Mach Learn (2017) 106:1971–1991
DOI 10.1007/s10994-017-5647-3

Soft quantification in statistical relational learning

Golnoosh Farnadi[1,2] · **Stephen H. Bach**[3] · **Marie-Francine Moens**[2] · **Lise Getoor**[4] · **Martine De Cock**[5]

Received: 19 February 2016 / Accepted: 1 June 2017 / Published online: 12 July 2017

Abstract We present a new statistical relational learning (SRL) framework that supports reasoning with soft quantifiers, such as "most" and "a few." We define the syntax and the semantics of this language, which we call PSL^Q, and present a most probable explanation inference algorithm for it. To the best of our knowledge, PSL^Q is the first SRL framework that combines soft quantifiers with first-order logic rules for modelling uncertain relational data. Our experimental results for two real-world applications, link prediction in social trust networks and user profiling in social networks, demonstrate that the use of soft quantifiers not only allows for a natural and intuitive formulation of domain knowledge, but also improves inference accuracy.

Editors: Katsumi Inoue, Akihiro Yamamoto, and Hayato Ohwada.

✉ Golnoosh Farnadi
golnoosh.farnadi@ugent.be

Stephen H. Bach
bach@cs.stanford.edu

Marie-Francine Moens
sien.moens@cs.kuleuven.be

Lise Getoor
getoor@soe.ucsc.edu

Martine De Cock
mdecock@uw.edu

1 Department of Applied Mathematics, Computer Science and Statistics, Ghent University, Ghent, Belgium

2 Department of Computer Science, Katholieke Universiteit Leuven, Leuven, Belgium

3 Department of Computer Science, Stanford University, Stanford, CA, USA

4 Statistical Relational Learning Group, University of California, Santa Cruz, CA, USA

5 Center for Data Science, University of Washington Tacoma, Tacoma, WA, USA

1 Introduction

Statistical relational learning (SRL) has become a popular paradigm for knowledge representation and inference in application domains with uncertain data that is of a complex, relational nature. A variety of different SRL frameworks has been developed over the last decade, based on ideas from probabilistic graphical models, first-order logic, and programming languages (see e.g., Muggleton and De Raedt 1994; Richardson and Domingos 2006; Getoor and Taskar 2007). Many of these frameworks use logical formulas to express statistical dependencies over relational data. The number of elements in the data that satisfy a formula is called its *quantification*. Quantification in first-order logic, and by extension SRL, is traditionally either existential ($\exists$) or universal ($\forall$). However, there are many modeling scenarios in which softer quantifiers, such as *most* and *a few*, are more appropriate.

For example, in models for social networks it is common to include the knowledge that the behaviour, beliefs, and preferences of friends all influence each other. How this information can be incorporated depends on the expressivity of the model. In a traditional probabilistic model, a dependency might be included for each pair of friends (corresponding to a universally quantified rule), each expressing the knowledge that it is more probable that two friends share a trait in common. An often cited example in SRL contexts describing smoking behaviour among friends is:
$\forall X \forall Y\, Friends(X, Y) \rightarrow (Smokes(X) \leftrightarrow Smokes(Y))$ (Richardson and Domingos 2006). This formula states that if two people are friends, then either both of them smoke or neither of them. In this case, the probability that a person smokes scales smoothly with the number of friends that smoke. However, many traits of interest might not behave this way, but instead exhibit "tipping points" in which having a trait only becomes more probable once *most* or *some* of one's friends have that trait (e.g., smoking behaviour). Expressing this dependency requires a soft quantifier, which none of the existing SRL frameworks allow.

What sets soft quantifiers apart from universal and existential quantification is that expressions that contain them are often true to a certain degree, as opposed to either being true or false. Indeed, the degree to which a statement such as "most of Bob's friends smoke" is true, increases with the percentage of smokers among Bob's friends. This increase is not necessarily linear; in fact, a common approach to compute the truth degree of soft quantified expressions is to map percentages to the scale [0, 1] using non-decreasing piecewise linear functions (Zadeh 1983). Previous SRL work (e.g., Milch et al. 2008; Jain et al. 2010; Poole et al. 2012) has considered hard quantifiers with thresholds such as *at least k*. Soft quantifiers, on the other hand, do not impose such hard thresholds but allow smooth, gradual transitions from falsehood to truth.

Furthermore, the dependence of predicted probabilities on population size in relational models such as Markov logic networks (MLNs) and relational logistic regression is addressed in Poole et al. (2014) and Kazemi et al. (2014). Soft quantifiers not only provide the flexibility of modelling complex relations, but their semantics also do not depend on the absolute population size. Hence soft quantifiers allow us to learn a model for some population size and apply the same model to another population size without the need for changes in the model, e.g., without introducing auxiliary variables to control whether the population size grows.

Many SRL applications could benefit from the availability of soft quantifiers. Collective document classification, for instance, relies on rules such as $\forall D \forall E \forall C (Cites(D, E) \wedge Class(D, C) \rightarrow Class(E, C))$ which expresses that if documents D and E are linked (e.g., by citation), and D belongs to class C, then E belongs to C (Bach et al. 2013). Soft quantifiers can express that a document should be assigned a class if *most* of its citing documents have that class, instead of one citing document. Similarly, in collaborative filtering, one can rely on the preferred products of a user to infer the behaviour of a similar user, i.e., $\forall U_1 \forall U_2 \forall J (Likes(U_1, J) \wedge Similar(U_1, U_2) \rightarrow Likes(U_2, J))$ (Bach et al. 2013). Using soft quantifiers would allow to infer preferences of a user based on *most* of the behaviours of a similar user, or by comparing one user with *most* of the users similar to him.

In this paper we present the first SRL framework that combines soft quantifiers with first-order logic rules for modelling uncertain relational data. A brief overview of our framework is presented in Farnadi et al. (2014). We start from probabilistic soft logic (PSL) (Bach et al. 2015), an existing SRL framework that defines templates for hinge-loss Markov random fields (Bach et al. 2013), and extend it to a new framework which we call PSL^Q. As is common in SRL frameworks, in PSL a model is defined by a set of logical rules using a finite set of atoms. However, unlike other SRL frameworks whose atoms are Boolean, atoms in PSL can take continuous values in the interval [0, 1]. Intuitively, value 0 means *false* and value 1 means *true*, while any value $v \in [0, 1]$ represents a partial degree of truth.

Our approach differs from existing research on quantifiers for logical reasoning in various ways. Studies on quantifiers in probabilistic logic settings deal with Boolean atoms (Lowd and Domingos 2007; Beltagy and Erk 2015; Van den Broeck et al. 2013), while in this paper atoms take on continuous values. The literature on fuzzy logic contains a fair amount of work on reasoning with continuous values (e.g., Prade et al. 2003; Cao et al. 2002), including the use of soft quantifiers (Bobillo and Straccia 2008), yet, to the best of our knowledge, there is no prior work on such soft quantifiers in the SRL community.

An early version of this paper appeared in Farnadi et al. (2015). In addition to the inclusion of proofs in Sect. 4, this paper extends this earlier work with more extensive evaluation, including a new application presented in Sect. 6. After recalling the preliminaries of PSL in Sect. 2, in Sect. 3 we introduce PSL^Q, a new SRL framework that supports reasoning with soft quantifiers, such as “most” and “a few.” Because this expressivity pushes beyond the capabilities of PSL, in Sect. 4 we introduce new inference and weight learning algorithms for PSL^Q. Finally, as a proof of concept, we present two PSL^Q models, one for predicting trust in social networks and another one for user profiling in social networks. We show that our PSL^Q model more accurately predicts trust in social networks than the current state-of-the-art approach in Sect. 5. Similarly, our PSL^Q model significantly outperforms its sibling PSL model in inferring age and gender in social networks in Sect. 6.

2 PSL: probabilistic soft logic

In this section, we review the syntax and semantics of probabilistic soft logic (PSL), a probabilistic programming language with a first-order logical syntax. PSL is a probabilistic programming language for defining hinge-loss Markov random fields (Bach et al. 2015) that has been used in various domains, including bioinformatics (Fakhraei et al. 2014; Sridhar et al. 2016), knowledge graph identification (Pujara et al. 2013), recommender systems (Kouki et al. 2015), natural language processing (Beltagy et al. 2014; Deng and Wiebe 2015; Ebrahimi et al. 2016), information extraction (Liu et al. 2016), information retrieval (Alshukaili et al. 2016), and social network analysis (Huang et al. 2013; West et al. 2014), among many others.

Multiple interpretations of PSL semantics are possible. In this paper, we use the Łukasiewicz logic (Klir and Yuan 1995) interpretation, because soft degrees of truth naturally complement soft quantifiers. We also focus on the subset of PSL syntax relevant to our approach. See Bach et al. (2015) for a full explanation of PSL syntax, semantics, and possible interpretations.

We start by defining atoms.

Definition 1 An **atom** is an expression of the form $p(a_1, a_2, \ldots, a_n)$ where p is a **predicate symbol**, and each argument $a_1, a_2, \ldots, a_n$ is either a constant or a variable. The finite set of all possible substitutions of a variable to a constant for a particular variable a_i is called its **domain** D_{a_i}. If all variables in $p(a_1, a_2, \ldots, a_n)$ are substituted by some constant from their respective domain, then we call the resulting atom a **ground atom**. We call $\neg p(a_1, a_2, \ldots, a_n)$ a **negated atom** which is the negation of $p(a_1, a_2, \ldots, a_n)$.

Under the Łukasiewicz logic interpretation, PSL atoms represent soft degrees of truth.

Definition 2 An **interpretation** I is a mapping that associates a truth value $I(p) \in [0, 1]$ to each ground atom p.

For example, $I(Knows(Alice, Bob)) = 0.7$ indicates that Alice knows Bob to degree 0.7. We next define programs and rules.

Definition 3 A **PSL program** is a collection of PSL rules. A **PSL rule** r is an expression of the form:

$$\lambda_r : T_1 \wedge T_2 \wedge \ldots \wedge T_w \rightarrow H_1 \vee H_2 \vee \ldots \vee H_l \tag{1}$$

where $T_1, T_2, \ldots, T_w, H_1, H_2, \ldots, H_l$ are atoms or negated atoms and $\lambda_r \in \mathbb{R}^+ \cup \infty$ is the weight of the rule r. We call $T_1 \wedge T_2 \wedge \ldots \wedge T_w$ the body of r (r_{body}), and $H_1 \vee H_2 \vee \ldots \vee H_l$ the head of r (r_{head}). **Grounding** a PSL rule r means instantiating all the variables with constants from their domains.

Rules 1–9 in Table 1 and rules 1–4 in Table 7 are examples of PSL programs. Conjunction $\wedge$ is interpreted by the Łukasiewicz t-norm, disjunction $\vee$ by the Łukasiewicz t-conorm, and negation $\neg$ by the Łukasiewicz negator.

Definition 4 The Łukasiewicz t-norm ($\tilde{\wedge}$) and the corresponding t-conorm ($\tilde{\vee}$) and negator ($\tilde{\neg}$) are defined as follows. For $m, n \in [0, 1]$ we have $m \tilde{\wedge} n = \max(0, m + n - 1)$, $m \tilde{\vee} n = \min(m + n, 1)$ and $\tilde{\neg} m = 1 - m$.

The ˜ indicates the relaxation over Boolean values. Using Definition 4, we can extend the interpretation of atoms to more complex formulas in Łukasiewicz logic.

Definition 5 Given an interpretation I, and p_1 and p_2 ground atoms, we have $I(p_1 \wedge p_2) = I(p_1) \tilde{\wedge} I(p_2)$, $I(p_1 \vee p_2) = I(p_1) \tilde{\vee} I(p_2)$ and $I(\neg p_1) = \tilde{\neg} I(p_1)$.

Remark 1 In Łukasiewicz logic, the expression $B \tilde{\rightarrow} H$ where $\tilde{\rightarrow}$ is implication, is logically equivalent to $\tilde{\neg} B \tilde{\vee} H$, thus the interpretation of a grounded PSL rule r is as follows:

$$I(r) = I(r_{body} \rightarrow r_{head}) = \tilde{\neg} I(r_{body}) \tilde{\vee} I(r_{head}) \tag{2}$$

Example 1 Consider the grounded PSL rule $Knows(Alice, Bob) \rightarrow Trusts(Alice, Bob)$, and suppose $I(Knows(Alice, Bob)) = 0.5$ and $I(Trusts(Alice, Bob)) = 0.4$. Then $I(r_{body}) = 0.5$, $I(r_{head}) = 0.4$ and $I(r) = 0.9$, i.e. rule r is satisfied to degree 0.9 under interpretation I.

The probability of truth value assignments in PSL is determined by the rules' **distance to satisfaction**.

Definition 6 The distance to satisfaction $d_r(I)$ of a rule r under an interpretation I is defined as:

$$d_r(I) = \max\{0, I(r_{body}) - I(r_{head})\} \tag{3}$$

By using Remark 1, one can show that a rule r is fully satisfied, i.e. satisfied to degree 1, when the truth value of its head is at least as high as the truth value of its body. Thus, the closer the interpretation of a grounded rule is to 1, the smaller its distance to satisfaction.

Remark 2 The distance to satisfaction of a PSL rule is equivalent to its negated interpretation: $d_r(I) = \tilde{\neg} I(r)$.

Example 2 Consider the example PSL rule in Example 1. Let's assume that we know $I(Knows(Alice, Bob)) = 0.7$ then to satisfy the rule, $I(Trusts(Alice, Bob)) \geq 0.7$.

If *Trusts(Alice,Bob)* is true to 0.5, then the rule is satisfied to degree $1 - 0.7 + 0.5 = 0.8$. Consider the rule r in Example 1: $d_r(I) = 0.1$, which is equal to $\tilde{\neg} I(r) = 1 - 0.9 = 0.1$. If $I(\textit{Trusts(Alice,Bob)}) = 0.6$, then the rule is satisfied (to degree 1) and the distance to satisfaction is 0.

A PSL program, i.e., a set of PSL rules, induces a distribution over interpretations I. Let R be the set of all grounded rules, then the probability density function is:

$$f(I) = \frac{1}{Z} \exp\left[-\sum_{r \in R} \lambda_r (d_r(I))^p\right] \tag{4}$$

$$Z = \int_I \exp\left[-\sum_{r \in R} \lambda_r (d_r(I))^p\right] \tag{5}$$

where λ_r is the weight of rule r, Z is a normalization constant, and $p \in \{1, 2\}$ provides a choice of two different loss functions, $p = 1$ (i.e., linear) favors interpretations that completely satisfy one rule at the expense of higher distance from satisfaction for conflicting rules, and $p = 2$ favors interpretations that satisfy all rules to some degree (i.e, quadratic). These probabilistic models are instances of hinge-loss Markov random fields (HL-MRF). For further explanation we refer to Bach et al. (2015).

Inference in PSL is performed by finding the most probable explanation (MPE) over a set of given evidence, which is equivalent to maximizing the density function in Eq. 4. For example, in a trust propagation application, given a set of trust relations between users, the goal of MPE inference is to infer the trust degree between all users. Later in this paper we discuss the inference phase in PSL in more detail.

3 PSL^Q: PSL with soft quantifiers

The statement that most of Bob's friends are smokers is expressed in PSL^Q with the **quantifier expression** $Most(T, Friend(T, Bob), Smokes(T))$. The general form of such quantifier expressions is $Q(V, F_1[V], F_2[V])$, in which Q denotes a soft quantifier, V denotes the variable over which the quantification ranges, and $F_1[V]$ and $F_2[V]$ are formulas containing V. These formulas can be atoms as well as negations, conjunctions or disjunctions of formulas.

Definition 7 A **quantifier expression** is an expression of the form

$$Q(V, F_1[V], F_2[V]) \tag{6}$$

where Q is a soft quantifier, and $F_1[V]$ and $F_2[V]$ are formulas containing a variable V. A **grounded quantifier expression** is obtained by instantiating all variables with constants from their domains except for V.

Example 3 Consider the two formulas $Knows(X, T)$ and $Trust(X, T)$, then $Most(T, Knows(X, T), Trusts(X, T))$ is a quantifier expression. By substituting X with Alice, we obtain the grounded quantifier expression $Most(T, Knows(Alice, T), - Trusts(Alice, T))$ which can be read as "Alice trusts most of the people she knows".

Definition 8 A **PSL^Q program** is a collection of PSL^Q rules. A **PSL^Q rule** has the same form as a PSL rule defined in Eq. 1 except that $T_1, T_2, \ldots, T_k$ are either atoms, negated atoms or **quantifier expressions**. Grounding a PSL^Q rule means instantiating all the variables with constants from their domain except for all the variables V in quantifier expressions $Q(V, F_1[V], F_2[V])$.

Rules 10–14 in Table 2 and rules 5–8 in Table 8 are examples of PSL^Q rules with quantifier expressions.

Analogously to how the interpretation of rules in PSL relies on operations from Łukasiewicz logic (see Definition 4), the interpretation of quantifier expressions in PSL^Q relies on quantifier mappings.

Definition 9 A **quantifier mapping** $\tilde{Q}$ is a $[0, 1] \to [0, 1]$ mapping. If $\tilde{Q}$ is non-decreasing and satisfies the boundary conditions $\tilde{Q}(0) = 0$ and $\tilde{Q}(1) = 1$, it is called a **coherent** quantifier mapping (Delgado et al. 2000).

We assume that for every soft quantifier Q an appropriate quantifier mapping $\tilde{Q}$ can be defined, i.e. a function that represents the meaning of Q.

Using two thresholds $\alpha \in [0, 1]$ and $\beta \in [0, 1]$, where $\alpha \leq \beta$, the following equation defines a parametrized family of such quantifier mappings:

$$\tilde{Q}_{[\alpha,\beta]}(x) = \begin{cases} 0 & \text{if } x < \alpha \\ \frac{x-\alpha}{\beta-\alpha} & \text{if } \alpha \leq x < \beta \\ 1 & \text{if } x \geq \beta \end{cases} \tag{7}$$

Figure 1 depicts a possible coherent quantifier mapping for the soft quantifier "a few" as $\tilde{Q}_{Few} = \tilde{Q}_{[0.1,0.4]}$ and for the soft quantifier "most" as $\tilde{Q}_{Most} = \tilde{Q}_{[0.25,0.75]}$. Note how $\tilde{Q}_{Few}$ is more relaxed than $\tilde{Q}_{Most}$. For example, using these mappings, the statement "a few friends of Bob smoke" is true to degree 1 as soon as 40% of Bob's friends are smokers, while 75% of Bob's friends are required to be smokers for the statement "most friends of Bob smoke" to be fully true. The evaluation section contains a detailed analysis on the effect of the choice of the thresholds α and β on the results obtained with MPE inference.

An interesting observation is that in practice friendship is not necessarily a black-and-white matter, i.e., people can be friends to varying degrees. For instance, $I(Frie\!-\!nd(Bob, Alice)) = 1$ and $I(Friend(Bob, Chris)) = 0.2$ denote that under interpretation I, Alice is a very close friend of Bob, while Chris is a more distant friend. Similarly, Chris might be a heavy smoker, while Alice might be only a light smoker. All these degrees can and should be taken into account when computing the truth degree of statements such as "a few friends of Bob smoke" and "most friends of Bob smoke" (Zadeh 1983).

Remark 3 Zadeh (1983) suggested to calculate the truth value of "Q A's are Bs", with $A : X \to [0, 1]$ and $B : X \to [0, 1]$ fuzzy sets, as:

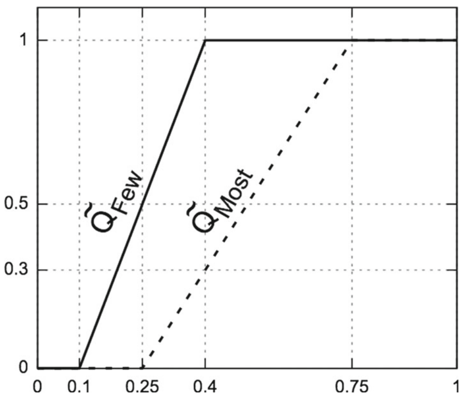

Fig. 1 Examples of quantifier mappings

$$\tilde{Q}\left(\frac{|A \cap B|}{|A|}\right) \tag{8}$$

where $A \cap B$ is a fuzzy set defined as:

$$A \cap B : X \to [0, 1] : x \mapsto A(x) \tilde{\wedge} B(x) \tag{9}$$

Remark 4 The **cardinality** of a fuzzy set $S : X \to [0, 1]$ is defined as:

$$|S| = \sum_{x \in X} S(x) \tag{10}$$

Definition 10 For a given interpretation I, the interpretation of a grounded quantifier expression $Q(V, F_1[V], F_2[V])$ is defined as

$$I(Q(V, F_1[V], F_2[V])) = \tilde{Q}\left(\frac{\sum_{x \in D_V} I(F_1(x)) \tilde{\wedge} I(F_2(x))}{\sum_{x \in D_V} I(F_1(x))}\right)$$

with $\tilde{Q}$ a quantifier mapping modelling Q.

Example 4 Let's consider an interpretation I in a sample trust network as shown in Fig. 2. Nodes represent users and each edge represents the trust relation between two users. Since a trust relation is asymmetric, the direction of the trust relation is shown with an arrow. The degree of the trust links are shown with a value under/above the links, e.g., $I(Trusts(Alice, Ann)) = 0.9$.

To calculate $I(Most(X, Trusts(Alice, X), Trusts(X, Bob)))$, i.e. the degree to which most trustees of Alice trust Bob under the interpretation I shown in Fig. 2, we calculate

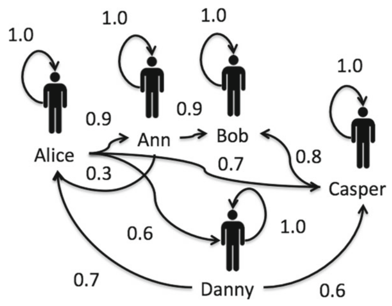

Fig. 2 Sample trust network between five users

$\sum_{x\in D}(\mathrm{I}(\mathrm{Trusts}(Alice, x))\tilde{\wedge}\, \mathrm{I}(\mathrm{Trusts}(x, Bob))) = 1.3$ and $\sum_{x\in D} \mathrm{I}(\mathrm{Trusts}(Alice, x)) = 3.2$ so $\tilde{Q}\left(\frac{1.3}{3.2}\right) \sim \tilde{Q}(0.41)$. By using the quantifier expression mapping of "most" in Fig. 1 we obtain $\tilde{Q}_{[0.25,0.75]}(0.41) = 0.32$. Thus, $I(Most(X, Trusts(Alice, X), Trusts(X, Bob)))$ states that "most trustees of Alice trust Bob" to degree 0.32.

4 Inference and weight learning in PSL^Q

Expressing soft quantifiers pushes beyond the capabilities of inference and weight learning methods in PSL. In this section, we introduce new methods for inference based on the most probable explanation inference method (MPE inference) and weight learning with maximum-likelihood estimation (MLE) in PSL^Q.

4.1 Inference

The goal of MPE "most probable explanation" inference is to find the most probable truth assignments I_{MPE} of unknown ground atoms given the evidence which is defined by the interpretation I. Let X be all the evidence, i.e., X is the set of ground atoms such that $\forall x \in X, I(x)$ is known, and let Y be the set of ground atoms such that $\forall y \in Y, I(y)$ is unknown. Then we have

$$I_{MPE}(Y) = arg \max_{I(Y)} P(I(Y)|I(X)) \tag{11}$$

and by Eq. 4 it follows that the goal of optimization is to minimize the weighted sum of the distances to satisfaction of all rules.

Remark 5 Suppose we want to optimize a $f : [0, 1]^n \rightarrow [0, 1]$ function consisting of applications of only piecewise linear functions, fractions of piecewise linear functions, min : $[0, 1]^2 \rightarrow [0, 1]$ and max : $[0, 1]^2 \rightarrow [0, 1]$. We can transform such an optimization problem as follows. For every expression of the form $\min(\phi, \psi)$, we introduce a variable $v_{\min(\phi,\psi)}$ and add the constraints $0 \leq \phi, \psi, v_{\min(\phi,\psi)} \leq 1$, $v_{\min(\phi,\psi)} \leq \phi$ and $v_{\min(\phi,\psi)} \leq \psi$. Similarly, for every expression of the form $\max(\phi, \psi)$, we introduce a variable $v_{\max(\phi,\psi)}$ and add the constraints: $0 \leq \phi, \psi, v_{\max(\phi,\psi)} \leq 1$, $v_{\max(\phi,\psi)} \geq \phi$, and $v_{\max(\phi,\psi)} \geq \psi$.

Define the function g as the original function f but all minima and maxima are replaced by their corresponding variables. Optimizing f is then equivalent to optimizing g under these constraints.

Proposition 1 *MPE inference for a PSL program is equivalent to solving a linear optimization problem.*

Proof The goal of optimization in PSL is to minimize the weighted sum of the distances to satisfaction of all rules, therefore we have:

$$I_{MPE}(Y) = arg \max_{I(Y)} - \sum_{r\in R} \lambda_r(d_r(I(X, Y))) \tag{12}$$

By the particular piecewise linear form of $d_r(I)$ (see Definition 6) and Remark 5, finding an MPE assignment is a linear optimization problem, which is solvable in polynomial time. Note that we consider the linear form which is $p = 1$ in Eq. 4 throughout this section. However, PSL supports $p = 2$ where finding an MPE assignment is a convex optimization problem, which is solvable in polynomial time.

PSL also supports aggregates, i.e., random variables with values determined by other variables. To preserve convexity, however, standard PSL only supports linear aggregates. □

Definition 11 An **aggregate** is a $[0, 1]^n \to [0, 1]$ mapping. If it is a linear mapping, it is called a **linear aggregate**, otherwise it is called a **non-linear aggregate**.

As an example, $f : [0, 1]^n \to [0, 1] : (t_1, \ldots, t_n) \mapsto \frac{t_1+t_2+\cdots+t_n}{n}$ is a linear aggregate. A PSL^Q program allows expressions that contain quantifier expressions. Since the interpretation of a grounded quantifier expression (See Definition 10) is based on a non-linear aggregate, finding a MPE assignment of a PSL^Q program with quantifier expressions is beyond the capabilities of the standard PSL MPE-solver. To deal with this, we will first categorize different types of grounded quantifier expressions, given the interpretation I denoting the evidence.

Definition 12 A grounded quantifier expression $Q(V, F_1[V], F_2[V])$, where for every $s \in D_V$, it holds that all ground atoms in the formulas $F_1[s]$ and $F_2[s]$ are in X, is called a **fully observed grounded quantifier expression (FOQE)**.

For instance, in a social network where the age and the friends of all users are known, by grounding $Most(B, Friend(A, B), Young(B))$, we obtain FOQEs. Note that for a FOQE $Q(V, F_1[V], F_2[V])$, we have that $I(Q(V, F_1[V], F_2[V]))$ is a known value in [0, 1].

Proposition 2 *MPE inference for a PSL^Q program with grounded quantifier expressions limited to type FOQE is equivalent to solving a linear optimization problem.*

Proof By replacing all the FOQEs in the program with new variables, where the value of these variables are known and are in [0, 1], there is no difference between MPE inference for a PSL^Q program without any quantifier expressions and for a PSL^Q program with grounded quantifier expressions of type FOQE. The only difference is the processing time needed to calculate the value of each quantifier expressions, which is $O(|X|)$. □

Definition 13 A grounded quantifier expression $Q(V, F_1[V], F_2[V])$, where for every $s \in D_V$, it holds that all ground atoms in the formula $F_1[s]$ are in X and there exists $t \in D_V$ such that at least one ground atom in the formula $F_2[t]$ is in Y, is called a **partially observed grounded quantifier expression of type one (POQE$^{(1)}$)**.

Suppose all friendship relations are known and the goal is to infer the age of all users based on the age of some, then by grounding $Most(B, Friend(A, B), Young(B))$, we obtain POQE$^{(1)}$s. By grounding the rules 5–8 in Table 8, we obtain examples of POQE$^{(1)}$s. Note that for a POQE$^{(1)}$ $Q(V, F_1[V], F_2[V])$, we have that $I(Q(V, F_1[V], F_2[V])) = \tilde{Q}(f(Y))$ where f is a piecewise linear function in variables belonging to Y.

Definition 14 A grounded quantifier expression $Q(V, F_1[V], F_2[V])$, for which there exists $t \in D_V$ such that at least one ground atom in the formula $F_1[t]$ is in Y, is called a **partially observed grounded quantifier expression of type two (POQE$^{(2)}$)**.

Note that for a POQE$^{(2)}$ $Q(V, F_1[V], F_2[V])$, we have that $I(Q(V, F_1[V], F_2[V])) = \tilde{Q}(f(Y))$ where f is a fraction of piecewise linear functions in variables belonging to Y. In link prediction applications, such as trust link prediction, we mostly deal with POQE$^{(2)}$s. By grounding the rules 10–14 in Table 2 using unknown trust relations, we obtain complex examples of POQE$^{(2)}$s.

In the following proposition we give an equivalent definition for the membership function in Eq. 7. By applying Remark 5 we will then be able to show that a PSL^Q program can be transformed to an optimization program with an objective function that is a weighted sum of linear and fractional linear functions.

Proposition 3 *The membership-function defined in Eq. 7 where $\alpha \in [0, 1]$, $\beta \in [0, 1]$, and $\alpha < \beta$ can be rewritten as:*

$$\tilde{Q}_{[\alpha,\beta]}(x) = \max\left(0, \frac{x-\alpha}{\beta-\alpha}\right) + \min\left(\frac{x-\alpha}{\beta-\alpha}, 1\right) - \frac{x-\alpha}{\beta-\alpha} \tag{13}$$

Proof By checking the boundary conditions $x < \alpha$, $\alpha \leq x < \beta$ and $x \geq \beta$, one can show that Eq. 13 is equivalent to Eq. 7. □

Proposition 4 *MPE inference for a PSL^Q program with grounded quantifier expressions limited to type FOQE and POQE$^{(1)}$ is equivalent to solving a linear optimization problem.*

Proof A grounded quantifier expression $Q(V, F_1[V], F_2[V])$ of type POQE$^{(1)}$ is of the form:

$$\tilde{Q}\left(\frac{\sum_{x \in D_V} \max(0, C_x + I(F_2(x)) - 1)}{\sum_{x \in D_V} C_x}\right)$$

where C_x is the outcome of calculating the value of the grounded formula $F_1(x)$, i.e., $C_x = I(F_1(x))$. Since all ground atoms in the formula $F_1(x)$ are in X, C_x is a constant value in [0, 1]. Let's assume $\lambda = \frac{1}{\sum_{x \in D_V} C_x}$ and $\gamma = \frac{1}{\beta-\alpha}$. Both λ and γ are constant values. By using Proposition 3 the POQE$^{(1)}$ is converted to:

$$\begin{aligned}
&= \max\left(0, \gamma \times \left(\left(\lambda \times \sum_{x \in D_V} \max(0, C_x + I(F_2(x)) - 1))\right) - \alpha\right)\right)\\
&\quad + \min\left(\gamma \times \left(\left(\lambda \times \sum_{x \in D_V} \max(0, C_x + I(F_2(x)) - 1))\right) - \alpha\right), 1\right)\\
&\quad - \gamma \times \left(\left(\lambda \times \sum_{x \in D_V} \max(0, C_x + I(F_2(x)) - 1))\right) - \alpha\right)
\end{aligned}$$

By using Remark 5, we first introduce new variables for all the inner expressions of the form $\max(0, C_x + I(F_2(x)) - 1))$ and replace them by their corresponding variables. Then we introduce new variables for the two outer maxima and minima expressions and replace them with their corresponding variables. As a result, each grounded quantifier expression of type POQE$^{(1)}$ is replaced with a linear expression and a set of linear constraints. MPE inference for the PSL^Q program limited to quantifier expressions of type FOQE and POQE$^{(1)}$ is then equivalent to minimizing a linear function under a set of linear constraints coming from replacing the minima and maxima in calculating the distance to satisfaction of grounded rules. □

MPE inference for a PSL^Q program with grounded quantifier expressions of type POQE$^{(2)}$ is solved with a sequence of linear optimization problems. Note that this is only a worst case scenario: if the grounded PSL^Q program has no POQE$^{(2)}$s then we obtain a linear program. In the presence of POQE$^{(2)}$s, we use a transformation similar to the approach of Isbell and Marlow (1956) to replace a linear fractional program (LFP) by a set of linear programs by establishing a convergent iterative process. The linear program at each iteration is determined by optimization of the linear program at the previous iteration. Note that our PSL^Q program with grounded quantifier expressions of type POQE$^{(2)}$ has an objective function that is a weighted sum linear and fractional linear functions, and is subject to linear equality and inequality constraints. Transformations of a LFP to a LP such as Charnes/Cooper transformation (Charnes and Cooper 1962) are not suitable for our problem domain.

Algorithm 1 Iterative MPE inference in PSL^Q

```
Require: PSL$^Q$ program $P$, evidence variables $X$ and random variables $Y$
1: $R \leftarrow \emptyset$
2: $I^{(0)}(Y) \leftarrow 0$
3: for $i := 1$ to $k$ do
4:   for $r \in P$ do
5:     $R_g \leftarrow$ ground($r$)
6:     for $r_g \in R_g$ do
7:       for every $Q$ of type POQE$^{(2)}$ in $r_g$ do
8:         $I(Q) \leftarrow \tilde{Q}(I(X) \cup I^{i-1}(Y))$
9:       end for
10:      $d_{r_g}(I) \leftarrow 1 - I(r_g)$
11:      if not $d_{r_g}(I) = 0$ then
12:        $R \leftarrow R \cup r_g$
13:      end if
14:    end for
15:  end for
16:  $f(I) \leftarrow$ generate($R$)
17:  $G(I) \leftarrow$ transform($f(I)$)
18:  $I^{(i)}(Y) \leftarrow$ optimize($G(I)$)
19: end for
```

The algorithm we propose for MPE inference (Algorithm 1) starts by initializing the set of all grounded rules (i.e., R) to an empty set and all random variables to zero (i.e., line 2). Then, an iterative process starts by grounding all rules in the PSL^Q program (i.e., line 3–5). For every grounded quantifier expression Q of type POQE$^{(2)}$, the value of Q is initialized by calculating the value over the known values ($I(X)$) and the current setting of the unknown values ($I^{(0)}(Y)$). In the algorithm, we use the notation $\tilde{Q}(I(X) \cup I^{i-1}(Y))$ to denote this new interpretation of Q at iteration i (i.e., line 7–9). For each rule r_g we then calculate the distance to satisfaction (i.e., line 10). Note that $I(r_g)$ and hence also $d_{r_g}(I)$ can be piecewise linear functions in Y, but here $d_{r_g}(I)$ does not contain fractions of piecewise linear functions since we calculate values for the POQE$^{(2)}$s. Next, we exclude the satisfied grounded rules (i.e., we exclude rules r_g such that $d_{r_g}(I) = 0$) from the optimization since their values will not change the optimization task (i.e., line 11–13). For the optimization task, $f(I)$ (Eq. 4) is calculated using the distance to satisfaction of all grounded rules (i.e., line 16). Since $f(I)$ does not contain fractions of piecewise linear functions, it can be transformed to a linear program (i.e., line 17). Finally, the inner optimization in PSL^Q is solved with PSL's scalable, parallelizable message-passing inference algorithm (Bach et al. 2013) (i.e., line 18). In each iteration, the values of the Qs get updated by the most probable assignment of random variables in the previous iteration ($I(X) \cup I^{(i-1)}(Y)$) (i.e., line 8). This process is iteratively repeated for a fixed number of times (i.e., k).

Note that by considering the quadratic form which is $p = 2$ in Eq. 4, MPE inference for a PSL^Q program with grounded quantifier expressions of type FOQE is similar to the MPE inference in a PSL program. However, MPE inference for a PSL^Q program with grounded quantifier expressions of either type POQE$^{(1)}$ or type POQE$^{(2)}$ is based on our proposed iterative MPE algorithm. The inner optimization in PSL^Q of the quadratic form is solved with PSL's inference algorithm (Bach et al. 2013) with squared hinge-loss functions.

4.2 Weight learning

The goal of weight learning based on maximum likelihood estimation (MLE) is to maximize the log likelihood of the rules' weight based on the training data in Eq. 4. Hence, the partial derivatives of log likelihood with respect to λ_i of rule $r_i \in R$ are

$$-\frac{\delta \log(f(I))}{\delta\lambda_i} = E_\lambda\left[\sum_{r\in R_g i}(d_r(I))^p\right] - \sum_{r\in R_g i}(d_r(I))^p \tag{14}$$

with E_λ the expected value under the distribution defined by λ, and $R_g i$ is the set of grounded rules of rule r_i. The optimization is based on the voted perception algorithm (Collins 2002), in which approximation is done by taking fixed-length steps in the direction of gradient and averaging the points after all steps; out of the scope steps are projected back into the feasible region. To make the approximation tractable, a MPE approximation is used that replaces the expectation in the gradient with the corresponding values in the MPE state. We use our proposed MPE approach for transforming POQE$^{(1)}$s and POQE$^{(2)}$s in our MLE algorithm. We omit the pseudocode of the MLE algorithm for a PSL^Q program to save space.

To investigate the effects of using soft quantifiers on real-world applications, we explore two applications. The first application is a link prediction task in which we have grounded quantifier expressions of type POQE$^{(2)}$ and test our iterative MPE inference in Sect. 5. The second application is a node labeling application in which we apply our transformed MPE inference using grounded quantifier expressions of type POQE$^{(1)}$ in Sect. 6.

5 Link prediction: social trust link prediction

Studies have shown that people tend to rely more on recommendations from people they trust than on online recommender systems which generate recommendations based on anonymous people similar to them. This observation has generated a rising interest in trust-enhanced recommendation systems (Victor et al. 2011). The recommendations generated by these systems are based on an (online) trust network, in which members of the community express whether they trust or distrust each other. In practice these networks are sparse because most people are connected to relatively few others. Trust-enhanced recommendation systems therefore rely on link prediction.

In Huang et al. (2013), trust relations between social media users are modeled and predicted using a PSL program based on the structural balance theory (Heider 1958). Structural balance theory implies the transitivity of a relation between users. Based on this theory, users are more prone to trust their neighbors in the network rather than unknown other users. Bach et al. (2013)[1] evaluated the PSL program based on the structural balance theory on data from *Epinions*,[2] an online consumer review site in which users can indicate whether they trust or distrust each other. Throughout this section, we will use the same sample of Epinions (Leskovec et al. 2010). The sample dataset includes 2000 users with 8675 relations, namely 7974 trust relations and only 701 distrust relations.

We perform eightfold cross-validation and to evaluate the results, we use three metrics, *AUC*: the area under the receiver operating characteristic curve, *PR*+: the area under the precision-recall curves for trust relations, and *PR*−: the area under the precision-recall curves

[1] Source code available at http://psl.linqs.org/.

[2] www.epinions.com.

Table 1 PSL rules for social trust link prediction

	Transitive rules
$R\#1$	$Knows(A,B) \wedge Trusts(A,B) \wedge Knows(B,C) \wedge Trusts(B,C) \wedge Knows(A,C) \rightarrow Trusts(A,C)$
$R\#2$	$Knows(A,B) \wedge \neg Trusts(A,B) \wedge Knows(B,C) \wedge Trusts(B,C) \wedge Knows(A,C) \rightarrow \neg Trusts(A,C)$
$R\#3$	$Knows(A,B) \wedge Trusts(A,B) \wedge Knows(B,C) \wedge \neg Trusts(B,C) \wedge Knows(A,C) \rightarrow \neg Trusts(A,C)$
$R\#4$	$Knows(A,B) \wedge \neg Trusts(A,B) \wedge Knows(B,C) \wedge \neg Trusts(B,C) \wedge Knows(A,C) \rightarrow Trusts(A,C)$
	Cyclic rule
$R\#5$	$Knows(A,B) \wedge Trusts(A,B) \wedge Knows(B,C) \wedge Trusts(B,C) \wedge Knows(C,A) \rightarrow Trusts(C,A)$
	Complementary rules
$R\#6$	$Knows(A,B) \wedge Knows(B,A) \wedge Trusts(B,A) \rightarrow Trusts(A,B)$
$R\#7$	$Knows(A,B) \wedge Knows(B,A) \wedge \neg Trusts(B,A) \rightarrow \neg Trusts(A,B)$
$R\#8$	$Knows(A,B) \wedge Average(\{Trusts\}) \rightarrow Trusts(A,B)$
$R\#9$	$Knows(A,B) \wedge Trusts(A,B) \rightarrow Average(\{Trusts\})$

for distrust relations. In each fold, we first learn the weights of the rules based on 7/8 of the trust network and then apply the learned model on the remaining 1/8 to infer the trust/distrust relations. Bach et al. used the program of Huang et al. (2013) which is composed of twenty PSL rules in order to predict the degree of trust between two individuals. Sixteen rules from these rules encode possible stable triangular structures involving the two individuals and a third one. For example, an individual is likely to trust people his or her friends trust. The program of Huang et al. (2013) is used to predict unobserved truth-values of $Trusts(A, B)$ for pairs of individuals. The results of this program are shown in the first line in Table 3.

In this paper, we propose a program based on 4 transitive rules (rules 1–4 in Table 1) and one rule which models the cyclic relation between 3 users (rule 5 in Table 1). Rules 6–9 in Table 1 are complementary rules for which we refer to Huang et al. (2013) for further explanation. The atom $Average(\{Trusts\})$ in rules 8 and 9 is a constant which refers to the global average value of observed trust scores. This atom is useful for the disconnected parts of the trust network without any known trust relation. These four rules are also used in the PSL program of Bach et al. (2013).

To investigate whether we can improve the accuracy of the predictions by introducing rules with soft quantifier expressions, we construct PSL^Q rules based on a triad relation over a set of users instead of a third party (rules 10–14) in Table 2. The full PSL^Q program then consists of all rules displayed in Tables 1 and 2 .

We examine what happens when changing the thresholds for the quantifier mappings $\tilde{Q}$ (Eq. 7). We have investigated ten different quantifier mappings by changing the values of α and β by steps of 0.25. In this way, we obtain ten different PSL^Q programs. For every program, we applied Algorithm 1 for all $k \in \{1, 2, \ldots, 10\}$. Note that for $k = 1$, since our PSL^Q program only contains quantifier expressions of type POQE$^{(2)}$, the output of the MPE inference is equivalent to the output generated by a PSL^Q program with only FOQEs by ignoring the unknown values. Figure 3 presents changes of the three metrics of these ten PSL^Q programs with different quantifier mappings. All ten PSL^Q programs outperform the PSL program (shown with a line) in all iterations and in all three metrics, except for the

Table 2 PSL^Q rules for social trust link prediction

	Transitive rules using soft quantifier
R#10	$Q(X, Knows(A, X) \wedge Trusts(A, X), Knows(X, C) \wedge Trusts(X, C)) \wedge Knows(A, C) \rightarrow Trusts(A, C)$
R#11	$Q(X, Knows(A, X) \wedge \neg Trusts(A, X), Knows(X, C) \wedge Trusts(X, C)) \wedge Knows(A, C) \rightarrow \neg Trusts(A, C)$
R#12	$Q(X, Knows(A, X) \wedge Trusts(A, X), Knows(X, C) \wedge \neg Trusts(X, C)) \wedge Knows(A, C) \rightarrow \neg Trusts(A, C)$
R#13	$Q(X, Knows(A, X) \wedge \neg Trusts(A, X), Knows(X, C) \wedge \neg Trusts(X, C)) \wedge Knows(A, C) \rightarrow Trusts(A, C)$
	Cyclic rules using soft quantifier
R#14	$Q(X, Knows(A, X) \wedge Trusts(A, X), Knows(X, C) \wedge Trusts(X, C)) \wedge Knows(C, A) \rightarrow Trusts(C, A)$

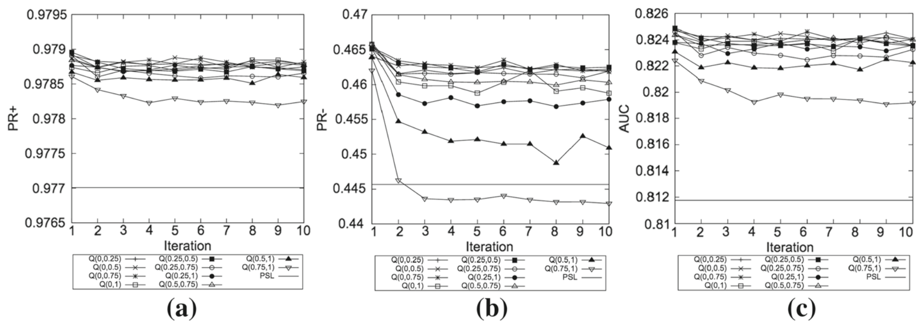

Fig. 3 **a** PR+ , **b** PR−, and **c** AUC when changing α and β in the quantifier mapping $\tilde{Q}$

Table 3 Values with a $*$ are statistically significant with a rejection threshold of 0.05 and values in bold are statistically significant with a rejection threshold of 0.1 using a paired t test w.r.t. the PSL program (Bach et al. 2013; Huang et al. 2013)

Method	PR+	PR−	AUC
PSL (Bach et al. 2013; Huang et al. 2013)	0.977	0.446	0.812
PSL^Q ($\tilde{Q}_{[0,0.25]}$), ($k = 1$)	**0.979***	**0.467***	**0.825***
PSL^Q ($\tilde{Q}_{[0,0.25]}$), ($k = 10$)	**0.979***	**0.463**	**0.824***

Distrust prediction is more challenging than trust prediction (i.e., PR− values are overall lower than PR+ values) because of the unbalanced nature of the data (7974 trust vs. 701 distrust relations)

PSL^Q program with $\tilde{Q}_{[0.75,1]}$ in $PR-$ after the first two iterations. An explanation for this is the fact that people trust/distrust a third party as soon as a few/some of their trusted/distrusted friends trust/distrust that person and not most of them, i.e., more than 75%. Interestingly, by decreasing both α and β values, results get better. The program with the best predicting scores is PSL^Q with $\tilde{Q}_{[0,0.25]}$ as a quantifier mapping representing "a few" (see Table 3).

Figure 4 presents the convergence curve of the 8 PSL^Q programs with quantifier mapping $\tilde{Q}_{[0,0.25]}$ based on eightfold for 50 iterations during training. At each iteration k, all the grounded rules in the PSL^Q program are generated and then the weighted sum of distance to

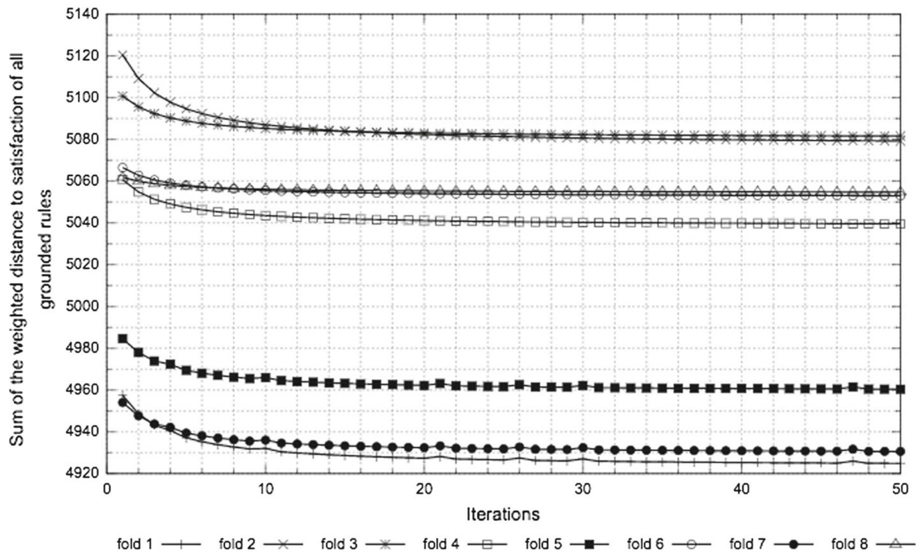

Fig. 4 Visualizing the weighted sum of the distance to satisfaction of grounded rules for eightfold in 50 iterations of MPE inference for social trust PSL^Q program

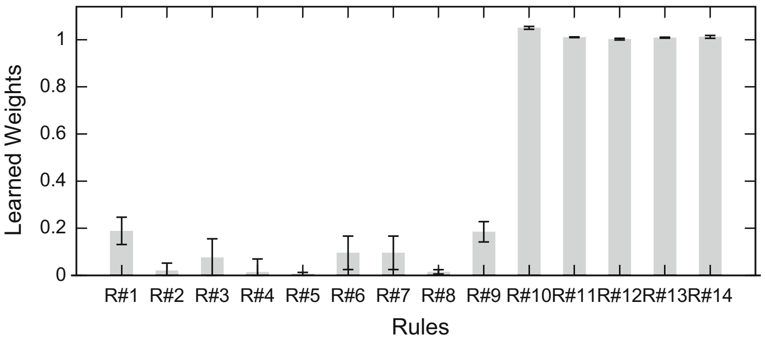

Fig. 5 Learned values of the weight of the 14 rules of the PSL^Q program

satisfaction of all grounded rules is calculated. As shown in Fig. 4, MPE inference converges for all eightfold. After $k = 10$ the variation rate in the outcome of calculating the weighted sum of distance to satisfaction value is very low. Note that the range of the weighted sum of the distance to satisfaction of each fold is different from each other because the program is completely different.

Figure 5 emphasizes the importance of the PSL^Q rules with quantifier expressions (rules 10–14) after the weight learning phase. Bars represent average and error bars represent minimum and maximum weights of the rules learned in eightfold for the PSL^Q program with quantifier mapping $\tilde{Q}_{[0,0.25]}$. These results show that using soft quantifiers not only improves the accuracy of trust and distrust predictions but also that the rules containing soft quantifiers, i.e. rules 10–14, play a major part in this by dominating all other rules in terms of weight. In these experiments, we used one quantifier mapping for all the quantifiers in a PSL^Q program;

Table 4 Details about the Netlog data sample

Data	#Users	#Female		#Male		#Young		#Non-young	
Core	3015	1,097	36%	1918	64%	1,501	50%	1514	50%
Public core	2241	697	31%	1544	69%	946	42%	1295	58%
Private core	765	399	52%	366	48%	553	72%	212	28%
Background	171,439	81,290	47%	90,149	53%	91,588	53%	79,860	47%
Sample	174,454	82,387	47%	92,067	53%	93,090	53%	81,364	47%

The age groups *Young* and *Non-young* were created around a split threshold of 25

however it is possible to use different mapping functions for each quantifier expression in a PSL^Q program, which is an interesting direction for future research.

Training and predicting with PSL takes little time. In our experiments, training each model takes about a few minutes, while predicting takes under a second per trust/distrust relation. For the case of the PSL^Q program, training takes almost k times (i.e, number of iterations) the time of training our baseline PSL program.

6 Node labeling: user profiling in social networks

Many applications benefit from reliable approaches of user profiling in social networks. Such applications exist in various fields, from personalized advertising to reputation management (Dijkmans et al. 2015; Ha et al. 2015). In this section, we present and evaluate a PSL^Q program for inferring age and gender of social network users.

We collected user profiles from Netlog, an early social network that had over 90 million users worldwide at its peak. By applying snowball sampling, and starting from one user, we crawled the profiles of 3015 users, called the *core users* henceforth. Out of these 3015 profiles, 765 users (25%) have private profiles (the *private core users*) and 2241 (75%) have public profiles (the *public core users*). Next we crawled the user profiles of all the friends of the public core users, resulting in 171,439 additional profiles, referred to as the *background users*. Note that we could not do the same for the private core users, as their friend lists are not publicly accessible. We ended up with a sample network including 174,454 users and 277,191 friendship links. For all users in the sample, we extracted their age and gender, which is publicly available for all users in Netlog. Table 4 describes detailed information regarding the sample.[3]

The gender classes are fairly imbalanced among the core users (64% male vs. 36% female) and even more so among the public core users (69% male vs. 31% female). The average age of users in the overall sample is 29 years old, and the median is 25 years old. We choose the threshold of being 25 years old to divide users into two age groups of *Young* and *Non-young* users. The resulting age groups in the sample are uniform with 53% Young users and 47% Non-young users. Interestingly, the majority of private core users (i.e., 72%) are Young users, which indicates that Young users are more concerned about their privacy.

Table 5 includes a description of the social graph. Since we used snowball sampling, the graph consists of a single connected component. The average degree of the public core users is substantially higher than the average degree of users in the sample overall because for each

[3] The Netlog dataset is available at: http://www.cwi.ugent.be/NetlogDataSet.html.

Table 5 Description of the network properties of the Netlog sample graph

Attribute	#Nodes	Avg degree	Min degree	Max degree
All users	174,454	3	1	1183
Public core users	2241	125	1	480

Table 6 Analysis of users' age and gender given their friends age and gender clearly indicate the *age homophily* and *gender heterophily* in the graph

N_{p_1,p_2}		p_2 Female	Male
p_1	Female	25%	75%
	Male	61%	39%
N_{p_1,p_2}		p_2	
		Young	Non-young
p_1	Young	89%	11%
	Non-young	37%	63%

public core user we purposely crawled and included all their friends, which we did not do for the background users.

Given an interpretation I, and P_1 and P_2 predicate symbols such as $Female$, $Male$, $Young$ or *Non-young*, we explore the relation between the public core users age and gender and the age and gender of their friends by calculating N_{P_1,P_2} as:

$$N_{P_1,P_2} = \frac{1}{|C_{P_1}|}\frac{|\{v|P_2(v) \wedge (u,v) \in E\}|}{|\{v|(u,v) \in E\}|}$$

where $\Pi = \{u|u \text{ is a public core user}\}$, $C_P = \{u|u \in \Pi \wedge P(u)\}$, and E denoted the friendship relations in the Netlog sample graph. The results are presented in Table 6 which gives a first insight on how gender of users are structured within the Netlog network. Users' behaviour in Netlog is of the type *gender heterophily*, which indicates the tendency of users to connect with the opposite gender, deriving from the particular usage of this social network site for flirting and dating purposes. The age results in Table 6 on the other hand clearly indicate that users have a strong tendency to connect within their same age group, i.e., *age homophily* behaviour. These results are in line with the properties of the Netlog network in Farnadi et al. (2015), and are apparent from Fig. 6 as well.

In the gender graph (Fig. 6), blue (i.e., male) and red (i.e., female) users are mostly connected to the opposite color, while in the age graph (Fig. 6) there is a clear tendency of connecting with the same color.

Below we present PSL and PSL^Q programs to be used in scenarios where we only have the age and gender of some users and our aim is to infer the age and gender of the remaining users given the social relations between them. We perform tenfold cross validation by randomly dividing the *public core users* in 10 parts. In each fold, one part is used for testing and the other 9 parts are added to the *background* and *private core* users and used as the training set for that fold.

We built a PSL program as shown in Table 7 based on our observation that the Netlog network exhibits gender heterophily and age homophily property. For example, the first PSL rule indicates that if two users are friends and one of them is female, it is more probable that the other user is male. Since being female and being male is mutual exclusive, where $Female(A) =$

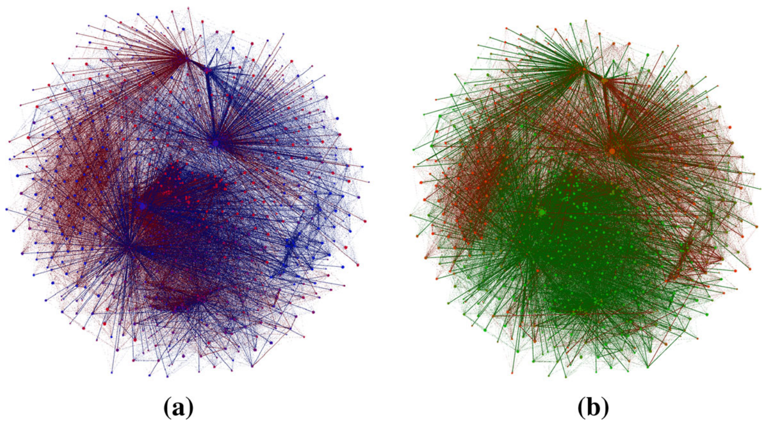

Fig. 6 Visualizing the Netlog sample graph with Gephi (Bastian et al. 2009) using OpenOrd layout. *Red nodes* represent female users and *blue nodes* represent male users in (**a**) and similarly, *green nodes* represent Young users and *orange nodes* represent Non-young users in (**b**). **a** Gender graph. **b** Age graph

Table 7 PSL rules for user profiling

	Gender heterophily rules
$R\#1$	$Friend(A, B) \wedge Female(A) \rightarrow \neg Female(B)$
$R\#2$	$Friend(A, B) \wedge \neg Female(A) \rightarrow Female(B)$
	Age homophily rules
$R\#3$	$Friend(A, B) \wedge Young(A) \rightarrow Young(B)$
$R\#4$	$Friend(A, B) \wedge \neg Young(A) \rightarrow \neg Young(B)$

Table 8 PSL^Q rules for user profiling

	Gender heterophily rules using soft quantifier
$R\#5$	$Q(X, Friend(A, X), Female(X)) \rightarrow \neg Female(A)$
$R\#6$	$Q(X, Friend(A, X), \neg Female(X)) \rightarrow Female(A)$
	Age homophily rules using soft quantifier
$R\#7$	$Q(X, Friend(A, X), Young(X)) \rightarrow Young(A)$
$R\#8$	$Q(X, Friend(A, X), \neg Young(X)) \rightarrow \neg Young(A)$

$\neg Male(A)$, we only need to consider one predicate for gender. For example, $I(Female(X)) = 1$ indicates that user X is Female and $I(Female(X)) = 0$ indicates that user X is male. Similarly, we only consider one predicate for the age group. For example, $I(Young(X)) = 1$ indicates that user X is young. Since friendship relations in Netlog are undirected and symmetric, $I(Friend(A, B)) = I(Friend(B, A))$. To include both directions in the program, we populate both friendship directions in our graph, which produces 554,374 friendship relations in our dataset.

To examine the effects of using soft quantifiers, we built a PSL^Q program as shown in Table 8. The first two rules express the intuition behind gender heterophily, meaning that if most of A's friends are female, then A is male (rule $R\#5$), and that if most of A's friends are male, then A is female (rule $R\#6$). Along those lines, rule $R\#7$ and $R\#8$ express age homophily.

Table 9 Result of age and gender prediction in Netlog

Method	Gender prediction				Age prediction			
	PR+	PR-	AUC	Accuracy	PR+	PR−	AUC	Accuracy
Majority baseline	0	1	0.53	0.53	0	1	0.53	0.53
PSL	0.31	0.70	0.50	0.69	0.42	0.59	0.51	0.57
PSL^Q($Q_{[0.0.25]}$)	**0.63**	**0.88**	**0.80**	**0.80**	**0.64**	**0.79**	**0.76**	**0.79**
PSL^Q($Q_{[0.0.5]}$)	**0.66**	**0.91**	**0.84**	**0.72**	**0.67**	**0.83**	**0.81**	**0.74**

Values in bold are statistically significant with a rejection threshold of 0.01 using a paired *t* test w.r.t. the PSL program and majority baseline

We examine different quantifier mappings to find the best α and β values. We choose a step of 0.25 to produce 10 different quantifier mappings and evaluate them based on PR−, PR+, AUC and overall accuracy (i.e., the number of correct predictions made divided by the total number of predictions made). As shown in Table 9, the best AUC scores are obtained with the PSL^Q program with $Q_{[0,0.5]}$ as a quantifier mapping representing "most". The PSL^Q program with $Q_{[0,0.25]}$ as a quantifier mapping, which indicates "a few", gets the best overall accuracy for both the age and the gender prediction task. Both PSL^Q programs significantly outperform their sibling PSL program and the majority baseline. The PSL^Q programs owe their high performance to their local averaging behaviour for each user while the behaviour of the PSL program is less local, and, for the PSL rules presented in this paper, approximates global averaging behaviour.

7 Conclusion

In this paper, we have introduced PSL^Q, the first SRL framework that supports reasoning with soft quantifiers, such as "most" and "a few". PSL^Q is a powerful and expressive language to model uncertain relational data in an intuitive way. Since this expressivity pushed beyond the capabilities of existing PSL-MPE solvers, we have introduced and implemented new inference and weight learning algorithms that can handle rules with soft quantifiers. We have shown how the higher expressivity of PSL^Q can lead to better results in practice by extending an existing PSL program for link prediction in social trust networks with rules that contain soft quantifiers. Similarly, we have presented results for both PSL and PSL^Q programs for user profiling in a social network. We have presented the effects of using different interpretations of soft quantifiers in both applications. As a next step, we want to include an automatic way of learning the best interpretation for each quantifier expression in a PSL^Q program. Besides trust link prediction and user profiling, many other applications could benefit from the use of soft quantifiers. Exploring the effects of using soft quantifiers in PSL^Q programs for various applications in different domains is therefore another promising research direction. Furthermore, in addition to the approach of Zadeh that we have used in this paper, other approaches for soft quantifiers have been proposed (Delgado et al. 2014), most notably Yager's OWA-operators (Yager 1988); we plan to investigate them in our future work.

Acknowledgements This work was funded in part by the SBO-program of the Flemish Agency for Innovation by Science and Technology (IWT-SBO-Nr. 110067) and NSF Grant IIS1218488. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the NSF.

References

Alshukaili, D., Fernandes, A. A. A., & Paton, N. W. (2016). Structuring linked data search results using probabilistic soft logic. In *International semantic web conference (ISWC)*.

Bach, S. H., Broecheler, M., Huang, B., & Getoor, L. (2015). Hinge-loss Markov random fields and probabilistic soft logic. arXiv:1505.04406 [cs.LG].

Bach, S. H., Huang, B., London, B., & Getoor, L. (2013). Hinge-loss Markov random fields: Convex inference for structured prediction. In *Proceedings of the Uncertainty in Artificial Intelligence (UAI)*.

Bastian, M., Sebastien, H., & Mathieu, J. (2009). Gephi: An open source software for exploring and manipulating networks. In *Proceedings of the international AAAI conference on web and social media (ICWSM)* (pp. 361–362).

Beltagy, I., & Erk, K. (2015). On the proper treatment of quantifiers in probabilistic logic semantics. In *Proceedings of the 11th international conference on computational semantics (IWCS)* (p. 140).

Beltagy, I., Erk, K., & Mooney, R. J. (2014). Probabilistic soft logic for semantic textual similarity. In *Proceedings of the 52nd annual meeting of the association for computational linguistics (ACL)* (pp. 1210–1219).

Bobillo, F., & Straccia, U. (2008). fuzzyDL: An expressive fuzzy description logic reasoner. In *Proceedings of the IEEE international conference on fuzzy systems (FUZZ-IEEE)* (pp. 923–930).

Cao, T. H., Rossiter, J. M., Martin, T. P., & Baldwin, J. F. (2002). On the implementation of Fril++ for object-oriented logic programming with uncertainty and fuzziness. In *Technologies for constructing intelligent systems 2* (pp. 393–406). Physica-Verlag HD.

Charnes, A., & Cooper, W. W. (1962). Programming with linear fractional functionals. *Naval Research Logistics Quarterly*, *9*, 181–186.

Collins, M. (2002). Discriminative training methods for hidden Markov models: Theory and experiments with perceptron algorithms. In *Proceedings of the international conference on empirical methods in natural language processing (ACL)* (pp. 1–8).

Delgado, M., Ruiz, M.-D., Sánchez, D., & Vila, M.-A. (2014). Fuzzy quantification: A state of the art. *Fuzzy Sets and Systems*, *242*, 1–30.

Delgado, M., Sánchez, D., & Vila, M. A. (2000). Fuzzy cardinality based evaluation of quantified sentences. *International Journal of Approximate Reasoning*, *23*, 23–66.

Deng, L., & Wiebe, J. (2015). Joint prediction for entity/event-level sentiment analysis using probabilistic soft logic models. In *Conference on empirical methods in natural language processing (EMNLP)*.

Dijkmans, C., Kerkhof, P., & Beukeboom, C. J. (2015). A stage to engage: Social media use and corporate reputation. *Tourism Management*, *47*, 58–67.

Ebrahimi, J., Dou, D., & Lowd, D. (2016). Weakly supervised tweet stance classification by relational bootstrapping. In *Conference on empirical methods in natural language processing (EMNLP)*.

Fakhraei, S., Huang, B., Raschid, L., & Getoor, L. (2014). Network-based drug-target interaction prediction with probabilistic soft logic. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, *11*, 775–787.

Farnadi, G., Bach, S., Blondeel, M., Moens, M.-F., Getoor, L., & De Cock, M. (2015). Statistical relational learning with soft quantifiers. In *Proceedings of 25th international conference on inductive logic programming (ILP)*.

Farnadi, G., Bach, S. H., Moens, M. F., Getoor, L., & De Cock, M. (2014). Extending PSL with fuzzy quantifiers. In *Proceedings of the Fourth International Workshop on Statistical Relational AI at AAAI (StarAI)*.

Farnadi, G., Mahdavifar, Z., Keller, I., Nelson, J., Teredesai, A., Moens, M.-F., & De Cock, M. (2015). scalable adaptive label propagation in Grappa. In *Proceedings of IEEE international conference on big data (IEEE-BigData)*.

Getoor, L., & Taskar, B. (2007). *Introduction to statistical relational learning*. Cambridge: MIT press.

Ha, I., Oh, K.-J., & Jo, G.-S. (2015). Personalized advertisement system using social relationship based user modeling. *Multimedia Tools and Applications*, *74*, 8801–8819.

Heider, F. (1958). *The psychology of interpersonal relations*. New York: Wiley.

Huang, B., Kimmig, A., Getoor, L., & Golbeck, J. (2013). A flexible framework for probabilistic models of social trust. In *Social computing, behavioral-cultural modeling and prediction* (pp. 265–273).

Isbell, J. R., & Marlow, W. H. (1956). Attrition games. *Naval Research Logistics Quarterly*, *3*, 71–94.

Jain, D., Barthels, A., & Beetz, M. (2010). Adaptive Markov logic networks: Learning statistical relational models with dynamic parameters. In *Proceedings of the European conference on artificial intelligence (ECAI)* (pp. 937–942).

Kazemi, S. M., Buchman, D., Kersting, K., Natarajan, S., & Poole, D. (2014). Relational logistic regression. In *Proceedings of the international conference on principles of knowledge representation and reasoning (KR)*.

Klir, G., & Yuan, B. (1995). *Fuzzy sets and fuzzy logic*. New Jersey: Prentice Hall.

Kouki, P., Fakhraei, S., Foulds, J., Eirinaki, M., & Getoor, L. (2015). HyPER: A flexible and extensible probabilistic framework for hybrid recommender systems. In *ACM conference on recommender systems (RecSys)*.

Leskovec, J., Huttenlocher, D., & Kleinberg, J. (2010). Signed Networks in Social Media. In *Proceedings of the 28th ACM conference on human factors in computing systems (CHI)*.

Liu, S., Liu, K., He, S., & Zhao, J. (2016). A probabilistic soft logic based approach to exploiting latent and global information in event classification. In *AAAI conference on artificial intelligence (AAAI)*.

Lowd, D., & Domingos, P. (2007). Recursive random fields. In *Proceedings of the international joint conference on artificial intelligence (IJCAI)* (pp. 950–955).

Milch, B., Zettlemoyer, L. S., Kersting, K., Haimes, M., & Kaelbling, L. P. (2008). Lifted probabilistic inference with counting formulas. In *Proceedings of the international conference on artificial intelligence (AAAi)* (Vol. 8, pp. 1062–1068).

Muggleton, S., & De Raedt, L. (1994). Inductive logic programming: Theory and methods. *The Journal of Logic Programming*, *19*, 629–679.

Poole, D., Buchman, D., Kazemi, S. M., Kersting, K., & Natarajan, S. (2014). Population size extrapolation in relational probabilistic modelling. In *Proceedings of the international conference on scalable uncertainty management (SUM)* (pp. 292–305). Springer.

Poole, D., Buchman, D., Natarajan, S., & Kersting, K. (2012). Aggregation and population growth: The relational logistic regression and Markov logic cases. In *Proceedings of the international workshop on statistical relational AI at UAI (StarAI)*.

Prade, H., Richard, G., & Serrurier, M. (2003). Learning first order fuzzy logic rules. In *Proceedings of the 10th international fuzzy systems world congress (IFSA)* (pp. 702–709). Springer.

Pujara, J., Miao, H., Getoor, L., & Cohen, W. (2013). Knowledge graph identification. In *Proceedings of the international semantic web conference (ISWC)*.

Richardson, M., & Domingos, P. (2006). Markov logic networks. *Machine Learning*, *62*, 107–136.

Sridhar, D., Fakhraei, S., & Getoor, L. (2016). A probabilistic approach for collective similarity-based drug-drug interaction prediction. *Bioinformatics*, *32*(20), 3175–3182.

Van den Broeck, G., Meert, W., & Darwiche, A. (2013). Skolemization for weighted first-order model counting. arXiv preprint arXiv:1312.5378.

Victor, P., Cornelis, C., & De Cock, M. (2011). Trust and recommendations. In *Recommender systems handbook* (pp. 645–675). Springer.

West, R., Paskov, H. S., Leskovec, J., & Potts, C. (2014). Exploiting social network structure for person-to-person sentiment analysis. *Transactions of the Association for Computational Linguistics (TACL)*, *2*, 297–310.

Yager, R. R. (1988). On ordered weighted averaging aggregation operators in multicriteria decision making. In *IEEE transactions on systems, man and cybernetics (IEEE SMC)* (pp. 183–190).

Zadeh, L. A. (1983). A computational approach to fuzzy quantifiers in natural languages. *Computers and Mathematics with Applications*, *9*, 149–184.