A Tool for Negotiating Privacy Constraints in Online Social Networks

Dilara Keküllüoğlu[1] and **Nadin Kökciyan**[2] and **Pınar Yolum**[3]

1 Introduction

The content shared over online social networks increases rapidly. People share photos, opinions and videos of themselves. Additionally, people can share things about other people by tagging them or mentioning them. Even though users have control over what they share, they cannot manage what people have shared about them. The shared content may reveal information about the user, which the user might not wanted to share herself. This creates a privacy breach on the user's side. In current online social networks, a common way to deal with this is for the user to complain to the social network administration and ask the content to be removed. However, by the time the content is removed (if at all), many people might have seen it already. Ideally, it would be best if such a content was not shared in the first place.

In order to facilitate this, we develop a system where users are represented by software agents. Each agent is responsible for keeping track of its user's privacy rules and communicating with other agents. Before a content is being published, relevant agents communicate among themselves in form of negotiation to reach a consensus about how a post should be published. The negotiation takes place between the initiator, the person who prepares the post, and the negotiator, the person who is included in the post. We represent the privacy domain and the privacy rules semantically by the use of ontologies. Moreover, the decision making is done using utility functions. We have developed multiple negotiation strategies, which are utility based. This demonstration will mainly show how our negotiation system works as well as show how different, realistic scenarios are handled with different negotiation strategies so that privacy violations are avoided.

2 Negotiation Architecture

Our proposed negotiation architecture is based on semantic representation of negotiation concepts and privacy rules, but enables each agent to use its own utility functions to evaluate negotiation offers. We use PriNego [3] as the basis for the semantic aspects of negotiation. PriNego proposes a negotiation framework for privacy where each agent represents a user in the social network. Each agent is aware of the privacy concerns of its user but also has information about the social network, such as the friends of the user. This information is captured in an ontology that is represented in Web Ontology Language (OWL).

An agent captures its user's privacy constraints as semantic rules (privacy rules), which are represented with Semantic Web Rule Language (SWRL) rules [2]. A privacy rule describes a situation wherein an agent would reject a particular negotiation offer. Consider the scenario in Example 1, which we will use as our running example in the demonstration.

Example 1. Bob wants to share a picture of Alice with everyone. This picture is in Eat & Drink context. Alice does not want her colleagues to see her leisure pictures. Moreover, she does not want Errol to see any of her pictures.

Following the above example, when Bob initiates a negotiation with Alice, Alice evaluates Bob's post request according to her rules and decides whether to accept or to make a counter offer. This is followed by a similar move from Bob. That is, the negotiation continues in a turn-taking fashion.

A user might have various privacy constraints but these might not be equally important. To capture the fact that a rule is more important than a second rule, we associate a weight with each rule.

The evaluations done to decide whether to accept a negotiation offer as well as to create a new counter-offer constitute the *negotiation strategy* of an agent. Here, we require each agent to have a utility function that it can use to make this decision. The agent that initiates the negotiation (i.e., initiator) will have a different utility function than an agent that negotiates for her privacy (i.e., negotiator).

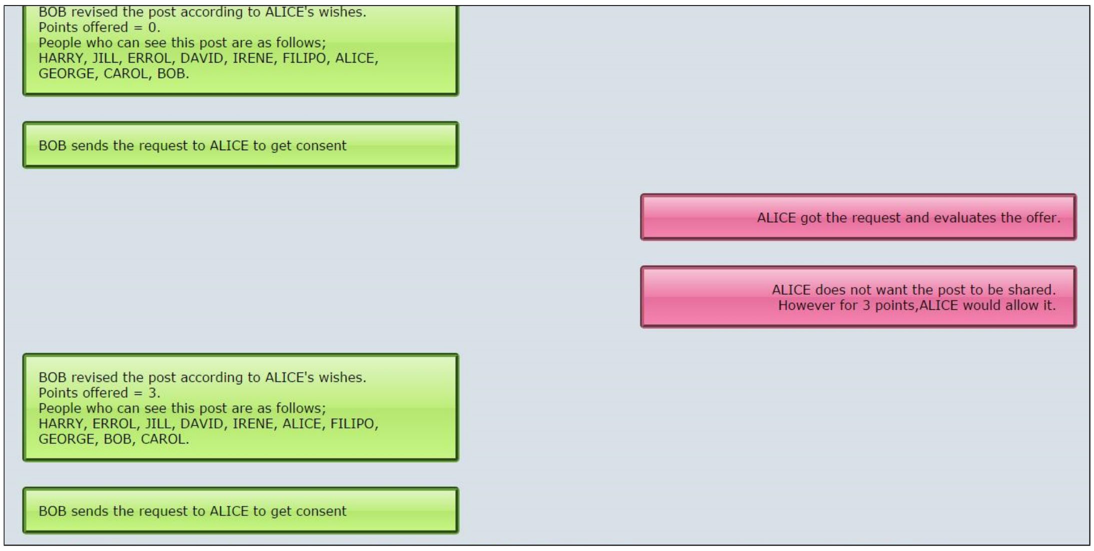
3 Demo Details

Our system is implemented as a Java-based web application deployed using the Tomcat server. In this system, agents communicate with each other through RESTful web services in Spring framework. These web services enable agents to start a negotiation, evaluate the incoming requests and update points, if necessary. In the front end, we use JSP and HTML to output the results of a negotiation.

Each agent has an ontology where they keep their social network structure and privacy rules. These ontologies are kept in OWL format, and we use OWL API [1] to work with ontologies in Java. The reasoning is done using Pellet reasoner [4]. An agent evaluates a post request by adding it to its ontology and using Pellet reasoner to infer a rejection if one exists.

We have developed three negotiation strategies that agents can use. This demonstration will show how all three negotiation strategies work on three different real-life examples.

[1] Bogazici University, Turkey, email: dilara.kekulluoglu@boun.edu.tr
[2] Bogazici University, Turkey, email: nadin.kokciyan@boun.edu.tr
[3] Bogazici University, Turkey, email: pinar.yolum@boun.edu.tr



BOB revised the post according to ALICE's wishes.
Points offered = 0.
People who can see this post are as follows;
HARRY, JILL, ERROL, DAVID, IRENE, FILIPO, ALICE, GEORGE, CAROL, BOB.

BOB sends the request to ALICE to get consent

ALICE got the request and evaluates the offer.

ALICE does not want the post to be shared.
However for 3 points,ALICE would allow it.

BOB revised the post according to ALICE's wishes.
Points offered = 3.
People who can see this post are as follows;
HARRY, ERROL, JILL, DAVID, IRENE, ALICE, FILIPO, GEORGE, BOB, CAROL.

BOB sends the request to ALICE to get consent

Figure 1. Negotiation steps in Example 1 when agents use Reciprocal Strategy (RP).

However, it is easy to add scenarios according to the viewers' wishes.

We run examples with all three strategies one by one in the system. The system outputs messages about the current situation of the ongoing negotiation, and we have created a simple web application that displays these messages. We will show how the negotiation takes place and compare between the outcomes when different strategies are used.

One of our proposed strategies is **Reciprocal Strategy (RP)**. This strategy is based on the idea that on every individual post, everyone's privacy cannot be preserved. However, if over many interactions, users can preserve their privacy largely, this is still acceptable. That is, on a single negotiation, the outcome of a negotiation is beneficial for all the negotiating agents; however one party is usually better than the others. This difference might be insignificant for many negotiations. The difference may get disadvantageous for the others if one party is favored most of the times. In this strategy, if one party is favored more in previous negotiations, then in the following negotiation she is forced to concede. To keep track of the previous negotiations, we use a point-based system where both parties have the same amount of points in the initial state (e.g., each 5pts). For every negotiation, agents exchange points depending on who is the initiator and how much benefit they get from that negotiation.

Figure 1 depicts a sample flow for our running example when agents use Reciprocal Strategy. The steps are as follows:

1. Bob creates the post request and sends this post request to Alice since she is tagged in the picture.
2. Alice takes this post request, and checks whether it conforms to her privacy concerns. There are three people (David, Irene and Errol) that she wants to remove from the audience. Hence, she puts these people in order of importance, and sends it to Bob.
3. Bob keeps the list of rejected people by Alice for revising the post request if necessary. He sends the same post request but with a point offer of 0.
4. Alice gets the post request and evaluates the post request by computing her utility. She does not accept it and asks Bob to give 3 points for her to accept the request as it is.
5. Bob gets the offer of Alice and calculates whether the new utility is above threshold if he gives 3 points to Alice. Bob sees that Alice's offer is acceptable so he sends the post request to Alice with the point offer given by Alice for confirmation.
6. Alice gets the post request and the corresponding point offer. She sees that the post request has not changed, and the point offer is what she has offered in the previous iteration. Alice agrees on sharing the content.
7. Bob shares the post request since they reach an agreement. Bob gives 3 points to Alice as promised.

Our demo video is available online at: `https://youtu.be/RGg9ReRC8SY`.

ACKNOWLEDGEMENTS

This work has been supported by TUBITAK.

REFERENCES

[1] Matthew Horridge and Sean Bechhofer, 'The OWL API: A Java API for OWL ontologies', *Semantic Web*, **2**(1), 11–21, (2011).

[2] Ian Horrocks, Peter F Patel-Schneider, Harold Boley, Said Tabet, Benjamin Grosof, Mike Dean, et al., 'SWRL: A semantic web rule language combining OWL and RuleML', *World Wide Web Consortium Member submission*, **21**, 79, (2004).

[3] Yavuz Mester, Nadin Kökciyan, and Pınar Yolum, 'Negotiating privacy constraints in online social networks', in *Advances in Social Computing and Multiagent Systems*, eds., Fernando Koch, Christian Guttmann, and Didac Busquets, volume 541 of *Communications in Computer and Information Science*, 112–129, Springer International Publishing, (2015).

[4] Evren Sirin, Bijan Parsia, Bernardo Cuenca Grau, Aditya Kalyanpur, and Yarden Katz, 'Pellet: A practical OWL-DL reasoner', *Web Semantics: Science, Services and Agents on the World Wide Web*, **5**(2), 51–53, (2007).