Grant Agreement No.: 823783
Call: H2020-FETPROACT-2018-2020
Topic: H2020-FETPROACT-2018-01
Type of action: RIA

WENET
INTERNET OF US

D3.1 Basic Social Relation Learning Module

Work package	WP 3
Task	Task 3.1
Due date	30/04/2020
Submission date	30/04/2020
Deliverable lead	OUC
Version	1.1
Authors	Isidoros Perikos (OUC) Vasileios Markos (OUC) Loizos Michael (OUC)
Reviewers	Nardine Osman (CSIC)

Abstract	This deliverable describes the functionality of the Social Context Builder module of the WeNet platform and presents the components it consists of. Specifically, it presents the approach that we have adopted for specifying three aspects of the social profile of a user: the social relationships between the user and other users, the user's preferences on how volunteers for a task initiated by the user should be presented to the user, and personalized explanations for aiding the user determine which among the volunteers for a task initiated by the user would be more appropriate to deal with that task.
Keywords	social relationships, social preferences, social explanations

Document Revision History

Version	Date	Description of change	List of contributor(s)
v0.1	24/01/2020	Initial structure	Isidoros Perikos (OUC) Vasileios Markos (OUC)
v0.2	03/02/2020	Addition of first content	Isidoros Perikos (OUC) Vasileios Markos (OUC)
v0.5	08/03/2020	First draft version	Isidoros Perikos (OUC) Vasileios Markos (OUC)
v1.0	15/04/2020	Pre-final version sent for internal review	Isidoros Perikos (OUC) Vasileios Markos (OUC) Loizos Michael (OUC)
v1.1	29/04/2020	Final version	Isidoros Perikos (OUC) Vasileios Markos (OUC) Loizos Michael (OUC)

Project co-funded by the European Commission in the H2020 Programme		
Nature of the deliverable:	**R**	
Dissemination Level		
PU	Public, fully open, e.g. web	✔
CL	Classified, information as referred to in Commission Decision 2001/844/EC	
CO	Confidential to WeNet project and Commission Services	

** R: Document, report (excluding the periodic and final reports)*

DEM: Demonstrator, pilot, prototype, plan designs

DEC: Websites, patents filing, press & media actions, videos, etc.

OTHER: Software, technical diagram, etc.

Co-funded by the Horizon 2020 Framework Programme of the European Union

EXECUTIVE SUMMARY

WeNet aims to design and develop a sociotechnical platform that allows people to connect through a machine-mediated process, and complete everyday tasks while respecting their individual differences, and embodying fundamental features of transparency and privacy.

The Social Context Builder module constitutes a core part of the WeNet platform: it is the module responsible for building and maintaining the social part of the profile of the users by leveraging the data collected by the various streams connected through the WeNet platform. Thus, this social part of each user's profile is continuously updated as a result of the user's interaction within the WeNet platform. With that key consideration in mind, in the first iteration for designing and developing the Social Context Builder, we have focused on the equally important consideration of the cold-start problem: the need for a user's social profile to be initiated in some meaningful manner from the moment that a user joins the WeNet platform, even before having access to WeNet interaction data, so that it can immediately be utilized to provide the user with meaningful suggestion and to enhance the user's WeNet experience. Accordingly, this deliverable presents our approach to designing and developing components that handle the online and continuous learning desiderata, but focuses more on how those components can utilize prior information or knowledge to solve the cold-start problem.

To account for the diverse way that users may choose to express their social relations, we have opted to design three components for the Social Context Builder: one that captures the user's social relationships in the typical network structure; one that captures the user's social preferences by ranking volunteers for a user-initiated task; and one that captures the user's social explanations by associating each volunteer for a user-initiated task with an explanation or an argument that is personal to the user, and that aids the user in selecting which among the volunteers to engage with to tackle the task that the user has requested help with.

The purpose of this deliverable is to present these three components and discuss their initial functionality. The deliverable also includes a case study that demonstrates the workflow of the interaction with the components in an example scenario on social eating, which has been discussed extensively by the consortium as a potentially prototypical scenario for the first iteration of the WeNet platform, while encompassing issues of privacy and diversity.

As anticipated in WP3, these three components will keep being revised and refined through the lifetime of the project, following an agile methodology, and guided by the data that will be gathered through the use of the WeNet platform in pilot studies. Each subsequent iteration of the modules will bring into the picture additional features, including the handling of diversity among users, the dealing with missing data, the balancing of privacy and transparency, etc. For this first iteration, we have, as planned, focused on the development of the basis of the components in a manner that accounts for their future refinement, and on the provision of cold-start solutions, given that, as of now, no WeNet interaction data are still available.

Co-funded by the Horizon 2020 Framework Programme of the European Union

TABLE OF CONTENTS

Co-funded by the Horizon 2020 Framework Programme of the European Union

LIST OF FIGURES

Co-funded by the Horizon 2020 Framework Programme of the European Union

LIST OF TABLES

Co-funded by the Horizon 2020 Framework Programme of the European Union

ABBREVIATIONS

DoA	Day of Arrival
KB	Knowledge Base
ML	Machine Learning
SCB	Social Context Builder
WP	Work-package

Co-funded by the Horizon 2020 Framework Programme of the European Union

1. INTRODUCTION

WeNet aims to design and develop a sociotechnical platform that allows people to connect through a machine-mediated process, and complete everyday tasks while respecting their individual differences, and embodying fundamental features of transparency and privacy.

A key aspect of the WeNet platform is the design and the development of the Social Context Builder module, which, according to WP3, is expected to comprise diversity-aware algorithms that learn social relations between users from streaming data, with the following features:

- Continuous monitoring of social interactions and adaptation.
- Accommodate diversity across users and social interactions.
- Cope with missing information, and balance a user's need for privacy and transparency in the decisions that affect them.
- Fair in identifying social relations, not social stereotypes.

The module is expected to be designed and developed throughout the lifetime of the project, across three sequentially-ordered tasks T3.1, T3.2, T3.3 (and with progress reported in the three corresponding deliverables D3.1, D3.2, D3.3) following an agile methodology. The first task, whose progress is reported in the current deliverable, aims to develop a basic social relation learning module, prior to having access to any data from the deployment of the WeNet platform. After such data are gathered, the module will be expanded to account for diversity issues across the platform's users, and will be subsequently further refined for the last iteration of its design and development, to be used with the final version of the platform.

As part of task T3.1, as reported in the current deliverable, we have focused on designing and implementing the first version of the module. Prior to the design and development of this first version of the module, we have explored the facets of the social profile of a user so that our design choices will be amenable to support features of subsequent module iterations.

In particular, in determining how to represent, reason with, and learn the social part of a profile's user, we have considered three key requirements:

- Diverse Expressions: Not every user is able to express or wishes to express their social relations in the same manner, for any number of reasons. The social profile should be able to capture diverse ways for users to express their social relations.
- Technical Desiderata: Any representation for the social relations of a user should be amenable to learning algorithms that are online and adaptive and can cope with missing information.
- Human Awareness: The encoding of social relations should support direct ways to provide transparency on how they are used to make recommendations, while respecting privacy / ethics.

Based on these requirements, we have designed and developed three components for capturing social relations. The remainder of this deliverable presents the design and development of algorithms for each of those components of the Social Context Builder. Accounting for the fact that no WeNet interaction data are still available, we have accordingly focused our work mostly on the cold-start problem, identifying solution on how the various components of the social profile of a user can be populated prior to the interaction of the user with other users within the WeNet platform, so that meaningful recommendations can be made to the user as soon as they join the platform. At the same time, our solutions were developed in a manner that supports the refinement of the social profile of the user after their

Co-funded by the Horizon 2020 Framework Programme of the European Union

joining the platform, and as soon interaction data with other users are available.

This document is divided into 7 chapters. Chapter 2 presents the social context builder and analyses its main functionalities with respect to the components for social relationships, social preferences, and social explanations. Chapter 3 presents the methodology for the specification of the users' social relationships as well as the calculation of the tie strength among them. Chapter 4 presents the methodology for the social preferences of the volunteers with respect to a task that a user set. Chapter 5 presents the methodology for providing explanations to the user about the selection of each volunteer. Following the description of the three components, Chapter 6 describes a case study on a social eating scenario that illustrates the workflow and the functionality of the components. Chapter 7 concludes the deliverable and outlines the next steps towards deliverable D3.2.

2. SOCIAL CONTEXT BUILDER

2.1 OVERALL FUNCTIONALITY

The Social Context Builder is responsible for building and maintaining the social details of a user profile, by leveraging the data collected by the various streams connected through the WeNet Commons APIs and by analyzing them.

The Social Context of a user will be represented as in the standard way of major social networks. There, a user follows some other users ("friends"), which can be either individual, or groups (companies, interest groups, clubs, etc.). Similarly, the user is followed back by other users. Thus, the Social Context is represented as a directed graph between users, which is the standard practice in social network analysis [1] .

The most important factor for us is being able to access the strength of a relationship. This is captured by the notion of "tie strength" [2] . Social relations are characterized as strong or weak ties, where strong ties indicate close relationships, and weak ties simpler acquaintances. Tie strength has been found to be dependent on various variables, such as time spent together, intimacy, reciprocal services, and others. An important study of these in the context of social networks can be found in [3] , where the authors attempt to measure those variables from Facebook data, and then access their predictive power for tie strength.

Finding tie strength and user compatibility is an active area of research, and many ideas have been proposed in the literature since then. Recently, machine learning-based methods have been employed for this task in all major social networks. We intend to investigate similar approaches in the context of WeNet's Social Context Builder. Additionally, another important factor is the ability to estimate the compatibility of two users who are not friends yet. Finally, balancing social ties and task experience is another challenging aspect, where further investigation is necessary. For example, social strength and trust is very important in the task of babysitting, while task experience might be more important in other cases.

Tie strength is likely to play a very significant role in determining the trust level of a social relationship, and thus the likelihood of that relationship to be used to disseminate information and to provide assistance with respect to some task [15] .

Another innovative functionality concerns the ability to explain the decisions made by the builder to the user. For example, when the Social Context Builder ranks users for a task, it is

Co-funded by the Horizon 2020 Framework Programme of the European Union

important to be able to explain to the user why the Social Context Builder proposes that particular user for the task.

Having models able to explain their decisions is a very active area of research in Machine Learning and Artificial Intelligence, and many approaches are being suggested. Some examples can be found in [4] for knowledge-based systems, or [5] for (statistical) Machine Learning models. In our work, we follow a knowledge-based approach that relies on rules (as explained in the following chapters). To summarize, the main functionalities of the components are the following:

i) Identification of the social relationships among users and the specification of the tie strength of their friendship. The relationships define the main type of relationship that people have (e.g. family members) while the weight of their social friendship (tie strength) derives their mutual interaction and interconnection.
ii) Ranking of the users with regards to a task that a user set
iii) Provision of the social explanations about the inclusion of volunteers (e.g., for a Social Eating task, explain that a volunteer was asked because they have excellent cooking skills). Such information is meant for both the task creator and for the identified volunteers as it allows the motivation and explains the reasons why a match was identified.

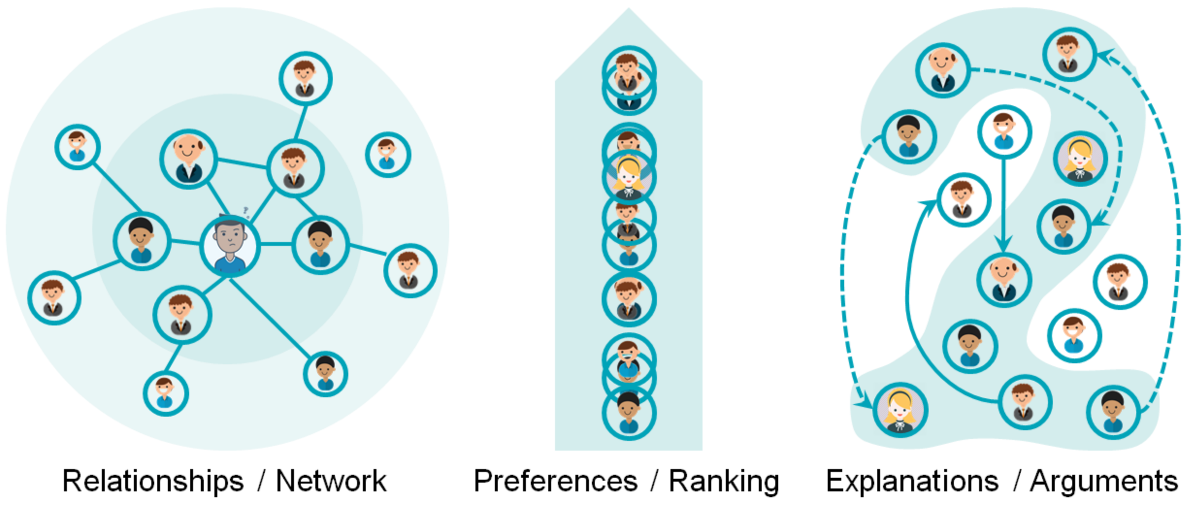


Figure 1: Illustration of social learning processes and representations

Our work was focused on the analysis of the functionality of the algorithms and the characteristics of the inference mechanisms that are to be used for reasoning about social relations. For this purpose, the aim of the learning algorithms was analyzed and their workflow was specified. Also, the interchange of information among the discrete mechanisms was designed with the aim to facilitate the concrete interconnection of the modules and the smooth interchange of data and information among them.

The Social relationships component aims to specify the social network of the user and the relationships it has with other users. This piece of information provides indicative and meaningful information for a user and will be taken into account by the procedures of WP5 for the identification of the volunteers for a task initiated by a requestor user.

Co-funded by the Horizon 2020 Framework Programme of the European Union

After the formulation of the list of the volunteers by WP5, the list is returned to the Social Context Builder for further analysis. Initially, the Social Preferences component takes as input the list of volunteers that was formulated by WP5, and analyzes the social and personal data of the volunteers (to the extent these are available), the characteristics of the task and performs a personalized ranking of the volunteers.

Finally, the Social Explanation component further analyzes the list of the volunteers and aims at providing arguments to the requestor about the acceptability of each volunteer with relation to the task at hand. The explanation is performed via a knowledge-based methodology that relies on rules to provide explanations and the reasons about the acceptance of a given volunteer for the user's task.

The first round of design and implementation that is reported in this deliverable is dedicated to the creation of basic components that will provide the main desired functionalities and will stand as a baseline. The design of the components will be expanded after the availability of social interaction data from the first WeNet pilot, so that the resulting implementations can be fine-tuned, trained, and tested on real WeNet data.

2.2 MACHINE COACHING AND PRUDENS TOOL

As part of our work, we have proposed and developed a general human-computer interaction paradigm and a corresponding tool to facilitate the acquisition of, and reasoning with, user-specific knowledge in a transparent manner. This section is devoted to discussing this paradigm and the corresponding tool. We will discuss later in this deliverable how the methodology and tool are used as part of two components of the Social Context Builder.

2.2.1. Machine Coaching

Machine coaching [4] is a human-computer interaction paradigm that puts forward an interactive form of knowledge acquisition between two participants:

- A human user who has in mind a set of heuristics and preferences according to which they make decisions when it comes to the specific task at hand – for the rest of this section we will refer to this set of heuristics and/or preferences as the user's *theory*;
- A cognitive agent who seeks to (transparently) elicit the aforementioned user's theory and is initialized with some default (possibly empty) theory regarding the task at hand.

The Machine Coaching Cycle (MCC) proceeds through the repetition of the following steps:

1. The human user seeks some piece of advice from the cognitive agent regarding some specific task;
2. The cognitive agent returns some piece of advice according to its own knowledge base regarding to that hand – which, initially, is the default knowledge base;
3. Then, the user has the option to interact with the cognitive agent by requesting some kind of explanation about the advice returned by the agent;
4. The agent, when prompted, returns an explanation for the advice it has provided in the form of an argument – i.e. a list of the rules that led it to draw its conclusions and yield that specific advice;
5. The user, in case they disagree with the line of argumentation the cognitive agent has presented, can provide their own argument which triggers

Co-funded by the Horizon 2020 Framework Programme of the European Union

the agent to update its knowledge base accordingly. This may be achieved, for instance, through the deletion of some specific rule presented in the agent's explanation from its knowledge base or by inserting a new rule with higher priority when compared to some other already existing rule in the agent's knowledge base.

The above cycle, under some certain protocols of user-agent interaction [4], is guaranteed to converge to the user's theory.

Before we proceed to demonstrate PRUDENS (PeRsonalized User-DEliberatioN Support), a tool which implements the abovementioned Machine Coaching Cycle, we should make the following remarks:

1. Machine Coaching is facilitating the *transparent* elicitation of the user's preferences and/or heuristics since at each step, the user is provided with the option to examine how the cognitive agent was lead to make some specific suggestion.
2. Eventually, Machine Coaching is a Machine Learning methodology and, as such, provides the user with *personalized* advice. That is, at the first request and up to the first time, a user will provide the cognitive agent with some counter-argument, the agent acts based upon the default knowledge base it is initially equipped with. However, as the user adds more and more rules which, typically, correspond to their own theory, the agent's advice, as well as the explanations related to them, are expected to bear more and more resemblance to what the user would have preferred.
3. The default knowledge base with which the agent reasons until it integrates the first counter-argument of the user is hard-coded, encoding simple common-sense heuristics regarding that task.
4. Machine Coaching also integrates some features of declarative programming, in the sense that any input provided is in the form of rules and/or literals of in first-order language.

Apart from the knowledge base, the agent has also access to a set of facts – which will be referred to as *context* – which are mostly related to the task as well as the volunteers that are being examined. So, a context is actually a description of the current situation upon which the user requests a piece of advice – e.g. it could be the specific profile of a volunteer.

2.2.2 PRUDENS

The PRUDENS tool constitutes a core part of the Social Context Builder. It is the major reasoning engine utilized in both the social explanation and social preferences components, and it enhances the transparency of the reasoning procedure. More precisely, PRUDENS is an implementation of the Machine Coaching cycle it is being developed to support the functionalities of social ranking and social explanation components. In that direction, PRUDENS seeks to iteratively converge to the user's theory with respect to some specific task so as to incorporate the user's knowledge and heuristics and, hence, yield more personalized results.

PRUDENS is a JAVA application that, given a knowledge base about some specific task and a certain context encoding facts about some specific situation referring to that task returns some piece of advice to the requestor, alongside with the corresponding explanation for that piece of advice, as predicted by the relevant Machine Coaching theory.

2.2.2.1 The language of PRUDENS

Co-funded by the Horizon 2020 Framework Programme of the European Union

Before proceeding to the description the key functionalities of the current version of PRUDENS, it is worth mentioning how rules and facts are encoded in it. PRUDENS. As described in 2.1 – see remark 4 – both the knowledge base as well as the list of facts (context) are described using some predefined first-order language. For our purposes, a prolog-like first-order language has been developed, which supports:

1. Constant symbols, which can take values such as "john", "at_office" or even numerical values – i.e. 1, 2, 3 and so on. In general, constants are representing certain entities or states in a specific situation. For instance, "john" may represent a certain volunteer and "office" may correspond to some certain office – e.g. a volunteer's job position. As a convention, non-numeric constant symbols may contain alphanumeric characters as well as the underscore character while they always start with a lower-case letter, so as to avoid confusion with variables – see next.
2. Variable symbols, such as "User", "Working_place" and "Cooking_skills". Variables serve as placeholders for constants and their names may include any alphanumeric character as well as the underscore character, while they always start with an upper-case letter, so as to be discriminated from constants.
3. Predicate symbols, such as "fatherOf(X,Y)" or "workingAt(User, office_1)". Predicates are defined by three parameters:
    a. Their name, i.e. the part preceding the left parenthesis "(" character – as with constants, the name of predicate may contain any alphanumeric/underscore character but it is mandatory that it starts with a lower-case letter;
    b. Their arity, i.e. the number arguments the may accept;
    c. Their arguments list, i.e. the comma-separated list included between left and right parentheses.

   So, in the case of "fatherOf(X,Y)", the predicate's name is "fatherOf", its arity is two (2) and the arguments list is [X, Y], where, in this case, both its arguments are un-instantiated variables. In general, predicates are used so as to describe relations between entities, so "fatherOf(X,Y)" may be interpreted as "X is the father of Y" and "workingAt(User, office)" may be interpreted as "User works at office_1", where "User" is some variable and "office_1" is some specific office.
4. Literal symbols, such as "john", "workingAt(User, office_1)" and "-available(User)". Literals are defined by two parameters:
    a. Their main part, which may be either a predicate, such as "workingAt(User, office_1)" or a constant such as in "john";
    b. Their sign, which may be either positive or negative. In general a literals sign is denoted using a "-" (minus character) if it is negative and no additional character if it is positive. The intended interpretation of a negative sign is that of logical negation, so "fatherOf(X,Y)" is interpreted as "X is father of Y" while "-fatherOf(X,Y)" is interpreted as "X is *not* father of Y".

   Two literals that have the same main part but different sign are called *conflicting* literals.
5. Rules, which consist of three parts:
    a. The rule's name, which may be any possible alphanumeric string (including underscore) which, by convention, is starting with a capital letter and is not including "**::**" as a substring, since "**::**" is used as a separator between the rule's name and its main part;
    b. The rule's body, which is a comma-separated list of literals;
    c. The rule's head, which is a single literal that succeeds the rule's body and is separated from it with the special string " implies ".

   Also, note that each rule ends with the special character "**;**" right after its head. An example of rule may be the following one:

   Rule_1 :: fatherOf(X,Y), siblings(Y,Z) **implies** fatherOf(X,Z)**;**

   The intended interpretation of a rule with body [L_1, L_2, …, L_n] and head H is that of an

Co-funded by the Horizon 2020 Framework Programme of the European Union

if-clause of the form “**if** L_1 and L_2 and … and L_n are true **then** H is also true”. So, the above rule may be interpreted as “**if** X is father of Y and Y and Z are siblings **then** X is father of Z”. Lastly, let us also mention that two rules that lead to conflicting literals are called *conflicting* rules.

Using the above language, a knowledge base is actually a list of rules enhanced with a priority relation amongst conflicting rules. As a convention, we assume that rules that appear first in the list are of higher priority than conflicting rules that appear lower in the list.

2.2.2.2 Functionalities supported by PRUDENS

PRUDENS, as it has been mentioned above, is an implementation of the MCC as described above. As part of this implementation, PRUDENS supports the following functionalities:

- A reasoning mechanism which, given a knowledge base and a context – i.e. a list of non-conflicting literals – makes inferences through repetitive application of modus ponens – based on the context’s literals and using the knowledge base’s rules – and by resolving any conflicts using the knowledge base’s priorities.
- For each literal that is finally inferred, PRUDENS is capable of returning an explanation about how this literal was inferred – i.e. the argument that, finally, led to the inclusion of this literal in the final output.
- It is capable of updating its knowledge base by deleting and/or adding rules as well as changing the priorities between existing conflicting rules.

With the above functionalities supported, the MCC is implemented in PRUDENS in the following way:

1. Initially, PRUDENS receives a default knowledge base including hard-coded rules that capture elementary heuristics about some specific task.
2. Upon the requestor’s request for assistance in that task, PRUDENS receives data about volunteers and decides *separately* for each one whether they should be asked or not in order to assist the requestor in that task.
3. The requestor is presented with the corresponding output and can, if they wish to, counter-argue against some explanations on why to ask some certain volunteer to assist in that task. The requestor’s feedback – addition/deletion of rule(s) and/or alteration of conflicting rules’ priorities – is received and integrated in PRUDENS’s knowledge base.

Thus, by repetition of the above steps, PRUDENS gradually builds a knowledge base that contains the heuristics and preferences of the requestor.

The above workflow of PRUDENS tool is illustrated in Figure 2.

Co-funded by the Horizon 2020 Framework Programme of the European Union

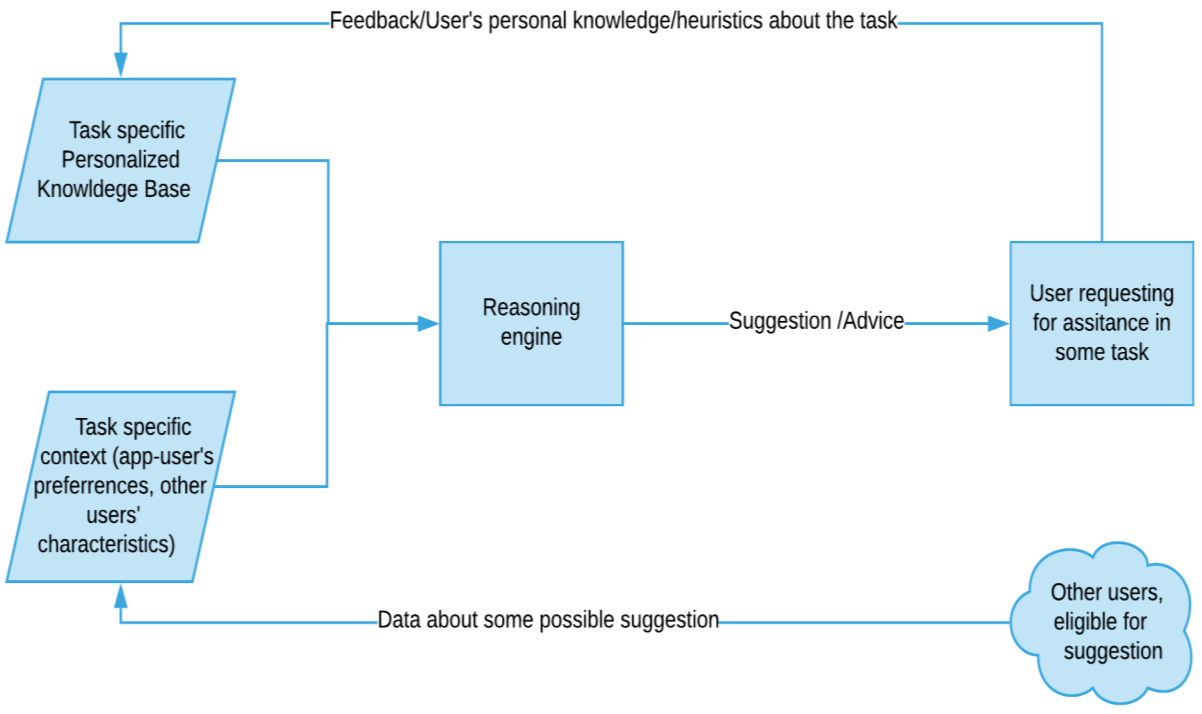

Feedback/User's personal knowledge/heuristics about the task
Task specific Personalized Knowldege Base
Reasoning engine
Suggestion /Advice
User requesting for assitance in some task
Task specific context (app-user's preferrences, other users' characteristics)
Data about some possible suggestion
Other users, eligible for suggestion

Figure 2: PRUDENS's interaction with data and users

In the following sections, the three components of the Social Context Builder are presented and their functionality is described.

3. SOCIAL RELATIONSHIPS

The first component of the Social Context Builder is the analysis of Social Relationships. It aims at analyzing the social interactions of the users and specifies two pieces of information: the relationships between users, and the tie strength of their relationships.

3.1 Relationship Specification

Initially, for the specification of the types of relationships between users, the personal information of each user is necessary to be harvested and analyzed. After that, based on the collected personal information of the users, their relationships are identified.

3.1.1 Personal Information

The personal information of the users captures the main characteristics of a person. They are related to main demographic info of the user and concern the place of living, work and education data, personal skills and interests. Specifically, the personal information that was recorded for the users concern the following:

Table 1: Personal information types

Co-funded by the Horizon 2020 Framework Programme of the European Union

Personal Information	Type	Explanation
Place of birth	Static, Single Value	The place that the user was born.
Date of birth	Static, Single Value	The date of birth of the user.
Place of living	Dynamic, Single Value	The main location-place that the user lives.
Work organization	Dynamic, Multi-value	The organization where the person currently works.
Education discipline	Dynamic, Multi-value	The discipline that the user studies.
Education organization	Dynamic, Multi-value	The institute where the user studies/studied.
Skills	Dynamic, Multi-value	The main skills that the user possesses.
Interests	Dynamic, Multi-value	The main interest of the user.

The type of personal information of a user can be either static or dynamic. A *static* denotes that the personal characteristic does not change over time and so, once harvested and specified can be used without the need to be continuously monitored or updated. In contrast, *dynamic* denotes that a characteristic can change over time and so, it is necessary to be continuously monitored and frequently updated.

3.1.2 Relationship Types

Based on the personal characteristics of the users, specific types of social relationships are specified. A knowledge-based approach is used to analyze social information (from Facebook) and determine main types of relationships among users. To that end, the approach takes as input two users, analyses the profile information of their accounts and based on the above personal information, specifies the relationships they have. The types of relationships that are identified between two users are the following:

Table 2: Main types of relationships

Relationship Type	Description

Co-funded by the Horizon 2020 Framework Programme of the European Union

Same family	Specifies that two users are members of the same family.
Same birthplace	Specifies that two users were born in the same city/town/country.
Same place of living	Specifies that two users live in the same location.
Same work organization	Specifies that two users work in the same organization.
Studies in the same organization	Specifies that two users study in the same organization.
Studies the same discipline	Specifies that two users study in the same discipline.

The exact relationships that two users have, are specified based on the analyses of their stored personal information.

3.2 TIE STRENGTH SPECIFICATION

Social relationships can be analyzed from different scopes; nevertheless, all users connections on Online Social Networks regardless of their "real life" relationship status are summarized into one type – friendship. Furthermore, users of online social networks often have lists of connections that also contain unknown users such as e.g. public figures who fall under the same "friendship" category as "real life" friends and acquaintances. Due to this lack of relationship differentiation based on its quality and intensity as discussed above, online social networks often have a difficult task deciding which information and recommendation to display to their users, how to provide a better and even more interesting service, and to whom to promote certain products.

It is important to determine the friendship intensity of online social network users based on their interaction online. Social network users can be represented by a social graph which consists of nodes and connections between them. The nodes represent users and the connections are some kind of social interaction among them [8] .

The Social Relationships component focuses on the weight of friendship, that is to what extent is user A friend to user B. The weight of a friendship between two users is calculated based on the analysis of the interaction between them and includes, with certain weight of significance, main communication and interaction parameters. In determining the weight of a friendship, one must be aware that a friendship is not necessarily a reflexive relationship, so it is possible that user A considers user B a better friend than user A is considered a friend by user B. The goal is to recognize and estimate the weight of the relationship between users by analyzing their interactions and their accounts. The existing binary connection, which describes whether someone is connected (e.g. they are friends) with someone else or not (0

Co-funded by the Horizon 2020 Framework Programme of the European Union

or 1), can become more detailed with a weight that represents the extent that someone is connected to someone else. Thus, we are building an implicit social network over an explicit social network (Figure 2). The formulated implicit social network is described with a directed weighted graph.

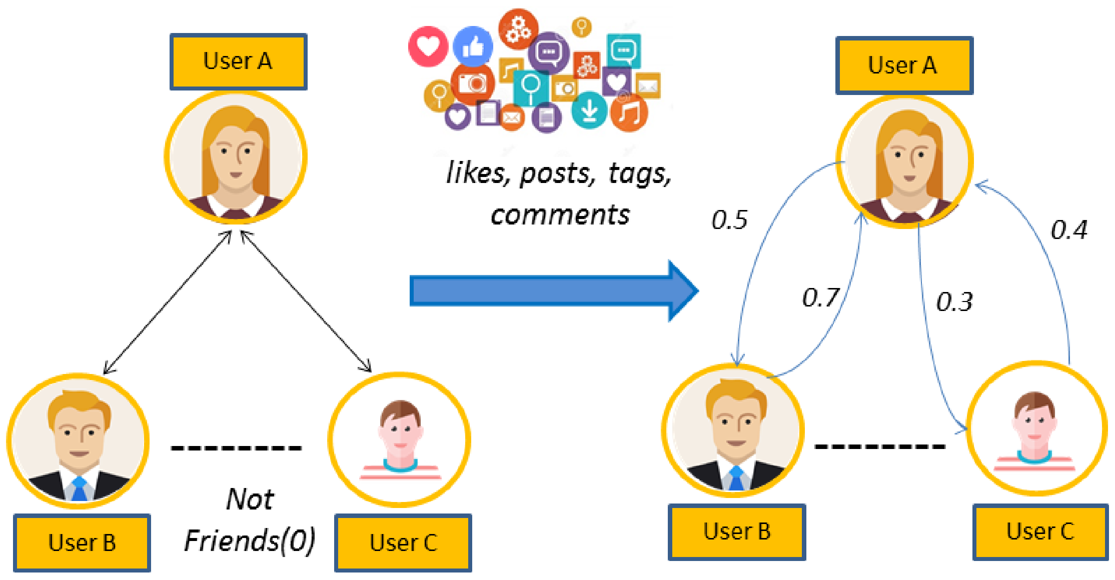


Figure 3: Weighted relationships and tie strength

Friendship is shown as a one-way connection from *user A* to *user B*, where *user A* is the *ego-user*, and *user B* is in the network friend of *user A.* The weight of friendship between user A and user B is not necessarily equal in both ways. The friendship weight is calculated based on a set of interaction parameters between the two users. The main interaction parameters that are analyzed and used in the calculation of the friendship weight are the following that are presented in Table 3.

Table 3: Interaction parameters

Interaction parameter	Description
Comments of User B on the posts of ego User A	Calculates the number of the comments that a user has made to another.
Likes of User B on the posts of ego User A	Calculates the likes that a user has made to another.
Tags that have together	Calculates the tags in photos that the two users have together

Co-funded by the Horizon 2020 Framework Programme of the European Union

After the collection of the interaction data between two users, the normalization process takes place. The numerical parameters of the users' interactions are normalized as follows:

$$x_{norm,i} = \frac{x_i - x_{min}}{x_{max} - x_{min}}$$

x_i is the i^{th} value of the parameter, x_{min} is the minimal value of that parameter, x_{max} is the maximal value and $x_{norm,i}$ is the normalized i^{th} value. Normalization techniques are used to transform various parameters of a large range of values to a lower range in order to facilitate processing. In following sections (Section 6)

The overall tie strength of one users with regard to another, based on their interaction is calculated as:

$$friendship\_tie\_weight = \sum_{1}^{n} \frac{w_i * x_{norm,i}}{n}$$

Where w_i represents the weight of the parameter i in the friendship tie calculation.

In the demonstration section (Section 6) the functionality of the social relationship component is illustrated and explained in detail.

It is worth emphasizing again that our emphasis here is on using external to the WeNet platform data, to the extent that those are made available with the consent of the user and following proper GDPR-compliant procedures. Nonetheless, it can be easily seen that the same ideas described above can be used when WeNet interaction data are available. In the same way that a "like" of one user by another is taken into account in computing their tie strength, so can the rating of a volunteer that has helped a user within the WeNet platform. We will focus on this part of the learning process as we transition from Task T3.1 to task T3.2, and as it becomes clear what regularities we will be able to observe in actual WeNet interaction data from the first WeNet pilot.

4. SOCIAL PREFERENCES

The Social Preferences component is responsible for ranking volunteers with respect to a specific task that a user poses. The component takes as input the list of volunteers that the procedures developed in the context of WP5 have specified and after that, performs a personalized ranking of the users and implements a knowledge-based methodology that utilizes rules to perform the user ranking.

As discussed in 2.2, PRUDENS is the major reasoning engine utilized in both the social preferences and the social explanations components. While the invocation of PRUDENS in a social explanations tasks seems straightforward — since PRUDENS is capable of reasoning and advising as well as providing explanations about its suggestions — the invocation of

Co-funded by the Horizon 2020 Framework Programme of the European Union

PRUDENS as a tool assisting social preferences is not done, as for now, in the same way. In social preferences, it is mostly the reasoning capabilities of PRUDENS that are being utilized in the following way: a) PRUDENS starts with a default knowledge base which contains rules regarding user ranking; b) then, when the user requests the corresponding user ranking with respect to some task, PRUDENS reasons and provides a partial order of the users to the requesting user.

Notably, the above procedure does not utilize the revision capability of PRUDENS – i.e. the user cannot counter-argue to the arguments provided for the ranking. Instead of directly asking the user about feedback regarding the ranking itself as well as its accuracy, it is intended that such information is being passively selected and then passed to PRUDENS as feedback.

Given that the ranking procedure based on PRUDENS is tailored to the characteristics of the task, in the Case Study section (Section 6) the functionality is illustrated.

Let us consider the following simple example. We assume that Bob wants some assistance to clean his attic and asks for some help. The application will return Bob some users that are considered appropriate for the task alongside with some explanation about why these users have been chosen (see 5 for more information) as well as a list in which all these users have been ranked from the most appropriate to the least one.

Then, after this list is presented to the Bob, he makes his choices according to his own personal criteria and, let us assume that, instead of picking the first two users in the list, he choose the third and the last one. This triggers a (passive) mechanism of preference elicitation that marks down the choices of Bob and takes them into consideration in future cases, by altering the level of trust in each criterion used during the ranking procedure.

Thus, the basic social learning algorithm that we have implemented for this ranking mechanism reduces the ranking of volunteers to a matter of a real-valued criterion, which is used by PRUDENS to do the actual ranking, while also leaving PRUDENS to offer simple explanations about its ranking. We are currently exploring richer preference structures, including ones based on Conditional Preference Networks [18] , for which we had developed in the past learning and reasoning algorithms [19] , [20] . We plan to investigate whether such approaches (which rely heavily on having full information on the items being ranked) can be appropriately extended to handle missing information, and whether their reasoning (which relies on a different mechanism than classical Modus Ponens) can be used to provide transparent and involved (based on the user’s profile, and the volunteers’ public parts of their profiles, thus respecting their privacy) explanations on how ranking is determined.

A more involved preference structure and corresponding learning and reasoning algorithms will also cleanly facilitate the introduction of diversity measures (which are expected to be implemented in our learning algorithms as part of Task T3.2) in the ranking. This can be done, for instance, by capturing preferences external to the user (e.g., coming from WP4) that promote the higher ranking of volunteers that are diverse (in the appropriate features, as determined by WP1) from the user, or from the volunteers that appear in the list as a whole. Although diversity could be promoted also through the inclusion

Co-funded by the Horizon 2020 Framework Programme of the European Union

of appropriate rules in the theory that PRUDENS uses for ranking, we expect that other structures might be able to accommodate this diversity requirement more cleanly. This investigation is part of our near-future work, as we work towards the next iteration of the component.

5. SOCIAL EXPLANATIONS

The Social Explanations component is responsible for providing arguments to the user about the acceptability of each volunteer. It utilizes PRUDENS and the Machine Coaching Cycle to provide explanations and the reasons on why a given volunteer (identified by WP5) for the requestor's task should eventually be accepted by the requestor to help with the task. It is worth noting that the explanations in question are not explanations of why a certain volunteer was proposed in the first place (by WP5). Rather, we are interested in a post-filtering step where the user is aided to confirm which, among those volunteers, will actually be involved in the task and interact with the requestor. Accordingly, the requestor is able to question the explanations that are provided, and coach the component towards learning to offer personalized explanations to the requestor (which might have little to do with the reasons that WP5 chose to return those volunteers in the first place).

The first version of the reasoning mechanism of the explanation component needs access to a propositional knowledge-base and a propositional context which encodes all the task-specific information; together, these yield a theory that provides suggestions to the user.

Let us now consider a simple example that describes the above functionality. Below, for reasons of simplicity, all the interaction is demonstrated at a high level – e.g. the rules are presented in natural language. A more in-depth demonstration of the MCC is provided in Chapter 6.

Let us assume that Alice needs some help with her everyday transportation from and to work and is, hence, looking for some other user who is willing to share expenses and give her a ride from and to her office. Let us also assume that, initially, the cognitive agent of the social explanations component is equipped with a knowledge base containing the following hard-coded rule:

Rule_1: If some user works close to Alice's office and lives near her house **then** this user may be a good candidate.

So, Alice decides to create a task on the WeNet platform looking for volunteers to help with her particular task. WP5 utilizes Alice's and others' social relationship network, and identifies a list of volunteers for the task. Alice is now looking for aid on which among those volunteers to select to actually help her with her task, and she gets the following explanation:

From all the volunteers I found only two that fit the criteria of the description provided. These are: Bob and Christine who were selected because they work close to your office and also live near your house.

Co-funded by the Horizon 2020 Framework Programme of the European Union

Alice, now, looking at the public parts of the profiles of the two users realizes that, while Christine is an adequate choice, as far as Bob is concerned, she thinks that he should not have been suggested at all. Her counter-argument is the following:

> *According to Bob's profile, indeed he is working close to my office and he is living in the apartment above mine, nevertheless, Bob is a night watchman in the museum next to my office, so he will never be able of giving me a ride to work.*

With this in mind, Alice informs the social explanations tool that a new rule should be included that states:

> **Rule_2: If** some volunteer works does not have the same working schedule as Alice **then** do not suggest that volunteer.

Also, Alice notifies the tool that this new rule should be of higher priority to **Rule_1** so as to "defeat" it in any case both are triggered for some volunteer.

So, the cognitive agent, using its new knowledge base which now contains both Rule_1 as well as Rule_2, with Rule_2 being of higher priority than Rule_1 would not suggest Bob as an acceptable volunteer for the task since:

- Rule_1 would be triggered since he works close to Alice's office and lives next to her;
- Rule_2 would also be triggered since he does not share the same working schedule with her.

The above rules lead to conflicting pieces of advice, since Rule_1 argues in favor of suggesting Bob while Rule_2 argues against such a suggestion. In order to resolve this conflict, the agent consults the rules' priorities and, since Rule_2 is of higher priority than Rule_1, it is this rule's piece of advice– i.e. not to suggest Bob – that is being proposed to the user. Although we will not delve into the details of how conflicts are resolved, suffices to say that both the Machine Coaching framework and PRUDENS are sufficiently expressive to capture a variety of phenomena (see [4], where we have introduced Machine Coaching).

Nonetheless, we will briefly mention that the approach that the Machine Coaching framework that we have developed and used for the purpose of representing, reasoning with, and learning Social Explanations offers transparency by design, personalization to the user, clean handling of missing information, including lack of access to the private parts of the profiles of volunteers, as needed to respect their privacy (since arguments can be triggered based on whatever information happens to be available), while remaining efficient in its operation.

As we move forward towards Task T3.2 and the use of real WeNet data, we expect to run large-scale experiments (data permitting) to evaluate the empirical scalability of our algorithms, and also the ability of users to coach the cognitive agent towards offering useful explanations. We further plan to include diversity-relevant rules in the knowledge-base of the cognitive assistant, which will be learned not by the personalized feedback of each individual user, but either be learned through a community-wide machine coaching process, or be identified by WP4 or WP9 as part of an incentivization scheme or a scheme to comply to certain ethical considerations. We expect to extend the machine coaching algorithm and implementation to make sure that such “add-on” rules retain their higher priority even when a particular user might wish to override them based on personal preferences.

6. CASE STUDY

In the context of the case study, we demonstrate the functionality of the three components and the methodologies in the scenario of Social Eating. The Social Eating scenario demonstrates the proof-of-concept functionality of the algorithms and illustrates the overall workflow of the Social Context Builder. Below we discuss the characteristics of the scenario, the data used and we explain the functionality of the components.

6.1 THE SOCIAL EATING SCENARIO

The Social Eating scenario has been identified by the WeNet consortium as a concrete use case to consider as we move forward, since it captures several of the key desiderata of an interaction within the WeNet platform. We need to emphasize that even though we have used this to demonstrate our developed components, nothing in our work so far is specific to this scenario.

According to the Social Eating scenario, then, a certain user is looking to have dinner with friends or other users that could become potential friends. Our work aims to assist the user through the WeNet platform in finding and inviting other users and friends to dinner as well in ranking volunteers and providing explanations on their acceptability to come to dinner.

Let us first give the high-level interaction of Bob within WeNet, along with the actions taken by the various components that we have designed and implemented:

- Bob creates a WeNet account, logins, joins the Social Eating app.
  Action: Parse Facebook to initialize his social relationship network.
- Bob creates a task: “multi-ethnic food, this Friday, my place”.
- A list of volunteers is compiled and presented to Bob.
  Action: Re-order list based on Bob’s social preference ranking.
- Bob inquires for an explanation of why some volunteer should be or not be in an acceptable group for the particular task.
  Action: Explain based on Bob’s social explanation arguments.
- Bob counter-argues if he disagrees with the explanation.
  Action: Update Bob’s profile on his social explanation arguments.
- Eventually, Bob confirms and invites a group of volunteers.
  Action: Based on Bob’s confirmed invitations, update his social relationship network and his social preference ranking.
- Bob engages in social eating, and rates the participants.
  Action: Based on Bob’s participant ratings, update his social relationship network and his social preference ranking.

We will now go through the steps of the interaction and discuss in more detail how our implemented components come into the picture.

After Bob joins the Social Eating app, and assuming that he has provided an informed

Co-funded by the Horizon 2020 Framework Programme of the European Union

consent to that end, the social relationship component analyzes Bob's Facebook account and identifies the friends of Bob that are also members of the WeNet platform, along with the tie strength he has with them. To determine the tie strength between Bob and each one of his friends, social interaction and behavior aspects like posts, comments, likes are analyzed. After that, the system has an initial version of the social relationships that Bob has and the tie strength with his friends.

Let's suppose that Bob creates a task in the system, in the context of Social Eating, named "multi-ethnic food, this Friday, my place", specifying the time and place of the event, and also presumably his desire for a diverse group of participants to attend, each cooking and bringing a dish from their own ethnic cuisine. A list of volunteers is compiled (from WP5) and is presented to Bob. The social preferences component analyzes the list and performs an ordering of the volunteers based on the characteristics of the task, the Bob's profile, and whatever information it has about the volunteers that respects the private parts of their profiles.

Bob could inquire for explanations of why some volunteer should be or not be in an acceptable group for the particular task. The social explanation component analyzes the characteristics of the volunteer (again, respecting the private part of their profile) and explains the reasons why the volunteer could be invited in the task. In the general case, the explanation might relate to the provisional inclusion or exclusion by Bob of another volunteer in the list of his invitees. Bob could provide counter arguments when the suggestions are not accompanied by convincing explanations, so that the suggestions improve over time.

Based on the final list of invitees chosen by Bob, and based on the feedback offered by Bob after the actual dinner, the social relationships and social preferences are also updated.

6.2 INITIALIZATION OF SOCIAL RELATIONSHIPS

In this subsection, we demonstrate the procedure and tool that we have developed to gather data from Facebook accounts towards initializing a user's social relationship network.

We explicitly note that the tool that we have developed was not used on any existing Facebook account. Instead, we created dummy Facebook accounts and populated them with made-up data as a way to test our developed tool and procedure. The tool will continue to be developed in this manner, using dummy accounts and made-up data, until the deployment of the WeNet platform, and until WeNet users provide an informed consent through appropriate GDPR-compliant procedures that would allow us to use the tool on their Facebook accounts.

Below we present the characteristics of the dataset that we have created for demonstration purposes, which was formulated and consists of the users' information as well as their public social interactions. The data of the users were collected via the Facebook accounts of some example dummy users that were assumed to be participating in a social dinner scenario. Also, the dataset was augmented with additional artificial data to facilitate the analysis of the users and illustrate in better detail the functionality of the components.

The dataset that has been created for the purposes of the demonstration and testing of the components' functionality assumes, as it has been already mentioned, that the application has access to two different types of data sources: a) Facebook data from the users that are obtained through their Facebook profile at the time of their signing-up to the WeNet platform; b) personal data provided directly by the user – e.g. through a questionnaire.

Co-funded by the Horizon 2020 Framework Programme of the European Union

Table 4: User data

Attribute	Level	Short Description
Place of birth	Single Value	The value of the birth place.
Date of birth	Single Value	The value of the birth date.
Place of living	List	The value of the current place of living.
Work organization	List	The list of the organizations that user works.
Education discipline	List	The list of education disciplines.
Education organization	List	The list of education organizations.
Skills	List	The list of skills.
Friends	Member of set: {no, somewhat, a_lot}	The classification of a user's friend as "not", "somewhat" or "a_lot" will serve as a "weighting" criterion in several other attributes below.
cook_skills	Member of set: {low, medium, high}	This field measures the level of the under-suggestion user's skills in cooking and, hence, possible suitability for a dinner, either as a cook or as a guest that would appreciate such a social gathering.
Social	Member of set: {not, somewhat, a_lot}	This field measures the amount to which a user is keen to participate in social events and gatherings such as parties, social dinners and so on. It is expected that highly social users may be more suitable a dinner as the one in our task.
friend_of_friend	Boolean Value	This field is included so as to capture a key feature of social gatherings: extending one's social circles. This is facilitated by including friends of friends as eligible candidates for suggestion under some circumstances.
Interests	Member of set: {irrelevant, somewhat-relevant, relevant}	This field is intended to encapsulate the level up to which the user's interests are relevant to the task at hand - in our case, organizing a social dinner. This is not necessarily indicating relevant skills; hence, it is considered a separate attribute from cook_skills.

Co-funded by the Horizon 2020 Framework Programme of the European Union

recent_ events	Member of set: {none, some, many}	This field indicates similar events to which the candidate user has participated; in our case, similar events may be other social dinners or social interactions that share a common structure. Using this field, we include temporal dynamics into our app's user analysis.

Two notes are in order. First, one could consider a number of other attributes (e.g., does the user have a car). We have chosen to restrict the discussion to attributes that are relevant to the Social Earing scenario for simplicity. Second, we have purposefully included in the table above personal pieces of information to make a certain point. Not all pieces of information will be available for every user. In particular, for the user who has consented to the use of their Facebook profile for the initialization of their social relationship network, all information can be assumed to be available, since both the private and public part of a user's profile can be used on their own application when interacting within the WeNet platform. On the other hand, information about users in the social circle of that user is clearly not fully available, as parts of it (e.g., gender) is considered to be private and cannot be accessed. Part of our ongoing work towards Task T3.2 is exactly dealing with the proper handling of such missing information in user profiles in a way that allows the proper computation of tie strengths.

For the other two components, no externally-provided data is assumed to be given for their initialization. Instead, we assume that a pre-specified theory is included in the components that ranks and explains the acceptability of volunteers prior to the availability of WeNet data.

6.3 DEMONSTRATION OF COMPONENTS

First, we illustrate the functionality of the social relationship component and after that, the ranking and the social explanation component.

6.3.1 Social Relationships

The Social Relationships components initially collects and analyzes the personal information of the relevant user, which captures the main characteristics and data of a person as they relate to the demographic info of the user, the place of living, work and education data, personal skills and interests. Based on these, specific types of relationships are determined.

The social information among users derives from the analyses of the interactions between two users and aims to specify main interaction parameters. The social interaction information that was collected concerns the interaction between pairs of users as found in posts and more specifically, the comments of the users in posts, the likes and the tags in photos. This information was collected automatically via the Facebook API as well as the Facebook scrapper that was designed and implemented.

The Facebook API and the crawler get a Facebook account and collect a set of information that are available from the account: the user's friends, the user's timeline, the user's information in the "about" area, as well as additional information related to the user.

The dataset that captures the users and their personal and interaction information is used in the case study as a means to show the workflow of the components and shed light on their exact functionality. Example users with example data are given below.

Co-funded by the Horizon 2020 Framework Programme of the European Union

Table 5: Personal information for example users

User	Place of birth	Date of birth	Place of living	Work organization	Education discipline	Education organization
User_1	Athens	1988	Athens	-	Biology	UCY
User_2	Nicosia	1998	Nicosia	-	Computer Science	OUC
User_3	Athens	1992	Nicosia	IBM	-	-
User_4	Nicosia	1997	Nicosia	-	Computer Science	OUC
User_5	Larnaca	1991	Larnaca	-	Biology	UCY

Table 6: Social information regarding comments on posts for example users

User	User_1	User_2	User_3	User_4	User_5
User_1	-	2	0	3	1
User_2	0	-	2	6	2
User_3	1	0	-	1	0
User_4	1	8	1	-	0
User_5	2	1	0	1	-

Table 7: Social information regarding likes on posts for example users

User	User_1	User_2	User_3	User_4	User_5
User_1	-	10	0	11	6
User_2	8	-	12	18	0
User_3	5	0	-	3	8
User_4	2	14	4	-	0
User_5	6	2	2	4	-

Co-funded by the Horizon 2020 Framework Programme of the European Union

Table 8: Social information regarding tags for example users

User	User_1	User_2	User_3	User_4	User_5
User_1	-	0	0	2	1
User_2	1	-	0	1	2
User_3	3	0	-	0	2
User_4	2	1	2	-	0
User_5	1	1	0	0	-

Based on the personal information of the users, the types of the relationship among the users that are specified are the following:

User_1
Same birthplace with User_3
Studies in the same discipline with User_5
Studies in the same organization with User_5

User_2
Same birthplace with User_4
Same place of living with User_3
Same place of living with User_4
Studies in the same discipline with User_4
Studies in the same organization with User_4

User_3
Same birthplace with User_1
Same place of living with User_2
Same place of living with User_4

User_4
Same birthplace with User_2
Same place of living with User_2
Same place of living with User_3
Studies in the same discipline with User_2
Studies in the same organization with User_2

User_5
Studies in the same discipline with User_1
Studies in the same organization with User_1

After that, the social interaction among the users within the Facebook platform is analyzed in order to assess their connection and calculate the tie strength of them. The tie strength is calculated based on the comments the likes and the tags of the users. For the calculation, we normalize the value of the parameter examined following the

Co-funded by the Horizon 2020 Framework Programme of the European Union

equations of Section 3. The tie strength is calculated and presented in following table. In the context of the case study equal weight is given to each social parameter.

Table 9: Tie strength for example users

User	User_1	User_2	User_3	User_4	User_5
User_1	-	0.53	0	0.67	0.63
User_2	0.31	-	0.33	0.83	0.44
User_3	0	0.17	-	0.46	0.56
User_4	0.42	0.78	0.36	-	0
User_5	0.67	0.61	0.11	0.39	-

Below we illustrate the tie strength on the network consisting of the five example users.

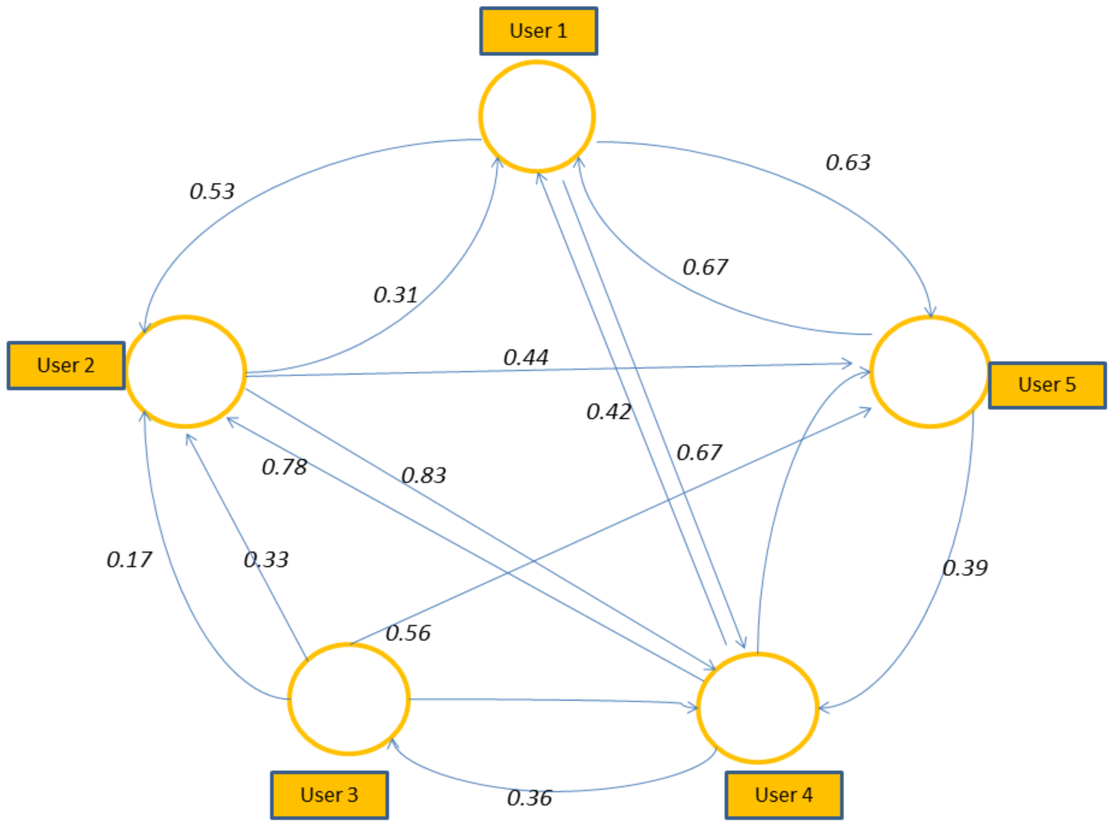


Figure 4: Network tie strength based on user social interactions

This initial social relationship network can be used by WP5 to identify volunteers after a user creates a task. As we have already mentioned, the user's final

Co-funded by the Horizon 2020 Framework Programme of the European Union

choice of which volunteers to interact with, and the rating the user gives to their interaction with that volunteer can easily be used to update the initial social relationship network. As the details of this updating mechanism are highly data-specific, they can be determined only after we have access to actual WeNet data, and will be incorporated as part of our work for Task 3.2. For now, a simple (as expected for Task 3.1) solution is used, where a person volunteering is taken as evidence to strengthen the tie from the volunteer towards the requestor, while a requestor accepting and rating positively a volunteer is taken as evidence to strengthen the tie from the requestor towards the volunteer. In case a volunteer is explicitly rejected from being selected, or is rated negatively, this is taken as evidence to weaken the corresponding tie.

6.3.2 Social Preferences

The ranking methodology, using PRUDENS, is summarized in the next figure which depicts the basic data flows during the social preferences procedure.

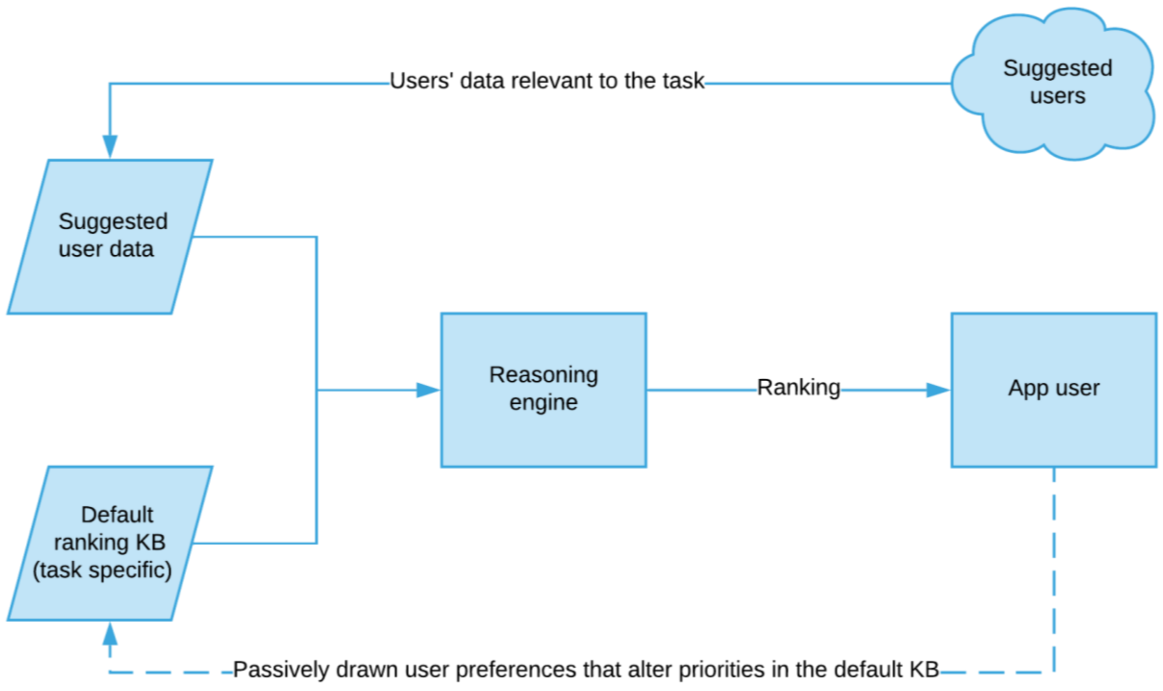

Users' data relevant to the task
Suggested users
Suggested user data
Reasoning engine
Ranking
App user
Default ranking KB (task specific)
Passively drawn user preferences that alter priorities in the default KB

Figure 5: The key data flows in the Social Preferences component

As a matter of initializing the ranking procedure, in the absence of WeNet interaction data for a particular user who has just joined the WeNet platform, the following default knowledge base is used for the Social Preferences component, as depicted in the table below, where App_user is the task requestor and the one who will be viewing the ranking, whereas User_1 and User_2 are variables / placeholders for any pair of volunteers in that list.

Table 10: Default ranking knowledge base

Rule's name	Rule's body	Rule's head

Co-funded by the Horizon 2020 Framework Programme of the European Union

Ranking_1	conflicts(App_user,User_1,A), conflicts(App_user,User_2,A), in_common(App_user,User_1,B), in_common(App_user,User_2,B), cooking_skills(User_1,X), cooking_skills(User_2,Y), ?X>Y	higher(User_1,User_2)
Ranking_2	conflicts(App_user,User_1,A), conflicts(App_user,User_2,A), in_common(App_user,User_1,X), in_common(App_user,User_2,Y), ?X>Y	higher(User_1,User_2)
Ranking_3	conflicts(App_user,User_1,X), conflicts(App_user,User_2,Y), ?X>Y	higher(User_2,User_1)

To demonstrate the application of the rules above for ranking purposes, consider the following example set of users who act as volunteers in our Social Eating scenario. The characteristics for each volunteer are computed based on the public part of their profile, and based on the profile of the requestor and any task-related information that is available.

Table 11: Characteristics of the users

User	Characteristics
user_1	conflicts(app_user,user_1,0), in_common(app_user,user_1,2), cooking_skills(app_user,user_1,4)
user_2	conflicts(app_user,user_1,0), in_common(app_user,user_1,2), cooking_skills(app_user,user_1,3)
user_3	conflicts(app_user,user_1,0), in_common(app_user,user_1,1), cooking_skills(app_user,user_1,4)
user_4	conflicts(app_user,user_1,1), in_common(app_user,user_1,1), cooking_skills(app_user,user_1,3)
user_5	conflicts(app_user,user_1,3), in_common(app_user,user_1,2), cooking_skills(app_user,user_1,4)

Now, using the above, we will gradually infer a ranking of the above users as follows. At first, the characteristics of user_1 and user_2 trigger rule **Ranking_1** which yields the literal

higher(user_1,user_2). Then, user_1 and user_3 trigger rule **Ranking_2** which yields that **higher(user_1,user_3)**. As for the pairs (user_1,user_4) and (user_1,user_5), they both trigger rule **Ranking_3** which yields **higher(user_1,user_4)** and **higher(user_1,user_5)** respectively. Next, we proceed with the pair (user_2,user_3) which triggers rule **Ranking_2** which yields **higher(user_2,user_3)**. As far as the pairs (user_2,user_4) and (user_2,user_5) are concerned they both trigger rule **Ranking_3** which yields **higher(user_2,user_4)** and **higher(user_2,user_5)** respectively. The same applies to the pairs (user_3,user_4), (user_3,user_5) and (user_4,user_5), which all trigger rule **Ranking_3** which yields in each case **higher(user_3,user_4)**, **higher(user_3,user_5**, **higher(user_4,user_5)**.

The above can be summarized in the following table, where light-blue squares in position **(i,j)** denote that user **i** is ranked higher than user **j**.

Table 12: Ranking of the users

j / i	user_1	user_2	user_3	user_4	user_5
user_1		🟦	🟦	🟦	🟦
user_2			🟦	🟦	🟦
user_3				🟦	🟦
user_4					🟦
user_5					

The above can be summarized in the Hasse diagram shown in next figure.

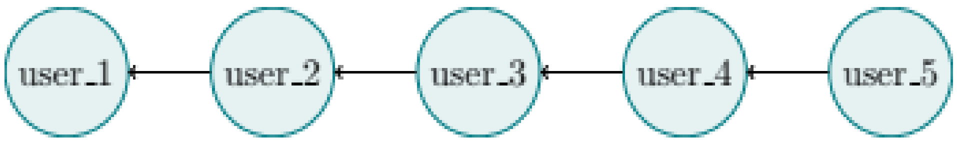


Figure 6: Hasse diagram for the first set of ranked users

This is, then, the ranking presented to the requestor.

As another case of user ranking, let us consider the following set of users / volunteers:

Co-funded by the Horizon 2020 Framework Programme of the European Union

Table 13: Characteristics of the users

User	Characteristics
user_1	conflicts(app_user,user_1,0), in_common(app_user,user_1,2), cooking_skills(app_user,user_1,4)
user_2	conflicts(app_user,user_1,0), in_common(app_user,user_1,2), cooking_skills(app_user,user_1,3)
user_3	conflicts(app_user,user_1,0), in_common(app_user,user_1,1), cooking_skills(app_user,user_1,3)
user_4	conflicts(app_user,user_1,1), in_common(app_user,user_1,1), cooking_skills(app_user,user_1,3)
user_5	conflicts(app_user,user_1,3), in_common(app_user,user_1,2), cooking_skills(app_user,user_1,4)

The actual difference between this and the previous dataset is that now user_3 and user_4 have exactly the same characteristics as far as their cooking skills, the common attributes they share with the app user, and the frequency with which they conflict with the app user. So, as in the previous case, user_1 is ranked higher than any other user and user_2 is ranked higher than user_3, user_4 and user_5, while user_5 remains the last user in the ranking. It remains to compare user_3 and user_4 but, since they are identical with respect to our default knowledge base's ranking criteria, no rule is triggered and, hence, these two users are not comparable. So, we obtain a ranking table that is the one below:

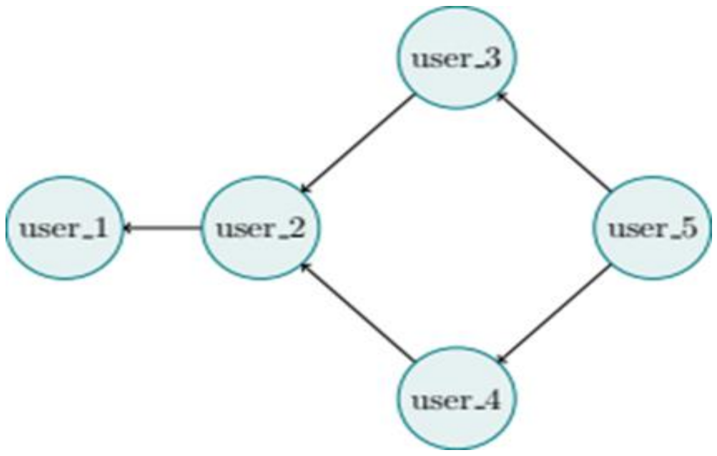
Table 14: Ranking of the users

j i	user_1	user_2	user_3	user_4	user_5
user_1		▨	▨	▨	▨
user_2			▨	▨	▨
user_3					▨

Co-funded by the Horizon 2020 Framework Programme of the European Union

user_4					▨
user_5					

The corresponding Hasse diagram is shown below.



$$\begin{tikzcd}
 & & \text{user\_3} \arrow[dl] & \\
\text{user\_1} & \text{user\_2} \arrow[l] & & \text{user\_5} \arrow[ul] \arrow[dl] \\
 & & \text{user\_4} \arrow[ul] &
\end{tikzcd}$$

Figure 7: Hasse diagram for the second set of ranked users

Such non-linearizable rankings cannot, of course, be expressed as a list, which poses an interesting challenge for the WeNet group designing the user-facing interface of WeNet apps. As a simple workaround, one can randomly break ties and forcibly linearize the ranking. Whether such a workaround is appropriate is something to be determined once WeNet data becomes available through the pilots, and the effect of a forced linearization (over a more fancy non-linear presentation in the user interface) can be empirically determined. Such data will also provide an indication of how the preferences leading to the ranking can be updated in a meaningful manner, so that the ranking converges to the requestor's true preferences.

6.3.3 Social Explanations

The central tool utilized for the social explanation component is PRUDENS. At this point, we will present a thorough demonstration of the functionality of PRUDENS in some exemplary and distinct cases in order to provide a sufficient view of its capabilities. Before presenting each case separately, we will first provide the default knowledge base that is being utilized by PRUDENS in all the cases below, as the initial theory that PRUDENS uses when a user just joins the WeNet platform, and prior to the component having access to WeNet data to further refine this initial theory. As we have already discussed, the Machine Coaching paradigm and PRUDENS are already designed and implemented to cope with this theory refinement. This functionality will be thoroughly tested when the WeNet platform is deployed, and when actual users start interacting with the Social Explanations component. Nonetheless

For simplicity of exposition, rules in the table below are given in order of priority (in those

Co-funded by the Horizon 2020 Framework Programme of the European Union

cases that two rules are conflicting). So, in the knowledge base presented in Table 15 the rule “Allergy_1” is of higher priority than e.g. rule “Suggestion_1a”. Also, as before, App_user is the task requestor and the one who will be viewing the explanations, whereas User_id is a variable / placeholder for any volunteers in the list for which explanations are sought.

Table 15: PRUDENS's default knowledge base for the Social Explanations component

Rule’s name	Rule’s body	Rule’s head
Conflict_1	conflicts(App_user,User_id,3)	(App_userpriority(User_id, 0)
Conflict_2	conflicts(App_user,User_id,2)	priority(User_id,1)
Conflict_3	conflicts(App_user,User_id,1)	priority(User_id,2)
Conflict_4	conflicts(App_user,User_id,0)	priority(User_id,3)
Allergy_1	allergy(User_id,Cuisine), cooked(Cuisine)	-suggest(User_id)
Cook_1	cooking_skills(User_id,4), pref_Cuisine_cook(User_id,Cuisine)	cooking_adequate(User_id ,Cuisine)
Cook_2	cooking_adequate(User_id, Cuisine), willing_to_cook(User_id)	cooked(Cuisine)
Cook_3	pref_Cuisine_eat(User_id, Cuisine), cooked(Cuisine), -allergy(User_id,Cuisine)	cooking_eligible(User_id)
Common_1	common_culture(App_user,User_id), common_interests(App_user,User_id), common_origins(Requestor,User_id), common_workplace(App_user,User_id)	in_common(App_user,User_id,2)
Common_2	common_culture(App_user,User_id), common_interests(App_user,User_id), common_origins(App_user,User_id)	in_common(App_user,User_id,1)

Co-funded by the Horizon 2020 Framework Programme of the European Union

Common_3	common_culture(App_user,User_id), common_interests(App_user,User_id), common_workplace(App_user,User_id)	in_common(App_user,User_id,1)
Common_4	common_culture(App_user,User_id), common_workplace(App_user,User_id), common_origins(App_user,User_id)	in_common(App_user,User_id,1)
Common_5	common_workplace(App_user,User_id), common_interests(App_user,User_id), common_origins(App_user,User_id)	in_common(App_user,User_id,1)
Suggestion_1a	experienced(User_id), in_common(App_user,User_id,2), friends(App_user,User_id), priority(User_id,3), cooking_adequate(User_id,Cuisine), cooked(Cuisine)	suggest(User_id)
Suggestion_1b	experienced(User_id), in_common(App_user,User_id,2), friends(App_user,User_id), priority(User_id,3), cooking_eligible(User_id,Cuisine), cooked(Cuisine)	suggest(User_id)
Suggestion_2a	experienced(User_id), in_common(App_user,User_id,2), somewhat_friends(App_user,User_id), priority(User_id,3), cooking_adequate(User_id,Cuisine), cooked(Cuisine)	suggest(User_id)
Suggestion_2b	experienced(User_id), in_common(App_user,User_id,2), somewhat_friends(App_user,User_id), priority(User_id,3), cooking_eligible(User_id,Cuisine), cooked(Cuisine)	suggest(User_id)
Suggestion_3a	experienced(User_id), in_common(App_user,User_id,1), friends(App_user,User_id), priority(User_id,3), cooking_adequate(User_id,Cuisine), cooked(Cuisine)	suggest(User_id)

Co-funded by the Horizon 2020 Framework Programme of the European Union

Suggestion_3b	experienced(User_id), in_common(App_user,User_id,1), friends(App_user,User_id), priority(User_id,3), cooking_eligible(User_id,Cuisine), cooked(Cuisine)	suggest(User_id)
Suggestion_4a	experienced(User_id), in_common(App_user,User_id,1), friends(App_user,User_id), priority(User_id,2), cooking_adequate(User_id,Cuisine), cooked(Cuisine)	suggest(User_id)
Suggestion_4b	experienced(User_id), in_common(App_user,User_id,1), friends(App_user,User_id), priority(User_id,2), cooking_eligible(User_id,Cuisine), cooked(Cuisine)	suggest(User_id)

Also, we provide below a list of four (4) example users that will be used for the needs of the demonstration of the explanation component. These users are simply used for the purposes of this demonstration. Similar data will be collected from real users by utilizing previous WeNet interactions and, in general, user data generated within the WeNet application.

Table 16: Attributes of the users

User	Attributes (in the form of a context)
user_1	conflicts(app_user,user_1,0); cooking_skills(user_1,4); pref_Cuisine_cook(user_1,italian); willing_to_cook(user_1); common_origins(app_user,user_1); common_origins(app_user,user_1); common_interests(app_user,user_1); common_workplace(app_user,user_1); friends(app_user,user_1); experienced(user_1);

Co-funded by the Horizon 2020 Framework Programme of the European Union

user_2	conflicts(app_user,user_2,1);
	cooked(japanese);
	pref_Cuisine_eat(user_2,japanese);
	-allergy(user_2,japanese);
	experienced(user_2);
	common_culture(app_user,user_2);
	common_interests(app_user,user_2);
	common_workplace(app_user,user_2);
	friends(app_user,user_2);

user_3	conflicts(app_user,user_3,0);
	cooking_skills(user_3,4);
	pref_Cuisine_cook(user_3,french);
	willing_to_cook(user_3);
	allergy(user_3,french);
	experienced(user_3);
	common_origins(app_user,user_3);
	common_interests(app_user,user_3);
	common_workplace(app_user,user_3);
	common_culture(app_user,user_3);
	friends(app_user,user_3);

user_4	conflicts(app_user,user_4,0);
	high_cooking_skills(user_4,vegan);
	pref_Cuisine_cook(vegan);
	willing_to_cook(user_4);
	experienced(user_4);
	common_origins(app_user,user_4);
	common_culture(app_user,user_4);
	friends(app_user,user_4);

USER_1

This user is an ideal case of a volunteer that is tightly related to the app user and is also an adequate choice for a cook in the Social Eating the app user wants to organize. These are described in the following context describing the situation at hand:

Using this context and the default knowledge base, we could have the following reasoning procedure. At first, **conflicts(app_user,user_1,0)** triggers rule **Conflict_4** which, after unification, yields **priority(user_1,3)**. One may interpret rule **Conflict_4** as “if a user has 0 conflicts with the app-user then set them to high priority for suggestion”.

Now, **cooking_skills(user_1,4)** and **pref_Cuisine_cook(user_1,italian)** trigger rule **Cook_1** which yields the literal **cooking_adequate(user_1,italian)**. This rule, in turn, may be interpreted as “if the user is a really good cook and they also cook some specific cuisine

Co-funded by the Horizon 2020 Framework Programme of the European Union

(in this case, Italian), then this user is adequate to cook this specific cuisine".

Now, the previous literal alongside with the context's literal **willing_to_cook(user_1)** using rule **Cook_2** implies **cooked(italian)**. This rule may be interpreted as "If a user who is considered adequate to cook some certain cuisine is also willing to cook at some dinner, then the cuisine that they cook will most probably be the one cooked in that dinner". For reasons of simplicity, we do not examine the case where a user is capable of cooking more than one cuisine since this would simply add to the complexity of the example without shedding more light to the way inferences are drawn by PRUDENS.

Next, the context literals **common_origins(app_user, user_1)**, **common_interests(app_user, user_1)**, **common_workplace(app_user, user_1)** trigger rule **Common_5** which returns the literal **in_common(app_user,user_1,1)**. This rule can be interpreted as "if a user is working at the same place as the app user, is sharing some common interests with the app user and also shares the same origin (e.g. hometown) as the app user, then these two users have some things in common".

Lastly, literals **experienced(user_1)**, **in_common(app_user,user_1,1)**, **cooked(italian)**, **cooking_adequate(user_1,italian), friends(app_user,user_1)**, **priority(user_1,3)** triggers rule **Suggest_2** which yields **suggest(user_1)**. The rationale behind this rule is that "if a user is an experienced one in such tasks, shares enough attributes with the app user, is appropriate to cook the cuisine that is to be cooked, shares a friendship relation with the app user and has no conflicts with the user, then this user should be suggested as a possible participant (among the available volunteers) to the dinner". So, user_1 is suggested with the explanation being the above argument.

USER_2

This is the case of a user who is considered adequate to be invited not as a cook — as it was the case with the first user — but as a simple guest who has a lot in common with the host. The user's profile is described in the second row of Table 16. Using the default knowledge base, we could have the following reasoning procedure. At first, **conflicts(app_user,user_2,1)** triggers rule **Conflict_3** which, yields **priority(user_2,2)**. This rule is interpreted in the following way "if a user has not that many conflicts with the app user, then they may be considered of some (but not high) priority for being a dinner participant".

Now, **cooked(japanese)**, **pref_Cuisine_eat(user_2,japanese)** and **-allergy(user_2,japanese)** trigger rule **Cook_3** which yields the literal **cooking_eligible(user_2,japanese)**. This rule may be interpreted in the following way: "if a user prefers some cuisine and this cuisine is the one that has been decided to be cooked in the dinner and also this user has no allergy to recipes of that cuisine then this user is eligible for further consideration".

Now, the context literals **common_culture(app_user, user_2)**, **common_interests(app_user, user_2)**, **common_workplace(app_user, user_2)** trigger rule **Common_5** which returns the literal **in_common(app_user,user_2,1)** and is

Co-funded by the Horizon 2020 Framework Programme of the European Union

interpreted as it has been explained in a previous situation. Now, literals **experienced(user_2)**, **in_common(app_user,user_2,1)** **cooked(japanese)**, **cooking_eligible(user_2, japanese)**, **friends(app_user,user_2)**, **priority(user_2,2)** triggers rule **Suggest_4b** which yields **suggest(user_2)**. This rule may be interpreted as "if the user is experienced is such tasks – i.e. social dinners – and shares with the app user a friendship relationship as well as enough things in common, while they are also of some priority and 'compatible' with the cooked cuisine then this user should be suggested as a possible participant". So, user_2 is suggested with the explanation being the above argument.

USER_3

This is the case of a user who, despite being adequate according to all other criteria, has some allergy to the cuisine that is to be cooked and, hence, is not an acceptable choice among the volunteers. Again, this user is described by the context presented in Table 16, row 3. Using the above context and the default knowledge base, we could have the following reasoning procedure. At first, literal **conflicts(app_user,user_3,0)** triggers rule **Conflict_4** which yields **priority(user_3,3)** and is interpreted as it has been explained above. Then, literals **cooking_skills(user_3,4)** and **pref_Cuisine_cook(user_3, french)** trigger rule **Cook_1** which returns **cooking_adequate(user_3, french)** with an intended interpretation as the one explained in previous example. Next, **willing_to_cook(user_3)** and **cooking_adequate(user_3, french)** trigger rule **Cook_2** which yields **cooked(french)** again with an intended interpretation as in previous cases.

Now, **common_origins(app_user, user_3)**, **common_interests(app_user, user_3)**, **common_workplace(app_user, user_3)**, **common_culture(app_user, user_3)**, triggers rule **Common_1** which returns **in_common(app_user, user_3, 2)** and is interpreted in the following way: "if a user shares common origins, common cultural background, common interests, and works at the same place with the app user then they have a lot in common". The last inferred literal alongside with **cooked(french)**, **cooking_adequate(user_3,french)**, **friends(app_user, user_3)**, **experienced(user_3)** and **priority(user_3,3)** trigger rule **Suggest_1a** which returns **suggest(user_3)**.

However, the context's literal **allergy(user_3,french)** triggers rule **Allergy_1** which yields **-suggest(user_3)** and, since rule **Allergy_1** is of higher priority than any of the conflicting **suggest( )** rules, the final advice provided is **-suggest(user_3)** with the above argumentation.

USER_4

This user is a volunteer for which no advice can be provided by the social explanations component and, hence the default action of not suggesting this user is triggered. User_4 is described in Table 16, row 4. As with the previous cases, using this context and the default knowledge base we have the following reasoning procedure: **conflicts(app_user,user_4,0)** triggers rule **Conflict_4** which yields **priority(user_4,3)** and **high_cooking_skills(user_4,vegan)** along with **pref_Cuisine_cook(vegan)** trigger rule **Cook_1** which yields **cooking_adequate(user_4)**, with the same meaning as above. Then, this literal along with **willing_to_cook(user_4)** trigger rule **Cook_2** which yields **cooked(vegan)** with an interpretation similar to the previous

Co-funded by the Horizon 2020 Framework Programme of the European Union

cases. At this point, the reasoning procedure stops since no other rule can be triggered so as to lead to either **Allergy_1** or some of the **suggest( )** rules that provide any sort of advice – more precisely, **suggest(user_4)** or **-suggest(user_4)** in this case. As it has been discussed above, in this case, where no actual suggestion is made by the explanations component, the default advice returned is to **not** suggest the corresponding volunteer to participate in the social dinner.

COUNTER-ARGUMENTATION

In the previous examples we have shed light to the way reasoning is conducted within the scope of PRUDENS. With this last example we will discuss the learning aspect of the social explanations component. For this purpose we will consider a simple scenario in which the requestor receives some explanation they find unsatisfactory and provide a simple counter-argument to PRUDENS. Prompted by this counter-argument, PRUDENS alters accordingly its knowledge base so as to conform to the requestor's argument. In general, the learning process under the presented framework should be seen as an iteration of several such "coaching instances": the requestor asks for some advice, receives an explanation and, in case they are not satisfied by it, they ask PRUDENS to alter its knowledge base accordingly.

Let us now add the following rule in our knowledge base:

Rule's name	Rule's body	Rule's head
Cook_0	cooking_skills(User_id,X), ?X<4	-suggest(User_id)

Let us also consider this rule to be of higher priority than all the **suggest( )** rules in the initial knowledge base. Also, let us consider user_5 as the user we would like to decide whether to suggest or not to the requestor, who is described by the context below:

User	Attributes (in the form of a context)

user_5	conflicts(app_user,user_5,0);
	cooking_skills(user_5,3);
	pref_Cuisine_cook(user_5,italian);
	willing_to_cook(user_5);
	common_origins(app_user,user_5);
	common_origins(app_user,user_5);
	common_interests(app_user,user_5);
	common_workplace(app_user,user_5);
	friends(app_user,user_5);
	experienced(user_5);

Now, the context's literal **cooking_skills(user_5,3)** triggers rule **Cook_0**, since **3<4**, which yields **-suggest(user_5)**. Since rule **Cook_0** is of higher priority when compared to any other conflicting rule that may suggest user_5, the final suggestion will be not to suggest user_5 with the explanation being that this user's cooking skills are lower than level 4.

Co-funded by the Horizon 2020 Framework Programme of the European Union

In this case, the requestor, seeing this explanation, may argue against it with a counter-argument as the following one (given, initially, in natural language): “I am not looking only for cooking experts to join my dinner and, hence, I would like to loosen a little bit the criteria under which I reject any candidate users due to their cooking skills and do not reject users simply because they are not expert cooks” which could be translated in the following alternative rule:

Rule’s name	Rule’s body	Rule’s head
Cook_0_alt	cooking_skills(User_id,X), ?**X<3**	-suggest(User_id)

So, the app user requests that rule **Cook_0** is substituted by the new rule **Cook_0_alt.** Under the revised knowledge base, user_5 would be finally suggested.

As a final point, it is worth pointing out that we expect that most users will provide simple arguments while coaching the social explanations component. Nonetheless, it remains an interesting challenge to determine how this human-machine interaction will be carried out. Should we expect users to understand some basic form of logic, and with the help of a good user interface enter their counterarguments? Should we use some form of structured natural language to have the user’s communicate with the component? We are currently working on both directions, and we expect that actual WeNet users will provide the necessary empirical evaluation for which of these two approaches is most useful to them.

7. CONCLUSIONS AND NEXT STEPS

This deliverable presents the main functionality of the social context builder and describes the three components it comprises. The components have discrete functionalities that aim to analyze and specify the social relations of the users in a diverse manner. The deliverable also includes a case study that demonstrates the functionality of the components that were designed and developed in the scenario of Social Eating.

The further extension of the functionality of the components is an ongoing process. The first round of design and implementation that is reported in this deliverable was meant to lead to basic solutions to the identified problems, and to identify further issues that require attention, pending the availability of interaction data through the deployment of the WeNet platform.

Key among the next steps is the diversity-aware extensions of the components, in order to properly take into account diversity aspects of the users. Specifically, in the following period and until M28 we will design and implement extensions to the components that can meaningfully be applied across different user contexts and different social interactions, while also being able to cope with partial information in the profiles of the users, while maintaining the need for privacy and transparency in the operation of the Social Context Builder.

Co-funded by the Horizon 2020 Framework Programme of the European Union

REFERENCES

[1] Vega-Redondo, F. “Complex Social Networks”. Cambridge University Press, MA, 2007.

[2] Granovetter, M. S., “The Strength of Weak Ties: A Network Theory Revisited”. Sociological Theory, 1, 201-233, 1983.

[3] Gilbert, E. Karahalios, K. “Predicting Tie Strength with Social Media”. Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, pp. 211-220, 2009.

[4] Michael, L. “Machine Coaching”. IJCAI Workshop on Explainable Artificial Intelligence (XAI), 2019.

[5] Lakkaraju, H. E. Kamar, R. Coruana, J. Leskovec, “Faithful and Customizable Explanations of Black Box Models”. AIES, 2019.

[6] Krakan, S., Humski, L., & Skočir, Z. (2018). “Determination of Friendship Intensity Between Online Social Network Users based on their Interaction”. *Tehnički vjesnik*, *25*(3), 655-662.

[7] Zhao, X., Yuan, J., Li, G., Chen, X., & Li, Z. (2012). “Relationship Strength Estimation for Online Social Networks with the Study on Facebook”. *Neurocomputing*, *95*, 89-97.

[8] Majić, M., Skorin, J., Humski, L., & Skočir, Z. (2014). “Using the Interaction on Social Networks to Predict Real Life Friendship”. In *22nd International Conference on Software, Telecommunications and Computer Networks (SoftCOM)* (pp. 378-382). IEEE.

[9] Humski, L., Pintar, D., & Vranić, M. (2017). “Exploratory Analysis of Pairwise Interactions in Online Social Networks”. *Automatika*, *58*(4), 422-428.

[10] Petricioli, L., Humski, L., Vranić, M., & Pintar, D. (2020). “Data Set Synthesis Based on Known Correlations and Distributions for Expanded Social Graph Generation”. *IEEE Access*, *8*, 33013-33022.

[11] Humski, L., Pintar, D., & Vranić, M. (2018). “Analysis of Facebook Interaction as Basis for Synthetic Expanded Social Graph Generation”. *IEEE Access*, *7*, 6622-6636.

[12] Xiang, R., Neville, J., & Rogati, M. (2010). “Modeling Relationship Strength in Online Social Networks”. In *19th International Conference on World Wide Web* (pp. 981-990).

[13] Ilić, J., Humski, L., Pintar, D., Vranić, M., & Skočir, Z. (2016). “Proof of Concept for Comparison and Classification of Online Social Network Friends based on Tie Strength Calculation Model”. In *6th International Conference on Information Society and Technology*.

[14] Burke, M., & Kraut, R. E. (2016). “The Relationship Between Facebook Use and Well-Being Depends on Communication Type and Tie Strength”. *Journal of Computer-Mediated Communication*, *21*(4), 265-281.

[15] Arnaboldi, V., Guazzini, A., & Passarella, A. (2013). Egocentric online social networks: Analysis of key features and prediction of tie strength in Facebook. *Computer Communications*, *36*(10-11), 1130-1144.

[16] Jones, J. J., Settle, J. E., Bond, R. M., Fariss, C. J., Marlow, C., & Fowler, J. H. (2013). “Inferring Tie Strength from Online Directed Behavior”. *PloS One*,

[17] Gunning, D. (2017). “Explainable Artificial Intelligence (XAI)”. *Defense Advanced* Research *Projects Agency (DARPA), nd Web*, *2*.

[18] Boutilier. C, Brafman. R, Domshlak, C. Hoos, H. and D. Poole. CP-nets: A tool for representing and reasoning with conditional ceteris paribus preference statements. Journal of AI Research, 2003.

[19] Dimopoulos, Y., Michael, L., & Athienitou, F. (2009, June). Ceteris paribus preference elicitation with predictive guarantees. In Twenty-First International Joint Conference on Artificial Intelligence.

[20] Michael, L., & Papageorgiou, E. (2013, June). An empirical investigation of ceteris paribus learnability. In Twenty-Third International Joint Conference on Artificial Intelligence.

Co-funded by the Horizon 2020 Framework Programme of the European Union