  - as the head
- For example
  - in SWRL syntax, the security rule could be represented as:
    - **Body: User(?u) hasAccessTo(?u, ?pi)**
    - **Head: Authorized(?u, view, ?pi)**
      - ?u represents a variable to access the personal information,
      - ?pi represents a variable representing the personal information,
      - User(?u) represents that ?u is a user,
      - hasAccessTo(?u, ?pi) represents the action of accessing the personal informationAuthorized(?u, view, ?pi) represents that the user is authorized to view the information.

SWRL to express security policies

- to encode authorizations and restrictions in an ontology
  - encoding process is called the Security Authorization Knowledge Base (SAKB)
  - Example
    - Consider an online social network, where users can connect with each other, post messages, and comment on other people's posts.

      - Antecedent: The user is the owner of the post
      - Consequent: The user is authorized to delete the post

      - Antecedent: The user is an administrator
      - Consequent: The user is authorized to delete any post

Example of encoding a security rule in a social network using SWRL (Semantic Web Rule Language)

- Antecedent: The user "John" is a member of the social network.
- Consequent: The user "John" is authorized to post on his own profile page
- Encoded rule:
- User(?u) ^ hasUsername(?u, "John") -> hasAuthorization(?u, postOnOwnProfile)

- Question: "All users must be 18 years or older to access adult content on the platform.“
- Ans1:
  - Antecedent: User(?u), hasAge(?u, ?age), swrlb:greaterThan(?age, 17)
  - Consequent: Authorized(?u, accessAdultContent)
- Ans2:
  - User(?x), hasAge(?x, ?y), swrlb:lessThan(?y, 18),
  - Prohibition(?x, accessExplicitContent)

Example of encoding a security rule(contd')

- Consider a security policy in which only registered users are allowed to access the user profile information. To encode this policy, the following SWRL rule can be used:
  - Antecedent: User(?u), RegisteredUser(?u)
  - Consequent: AuthorizedAccess(?u, UserProfileInformation)

Authorizations and Prohibitions

- In the ontology, the concept of "User" can be defined as a class, with individual instances being users of the social network (e.g. JohnDoe, JaneDoe).Another class, "Action", can be defined to represent the different actions that can be performed in the social network (e.g. "Post", "Comment", "View").The "Resource" class can be defined to represent the resources in the social network (e.g. "Profile", "Post", "Comment").
- The relationships between the classes can be defined using properties, such as "canPerformAction" (with domain "User" and range "Action"), and "hasAccessTo" (with domain "User" and range "Resource").
- With these classes and properties in place, Write the SWRL rules to encode the security policies of the social network.

Answer

- Rule 1: (User(?u) & canPerformAction(?u, Post) & hasAccessTo(?u, Profile)) -> canPerformAction(?u, Post). This rule states that a user "?u" can post on their own profile if they have the "Post" action and access to their own profile.
- Rule 2: (User(?u) & canPerformAction(?u, Comment) & hasAccessTo(?u, Post)) -> canPerformAction(?u, Comment). This rule states that a user "?u" can comment on a post if they have the "Comment" action and access to the post.
- These rules represent authorizations, but prohibitions can also be encoded in the same way by negating the antecedent or consequent. For example:
- Rule 3: (User(?u) & canPerformAction(?u, Post) & hasAccessTo(?u, Post(?p))) -> canPerformAction(?u, Post(?p)). This rule states that a user "?u" cannot post on another user's post if they do not have access to that post.

- Suppose we have a social network where users can post content, comment on posts, and send messages to each other. We want to enforce a security rule that prevents users from sending messages to other users who they are not connected with as friends or followers.
- User(?user) ∧ Message(?message) ∧ recipient(?message, ?recipient) ∧ notConnected(?user, ?recipient) → denyMessage(?message)

- Suppose we want to ensure that only friends of a user can see their personal information, such as their phone number or email address. We can use SWRL to encode this rule
- SocialNetwork(?user) ∧ hasPhoneNumber(?user, ?phone) ∧ hasEmailAddress(?user, ?email) ∧ ¬FriendOf(?user, ?friend) → hideInformation(?phone, ?email)

Possible questions

1. How can security policies be represented in the SAKB using ontologies?
2. What is the role of the antecedent and consequent in encoding security rules in the SAKB?
3. How can authorizations and prohibitions be represented in the SAKB using SWRL?
4. Can you explain the difference between encoding access control policies and filtering policies in the SAKB?
5. How can the SAKB be used to enforce security policies in an online social network?

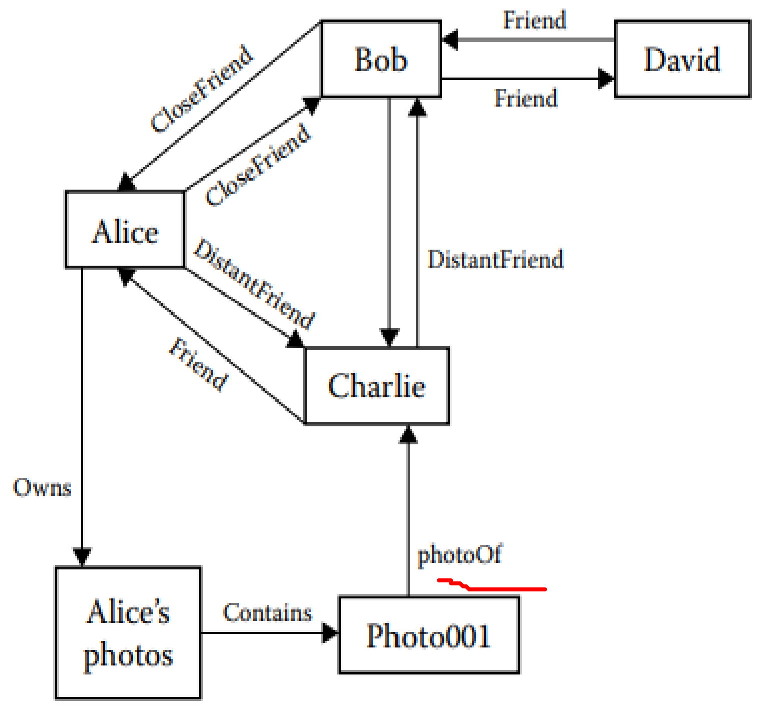
SECURITY POLICIES FOR OSN

R DEVIKA

APII/CSE/SOC

SECURITY POLICIES FOR OSN

Description of the small network used to illustrate the access control mechanism



- a PhotoAlbum uploaded by Alice.
- the need to revise traditional access control models and mechanisms for protecting resources in social networks.
- focusing only on access control policies
  - not taking into account the complexity of the social network scenario.

```
def __init__(self, name, age, email):
      self.name = name
      self.age = age
      self.email = email
      self.friends = []

   def add_friend(self, friend):
      self.friends.append(friend)
      friend.friends.append(self)

   def get_friends(self):
      return self.friends
```

```
# Create some users
alice = User("Alice", 25, "alice@example.com")
bob = User("Bob", 30, "bob@example.com")
charlie = User("Charlie", 27, "charlie@example.com")

# Add some friends
alice.add_friend(bob)
alice.add_friend(charlie)

# Get Alice's friends
alice_friends = alice.get_friends()
print(alice_friends)
```

users to create accounts, post messages, and view other users' messages:

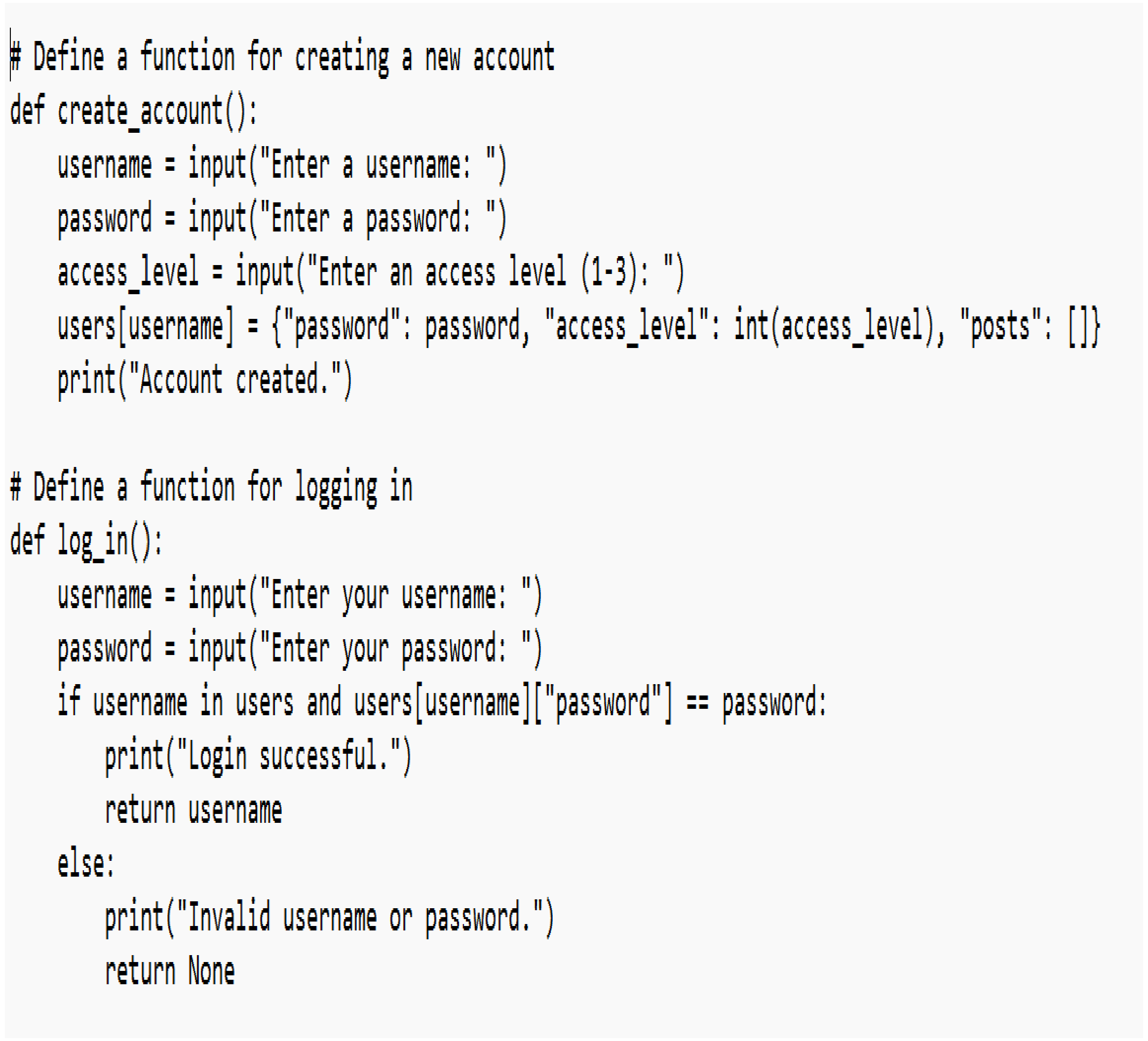
python

Describing a framework for managing access to resources in online social networks.

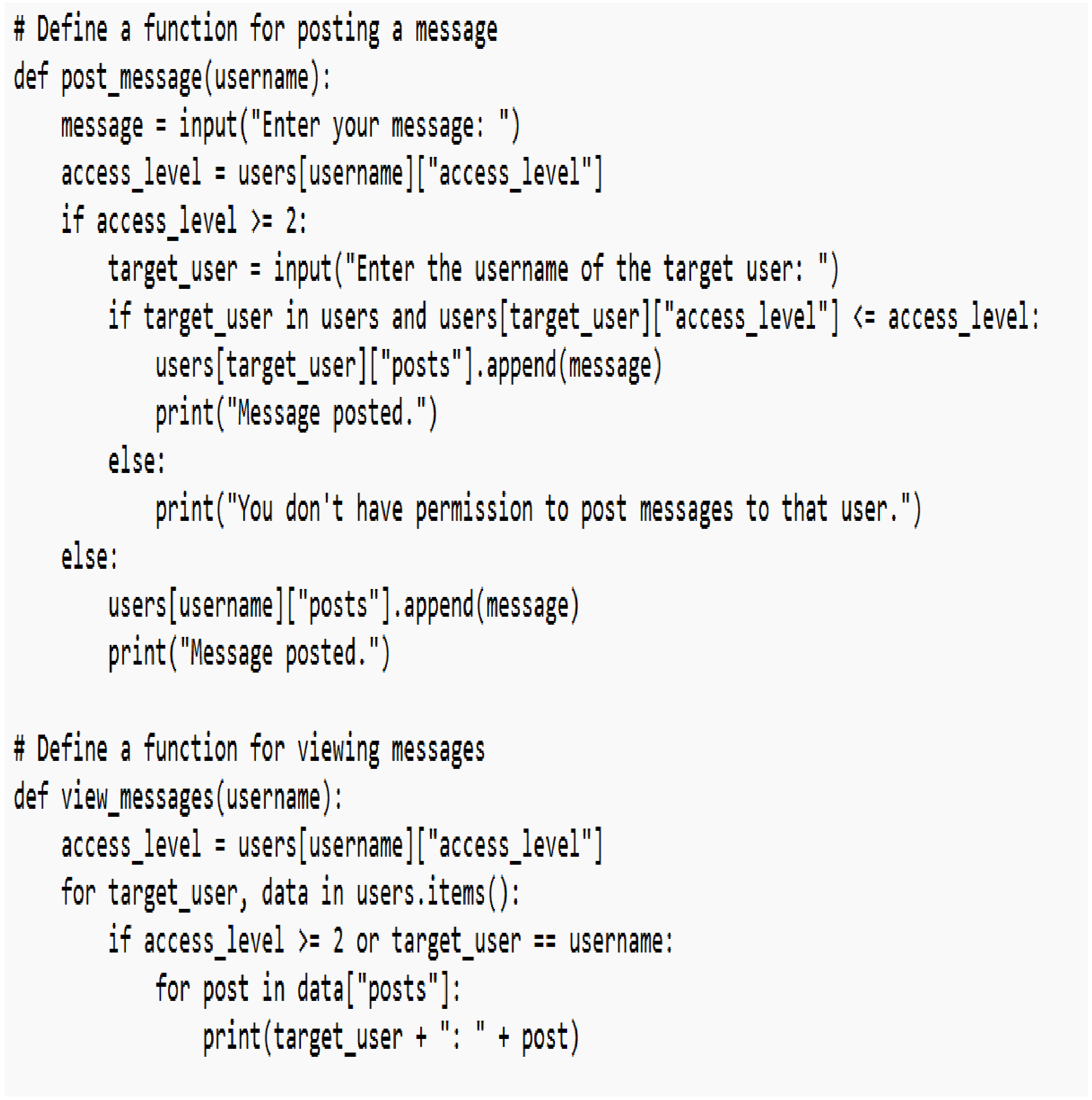
- based on previous research in the field, but includes some innovative changes.
- access control policies
  - relationships between different nodes in the network
  - take into account the level of trust between these relationships
  - policies
    - Ontologies
      - Representations of concepts such as resources, users, and actions.
  - defined at a more general level
    - rather than having to be specified for each individual resource and participant.
  - allows for the propagation of access control policies across different dimensions,
    - relationships and action

# Define a dictionary to store user accounts and their access levels

users = {}



```
# Define a function for creating a new account
def create_account():
    username = input("Enter a username: ")
    password = input("Enter a password: ")
    access_level = input("Enter an access level (1-3): ")
    users[username] = {"password": password, "access_level": int(access_level), "posts": []}
    print("Account created.")

# Define a function for logging in
def log_in():
    username = input("Enter your username: ")
    password = input("Enter your password: ")
    if username in users and users[username]["password"] == password:
        print("Login successful.")
        return username
    else:
        print("Invalid username or password.")
        return None
```



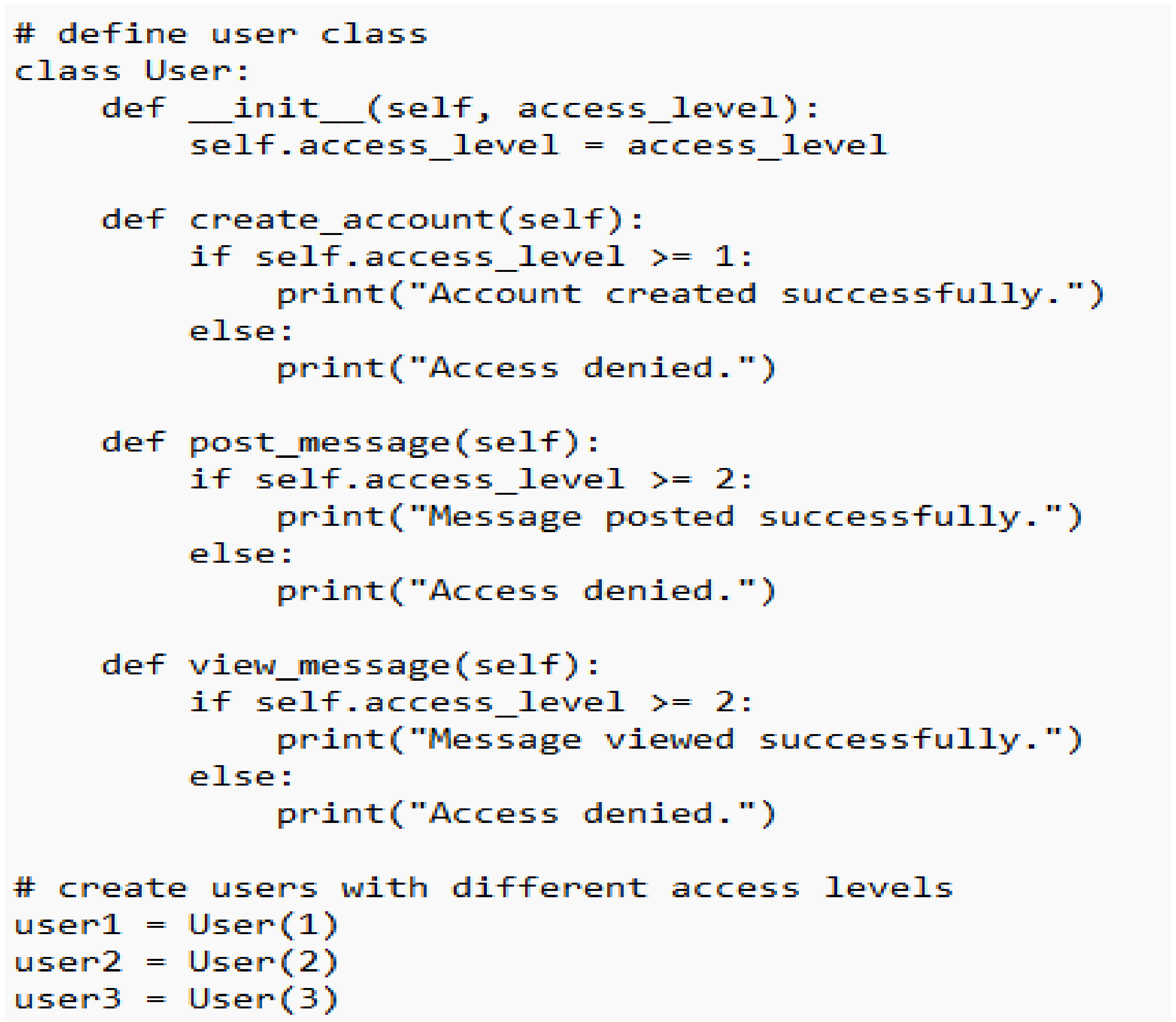
```
# Define a function for posting a message
def post_message(username):
    message = input("Enter your message: ")
    access_level = users[username]["access_level"]
    if access_level >= 2:
        target_user = input("Enter the username of the target user: ")
        if target_user in users and users[target_user]["access_level"] <= access_level:
            users[target_user]["posts"].append(message)
            print("Message posted.")
        else:
            print("You don't have permission to post messages to that user.")
    else:
        users[username]["posts"].append(message)
        print("Message posted.")

# Define a function for viewing messages
def view_messages(username):
    access_level = users[username]["access_level"]
    for target_user, data in users.items():
        if access_level >= 2 or target_user == username:
            for post in data["posts"]:
                print(target_user + ": " + post)
```

```
# Main loop
while True:
    print("1. Create account")
    print("2. Log in")
    print("3. Post message")
    print("4. View messages")
    print("5. Quit")
    choice = input("Enter your choice: ")
    if choice == "1":
        create_account()
    elif choice == "2":
        username = log_in()
    elif choice == "3":
        if username:
            post_message(username)
        else:
            print("You must be logged in to post a message.")
    elif choice == "4":
        if username:
            view_messages(username)
        else:
            print("You must be logged in to view messages.")
    elif choice == "5":
        break
    else:
        print("Invalid choice.")
```

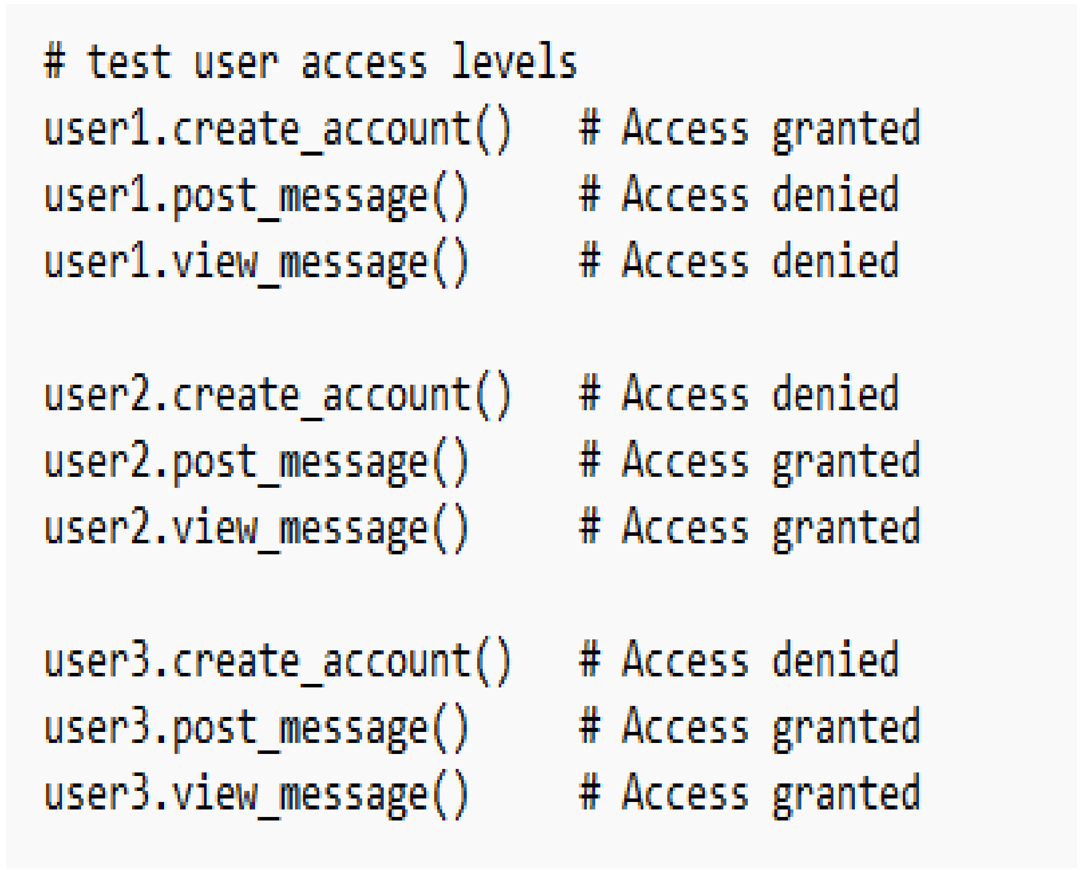
Filtering Policies

- An Online Social Network (OSN) is a platform where users can share various types of information.
- Some of the information shared on OSNs may be inappropriate for certain people.
- Filtering policies are introduced to determine what information should be filtered out.
- The filtering policies can be set by the user themselves or by a supervisor for a target user.
- The policies are defined using ontologies and can be applied to multiple users.
- Filtering policies control what information is filtered out when a user accesses the OSN, even if they have permission to view the resource.
- Filtering policies are different from access control policies, which control who has permission to access certain resources

- For example, a user with an access level of 1 will only be able to create accounts and log in, but will not be able to post messages or view messages. A user with an access level of 2 or 3 will be able to post and view messages, but not create accounts with an access



```
# define user class
class User:
    def __init__(self, access_level):
        self.access_level = access_level

    def create_account(self):
        if self.access_level >= 1:
            print("Account created successfully.")
        else:
            print("Access denied.")

    def post_message(self):
        if self.access_level >= 2:
            print("Message posted successfully.")
        else:
            print("Access denied.")

    def view_message(self):
        if self.access_level >= 2:
            print("Message viewed successfully.")
        else:
            print("Access denied.")

# create users with different access levels
user1 = User(1)
user2 = User(2)
user3 = User(3)
```



```
# test user access levels
user1.create_account()   # Access granted
user1.post_message()     # Access denied
user1.view_message()     # Access denied

user2.create_account()   # Access denied
user2.post_message()     # Access granted
user2.view_message()     # Access granted

user3.create_account()   # Access denied
user3.post_message()     # Access granted
user3.view_message()     # Access granted
```

Filtering policies are different from access control policies

ACCESS CONTROL POLICIES

- who can access a resource
  - such as specific posts, profiles, or groups
- For example
  - an access control policy might state that only users over the age of 18 can access certain

FILTERING POLICIES

- what information is filtered out
- when a user accesses the OSN
  - regardless of their authorization status.
- Example
  - all posts containing explicit language or images should be filtered out

Assignment questions:

Design a social networking application that allows users to set access control and filtering policies for their personal data and content. The application should include features such as:

- User authentication and authorization to ensure only authorized users can access protected data.
- Granular access control to allow users to specify who can view and interact with their content.
- Filtering policies to prevent unwanted or harmful content from being displayed to users.
- Notifications and alerts to inform users of any attempts to access their data or content without authorization.

Your program should be able to handle large amounts of data and provide fast and efficient access control and filtering policies. It should also be secure and protect user data from unauthorized access or theft

Some examples of administrative policies in an OSN setting

- a set of rules and regulations that dictate how access control and filtering policies are specified and enforced in the network
- purpose of an admin policy is to ensure that the security of the network and its users
  - **User conduct policy** it may prohibit hate speech, harassment, or sharing illegal content.
  - **Privacy policy** what information the OSN collects, how it's used, and who has access to it.
  - **Content moderation policy** it may explain how the OSN deals with spam, false information, or adult content
  - **advertising practices** are prohibited, and how the OSN ensures that advertising is safe and relevant to users.

User Conduct Policy in Social Network

- # User Conduct Policy in Social Network

- # Define the guidelines for user conduct

❑ guidelines = {

❑ 'Respect others': 'We ask that you treat all other users with respect and courtesy. This means no bullying, hate speech, or discrimination of any kind.',

❑ 'Keep it legal': 'Do not engage in any illegal activity on our platform. This includes but is not limited to sharing illegal content, conducting illegal transactions, or promoting illegal activities.',

❑ 'No spamming': 'Do not spam other users with unwanted messages or content.',

❑ 'Protect your privacy and the privacy of others': 'Do not share personal information about yourself or others, including contact information, addresses, or other identifying information.',

❑ 'Keep it clean': 'Do not post or share any explicit or offensive content on our platform.',

❑ 'Follow the rules': 'Follow any additional rules or guidelines that may be posted on our platform.'

- }

- # Display the guidelines to the user
- ❑print('Welcome to our social network.')
- ❑print('We want to create a safe and respectful community for all of our users, and we ask that you follow these guidelines when using our platform:\n')
- ❑for key, value in guidelines.items():
- ❑   print(key + ': ' + value)

- # Define consequences for violating the guidelines
- ❑consequences = 'Failure to adhere to these guidelines may result in account suspension or termination. We reserve the right to remove any content or take any other action necessary to ensure a safe and respectful community for all users.'

- # Display the consequences to the user
- ❑print('\n' + consequences)

# Privacy Policy for Social Network

```
# Define the privacy policy
privacy_policy = """
We are committed to protecting your privacy and ensuring the security of your personal
information. This Privacy Policy explains how we collect, use, and protect your personal
information on our social network.

1. Information We Collect
We collect information that you provide to us when you create an account, update your
profile, or use our services. This may include your name, email address, profile picture,
and other information that you choose to provide.

2. How We Use Your Information
We use your information to provide you with a personalized experience on our social
network. We may use your information to communicate with you, to improve our
services, and to ensure the security of our platform.
```

```
3. How We Share Your Information

We do not sell or rent your personal information to third parties. However, we may share your information with trusted third-party service providers who help us provide our services. We may also share your information to comply with legal obligations or protect our rights.


4. Your Choices

You have the right to access, modify, or delete your personal information at any time. You may also choose to opt out of receiving promotional emails from us.


5. Security of Your Information

We take reasonable steps to protect your personal information from unauthorized access or disclosure. However, no security measures can provide absolute protection. We cannot guarantee the security of your information.


6. Changes to Our Privacy Policy

We may update our Privacy Policy from time to time. We will notify you of any changes by posting the updated Privacy Policy on our social network.


7. Contact Us

If you have any questions or concerns about our Privacy Policy, please contact us at [contact email].
"""


# Display the privacy policy to the user

print(privacy_policy)
```

Data mining techniques can be applied to identify potential security threats, analyze user behavior, and detect abnormal activities on social networks.

- Anomaly detection:
  - identify patterns that deviate from the norm, and can be used to detect suspicious user behavior
    - detecting unusual login times or IP addresses,
    - identifying a sudden increase in the number of friend requests.
- Text mining:
  - to analyze the content of user posts and messages, and to identify potential security
    - cyberbullying, harassment, or hate speech
      - to detect phishing scams, spam, and other fraudulent activities.
- Sentiment analysis:
  - to analyze the tone and emotion of user posts and messages,
    - threats of violence or self-harm.
      - to detect changes in user behavior, such as sudden shifts in mood or sentiment.
- Social network analysis:
  - to analyze the structure of social networks,
  - to identify key influencers and their potential impact on the network.
    - the spread of misinformation, rumors, or propaganda
- Link Analysis
  - to analyze the connections between different users on the social network.
  - dentifying users who are connected to known security risks or who are sharing suspicious conte.

Question

- You have been hired as a consultant by a new social network called "Social Sphere" to develop a privacy policy for their platform. The platform allows users to connect with friends, post updates, photos, and videos, and join interest groups. The company's main revenue stream will come from targeted advertising based on user data.

  Your task is to draft a privacy policy that outlines what data the company will collect from users, how that data will be used, and how the company will protect users' privacy. You should consider the following issues:

1. What types of personal data will Social Sphere collect from users, and how will that data be used?
2. Will Social Sphere share users' data with third parties? If so, who will they share it with, and for what purpose?
3. How will Social Sphere protect users' data from unauthorized access or use?
4. Will users be able to control how their data is collected and used? If so, how?
5. How will Social Sphere notify users in the event of a data breach?

- Write a privacy policy that addresses each of these issues. Your privacy policy should be clear, concise, and easy for users to understand

- RNN for sentiment analysis in user-generated content on online social networks (OSNs) using Python and Keras
- code

- Code for Anomaly detection

Unit II

MODELING SOCIAL NETWORKS USING SEMANTIC WEB TECHNOLOGIES

R DEVIKA

APII/CSE/SOC

Type of Relationships

Five categories of social network data

(i) personal information

(ii) personal relationships,

(iii) social network resources,

(iv) relationships between users and resources, and

(v) actions that can be performed in a social network.

Modelling Personal Information

- Facebook can be modelled
  - using the FOAF ontology (Brickley and Miller, 2007).
    - an OWL-based format for representing personal information and an individual's social network.
    - various classes and properties to describe social network data
      - basic personal information, online account, projects, groups, documents, and images
  - no FOAF construct to capture the meaning for lookingFor
    - John Smith is looking for friendship
  - extensibility of the RDF/OWL language
    - easily solvable the meaning lookingFor

Example

Facebook profile ID 999999 by using a new Facebook ontology written in the RDF/OWL language

```
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns>
@prefix foaf: <http://xmlns.com/foaf/0.1/>
@prefix fb: <http://example.org/facebook>
<http://www.facebook.com/profile.php?id = 999999999>
foaf:name "John Smith"
<http://www.facebook.com/profile.php?id = 999999999>
fb:lookingFor "Friendship"
```

Existing ontologies such as FOAF could be easily extended to capture personal information available in OSNs

Modeling Personal Relationships

OSNs do not support

- not provide fine-grained connection definitions.
- you attended school or work with a friend
  - offers no way to express what that truly means i.e strength of the relationship.

- reification-based methodology for determining the strength of a relationship
  - Relationships would be modelled using a set of four RDF statements
  - SWRL would be unable to understand these relationships.
- a specification-recommended pattern
  - the n-ary relation pattern in order to adhere to W3C specifications
-

To model personal relationships using n-ary relation pattern

- . FriendshipRelation class
  - subclasses that denote a general strength of friendship
  - Family,
  - CloseFriend,
    - bestFriend
  - and DistantFriend,
- relationship types have no predefined meanings
  - the level of trust between the initiator and his friend
  - **TrustValue** data property,

Note

- unidirectional relationship in the social network
  - Single instance of class
- bidirectional nature of social network relations
  - Two instances of class

Modeling Resources

- OSN provides some resources
  - albums or walls
    - share information among individuals
  - Resource class
    - PhotoAlbum
      - a name and a description –data property
      - containsPhoto object property links into Photo
    - Photo
      - a name, a caption, and a path to the stored location of the file
    - Message
      - a sender, a receiver, a subject, a message, and a time stamp.
      - WallMessage
        - additional restrictions
          - only be sent to a single individual.

Modeling User/Resource Relationships

- Current applications such as Facebook only recognize ownership as the relationship between users and resources.
- This is not enough when it comes to control over access.
- A photo that contains two people, Smith and Jane Doe, is used as an example.
- Traditionally, Jane would have complete control over the photo.
- In this new model, Smith may also have some control over who can see the photo.
- Two classes are introduced to represent a photo album: Photo class and PhotoAlbum class.
- The Photo class has a name, caption, and location of the file.
- The PhotoAlbum class has a name and description.
- The two classes are linked through relationships such as containsPhoto and ownsAlbum.
- This same approach can be used to represent other relationships between users and resources.

Modeling Actions

- actions play a big role in user participation in a social network
- Key point: Actions play a crucial role in user participation in social networks
- Definition of actions: Properties that connect users, resources, and actions
- Hierarchies for actions: One action can be a subtype of another action
  - Example of action hierarchy: Read, Write, and Delete with Delete being a subtype of Write and Write being a subtype of Read
- Benefit of action hierarchy: If someone has authorization to delete, they also have authorization to write and read
- Extension of traditional access restrictions: New actions can be added, such as the action Post being a subtype of Write
- Consequence of allowing Post action: User can send private messages, write on wall, but cannot delete anything