Social norm recommendation for virtual agent societies

Paper ID: 46

No Institute Given

Abstract. Norms express expectations about behaviours of interacting entities in both human and software agent societies. While humans naturally possess the ability to recognize existing norms and learn new ones, software agents representing human users (e.g. avatars in virtual worlds) have to be endowed with such capabilities. Such a norm recognizing and learning agent, using observed actions and interactions of other agents in a situated environment, can be considered a *norm-aware* agent. This paper contributes to the agenda of creating a norm-aware agent, by proposing an architecture for social norm recommendation, comprising norm identification, norm classification, norm life-stage detection, and finally norm recommendation. These recommendations can then either be provided to a human user (e.g. the user of an embodied virtual agent in a new environment) or can be used by the agent itself for choosing appropriate actions. We use a simulation-based study to demonstrate how the four phases of the norm recommendation system work. The contributions of this paper are: (i) a comprehensive account of norm recommendation: identification, classification, life-stage detection and how to combine them into a recommendation, and (ii) a first evaluation of the approach, which advances the state of the art for this problem.

1 Introduction

Norms are of interest to researchers in multi-agent systems because they enable cooperation and coordination in software agent societies [1] and their smoother functioning by facilitating social order. Since norms can be used to help establish social order, MAS researchers have used this concept to build multi-agent systems.

Agents that know about norms in their societies do not need to recompute what the norms of the society are [4] and also do not often need to spend time in contemplating which actions are permitted, forbidden and obliged as they are aware of these norms. Also, agents that are aware of norms know that violating them has consequences. However, a new agent joining a society may not know the norms and needs to be equipped with some mechanism for recognizing them.

An agent may come to know about the norms of a society through several mechanisms: (i) an agent may ask another agent, (ii) it can also observe the actions and interactions of others and infer norms. In societies where norms are well-established and well-communicated, the agents know what the norms are. However, an agent may not know about norms where those of different communities are different or the norms may be in a state of flux (i.e. norms might be emerging or changing). For example, in domains such as virtual worlds and multi-player online-games, apart from strict rules on what agents can or cannot do, that are usually encoded in the system at design-time,

there might be norms that are generated at run-time, which do not exist *a priori*. For example, a specific reciprocity norm might be generated at run-time – derived from the generic "do unto others..." – in an online game, when one player who helps another escape from a dragon, expects reciprocal help [15] when another such a situation arises with roles reversed. Also, depending upon the population composition, norm may also change: for example, the influx of new agents might lead to norm change in the society. Additionally, when norms are in flux communication-based approaches for norm spreading may not be reliable since agents may miscommunicate what the norms are, or agents could lie about what the norms might be. Hence, there is a need for agents to recognize what the new norms are and also keep abreast with changing norms. One possible approach to recognize these norms is for agents to use an observation-based approach for inferring potential norms.

The problem of knowing the norms of a society is difficult because different societies have different norms and the uptake of different types of norms can vary. The work of Savarimuthu et al [9,10] and of Oren and Meneguzzi [7] has focussed on the problem of norm identification. We believe that a more comprehensive norm recommendation architecture is necessary that not only identifies norms, but also recommends norms taking into account norm salience – how often does a violation get sanctioned and the life-stage of the norm, namely whether it is emergent, stable or decaying.

Thus, the objective of this paper is the proposition and evaluation of a high-level architecture for norm recommendation that takes these factors into account. An effective mechanism would mean a new agent (human or software) joining an agent society (e.g. a virtual world) could receive normative advice about expected behaviour. In a virtual world scenario, where human users interact with one another using their avatars, we foresee such users would then be able to receive advice from the avatar thanks to the norm recommendation module. The human users can subsequently decide whether to take the advice or not. We demonstrate the functioning of the architecture using synthetic data and focussing on prohibition norms and show how its effectiveness improves upon published work. As the literature suggests, the approach should be equally applicable for obligation norms.

Section 2 reviews existing work on norm identification and situates the current work against that, while an overview of the norm recommendation system follows in Section 3. Section 3.1 presents the high-level details of the experimental design and also introduces the important parameters of the norm recommendation system which are then used in the subsequent sections. Section 3.2 presents the norm identification phase, highlighting in particular how this advances upon existing work both in principle and in practice. Section 3.3 presents the norm classification phase and Section 3.4 presents the norm life-stage detection phase. The three phases are drawn together to make an heuristic-based norm recommendation system in Section 3.5. We conclude with some discussion and ideas for future work in Section 4.

2 Background and contribution

Sen [12], Savarimuthu [11] and Villatoro [14] all report on investigations into how norms might emerge in societies. Agents in these works employ one of several mecha-

nisms such as imitation, advice-based learning (e.g. learning norms from leaders, peers etc.), and machine learning (e.g. Q-Learning) to learn a new norm. As norms are learnt by individual agents, this leads to norm emergence at the societal level. For an overview of norm learning mechanisms that facilitate norm emergence we refer the reader to [8]. Morales et al. [6] have investigated how new norms can be synthesized in the context of traffic domain. One observation on the above is that they consider only a limited number of actions (e.g. the work of Sen et al. [12] considers only two actions: whether an agent should give way to the left or to the right). Also, in most cases, only the emergence of a single norm is considered. However, in a more realistic setting, such as virtual worlds and massively multi-player games, where human users interact with other human or software agents, there are a large number of actions available for agents and also information (partial or complete) about other agents' actions might be available. For example, an agent in a virtual environment can observe actions that other agents perform and their interactions. Agents can employ data mining techniques to infer what the norms of the societies are based on the data on agent behaviour (obtained through observations). Corapi et al. [2] have explored how norms can be synthesized using an induction-based learning approach. This however is downstream from the problem addressed here: it describes a concrete method for revising a computational logic based norm representation that requires both the observations of actions related to the norm and the decision to revise the norm set have been made, which is exactly the purpose of the method outlined here.

The prohibition norm identification algorithm put forward in [10], in contrast to [12] above, considers up to four actions, associating those that occur $x\%$ of the time before the sanction event as having a causal relationship to the sanction. The exact value of x is controlled by a parameter called the Norm Inference Threshold (NIT), which is set by the (observing) agent. To illustrate the approach consider an agent that litters a park and which might receive a sanction from another agent. An observer, based on the actions and sanction it sees, can infer that littering is prohibited. More recently, Oren and Meneguzzi [7] have proposed algorithms for norm identification by means of Hierarchical Task Network (HTN) planning, however only conventions can be recognized using this approach since sanctions are not taken into account. The work of Criado et al. [3] proposes norm reasoning services for agents where agents can make use of a norm reasoning service to seek advice on norms. However, this work assumes that norms are known *a priori* as opposed to the situation in norm identification, where norms are not known ahead of time.

The approach to norm identification in [10] has several limitations that impact on the quality of the recommendations it makes:

L1: Focuses on norm identification that does not take into account potentially significant contextual information, such as: (i) determining the life-stage of a norm – that is whether it is emerging or decaying, for example – and factoring this into the recommendation process, and (ii) whether the norm is salient (i.e. how often does the violation of a norm actually result in a sanction).

L2: Infers norms from the punished event sequence (PES), without considering those occasions in which an action occurs, but does not lead to punishment (i.e. unpunished event sequence (UES)). Examples of PESs and UESs are provided in

Figure 2 and are discussed in detail in Section 3.2. For example, if action A happens five out of ten times in the PES, while in the other five sequences another action (e.g. B) is the reason for the sanction to occur and action A does not appear in those sequences, then the punishment probability[1] is 0.5 (i.e. $PP(A) = 0.5$). However, there could have been n other instances (e.g. 15) in the UES where the action A happens, but is not punished. The algorithm in [10] calculates PP only based on PES, whereas here we take both PES and UES into account. Hence, $PP(A) = 0.25$ because action A is punished 5 out of 20 occurrences.

L3: The limited number of actions that agents can perform (as noted above) as set by the experiment designer. Whereas here the algorithm determines the number of actions to take into account based on the available data.

L4: Consideration of one type of probability distribution over actions – called pre-determined – where there is a fixed probability assigned to each action in advance, to generate event sequences. The impact of other kinds of distribution (e.g. uniform, Gaussian) is not examined.

L5: High number of false positives when trying to identifying multiple co-existing norms. This is directly related to the dependence (of the algorithm in [10]) on the norm inference threshold (NIT) parameter, which has to be lowered to the punishment probability of the lowest norm in order to detect (all) the norms. For example, if there are three norms such as prohibit(*walk on the grass*), prohibit(*litter*) and prohibit(*eat*) in the context of behaviour in a park, and if the punishment probability for violating each of these is 0.5 and 0.2 and 0.1 respectively, then NIT must be less than or equal to 0.1 to identify all three. However, lowering NIT is counter-productive, because it generates false positives. Even though the false positive problem can be resolved at the norm verification stage, because the agent (in [10]) checks with another agent that the norm it has inferred is indeed correct, it remains problematic since the other agent may not respond truthfully. The cited work assumes the agents are truthful.

The current work addresses the five limitations above. We present a comprehensive norm recommendation system consisting of four phases: (i) identification, (ii) classification (iii) life-stage detection (iv) recommendation. This addresses **L1** above by taking into account more elements of the bigger picture. An agent uses the formula above for punishment probability (**L2**). As noted above, the number of actions to consider is determined automatically (**L3**). The new algorithm is demonstrated using 20 randomly generated actions and the impacts of three different distributions (uniform, Gaussian and pre-determined) that are used to generate actions that agents perform in the simulations (**L4**). We also demonstrate that the algorithm reduces the number of false positives and false negatives (**L5** – details in Section 3.2). The new algorithm does not use NIT and also does not require the norm verification stage employed in [10], thus removing

[1] **Punishment Probability (PP)**: The punishment probability of an action A is the ratio of the number of punished occurrences of A to the total number of occurrences of A as expressed in the formula:

$$PP(A) = \frac{\text{Number of occurrences of A where it was punished}}{\text{Total number of occurrences of A}}$$

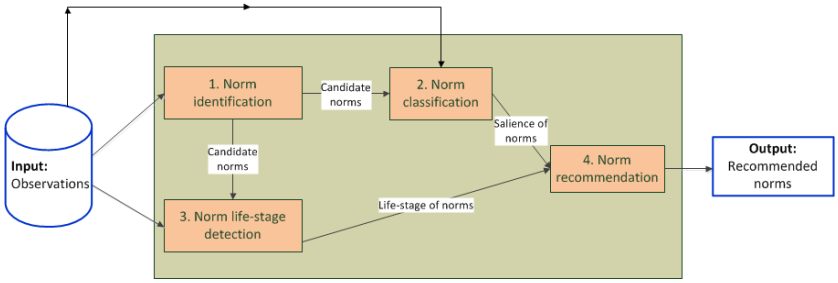


Fig. 1: Architecture of the norm recommendation system

(i) a dependence on an arbitrary parameter and (ii) reliance on the veracity of other agents, so making the approach more robust and better decoupled.

3 Norm recommendation architecture

Existing approaches to norm identification, such as [9, 10], focus on how norms are identified and do not consider the importance of contextual aspects in providing norm recommendation, such as detecting norm salience and the life stage of the norm. This section provides an overview of a norm recommendation architecture that brings these factors into the process. The system comprises four phases, as shown in Figure 1. The input is the set of observations of an individual agent and the output is a set of norms and recommendations on each regarding whether an agent should follow or not. The norm recommendation system forms a part of an agent's reasoning process, in that one of the inputs to the agent's decision making process is the output of the recommendation system, along with beliefs, desires and goals.

1. **Norm identification:** The first phase identifies what we call candidate norms. The criterion is the ratio of violations for punishments to total occurrences for a given action (i.e. $PP(A)$). This is used in the algorithm discussed in Section 3.2.
2. **Norm classification:** The salience of the candidate norms (the extent to which norm violations are sanctioned) are then analyzed in the norm classification stage. Norm classification is based on punishment probability of the candidate norms obtained from the previous stage and the frequency of actions that are governed by the norms. This stage uses the frequency of an action (low, high) and probability of punishment for that action (low, high) to establish the salience category of a candidate norm. Hence, the four norm salience categories are very high, high, low and very low as depicted in Figs 5(a)–5(d), and discussed in more detail in Section 3.3. It must be noted that norm classification takes a short term view of the norm (i.e. the agent looks at the salience of the norm at the current time step). The longer term is addressed by the next phase.
3. **Norm life-stage deduction:** Based on any historical data that is available, it is possible to assign a life-stage to a candidate norm. We use a 5-stage categorisation: emerging, growing, maturing, declining or decaying (i.e. severely declining). For example, the norm against smoking virtually did not exist before the 1970s. In the

1990s, it might been said to be emergent, followed by growing rapidly in the 2000s. From such situations, we conclude it is useful for an agent to be able to take the life-stage of a norm into account in determining its reaction to it. The life-stage of a norm provides a longer-term perspective, in that it situates the current adoption of a norm in a historical context, by using a history of observed interactions between agents over as long a period of time as is recorded.

4. **Norm recommendation:** Heuristics are now used, driven by the short and long term information of the previous two phases, to make a recommendation on whether to follow a norm or not. For example, if salience is low and the life-stage is decaying, then the agent can be recommended not to follow the norm. However, if salience is very high and life-stage is emerging, then an agent might be recommended to follow the norm. To ground this: if the salience of the norm against smoking is high (i.e. smokers are punished with high probability), then the agent is recommended not to smoke.

3.1 Experiment Design

In this section we summarise the framework in which the experiments are conducted. We also describe the key parameters that are used in the sections discussing the phases and the results.

The approach taken for the simulation experiments is abstracted from any actual scenario. Thus, the agents have a vocabulary of actions available to them, denoted A–Z, while the system has a vocabulary of sanctions, denoted #, @, $. Agent action selection is based on a probability distribution; in this paper we work with three: (i) uniform, so each action is equally likely, (ii) Gaussian (or normal), so actions closer to the middle of the (ordered) action list are more likely than those further away and (iii) pre-determined, where actions are assigned probabilities in advance (useful for bespoke distributions and testing). The system is configured (for a given run) with prohibition norms, by specifying as many action sequences leading to a sanction event as the experimenter wishes. The task for the agents is to discover this configuration.

In an experiment, an agent acts as an observer, collecting data on the actions of another agent and whether there are any consequent sanctions. An agent has a configurable history length (HL) that determines the length of the observation sequence and within that uses a window size (WS) to determine how many actions to consider prior to a sanction event when trying to establish a correlation between action(s) and sanction. This window size ranges from 1 up to HL as the agent seeks to discover the violating action sequence. An illustration of an experiment appears in Figure 2, with HL=20, WS=4 and a uniform distribution for actions. The one remaining experiment parameter is the generation window size (GWS), which determines the maximum delay between a violation and a sanction which is set by the experimenter. If GWS is set to 10, then the violations will be punished within the next 10 actions. Other experiment parameters (numbers of runs etc. are given in the context of the experimental results).

The next four sections (Sections 3.2 to 3.5) describe each of the four phases that lead to norm recommendation. Each of these sections present both the details of the corresponding phases and the experimental results obtained.

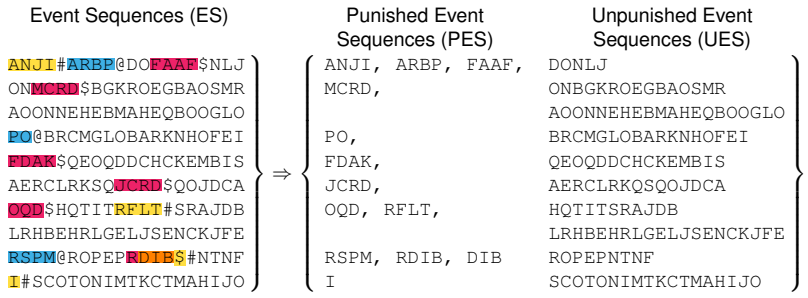


Fig. 2: An example of punished and unpunished event sequences

3.2 Norm identification

The objective of the norm identification phase is to identify the candidate norms. The input is the history of actions that an agent observes over a period of time. When a new agent joins a society, it records the observed actions and interactions in a log. We assume that agents can recognize sanctions, but do not know the reason for the sanctions (i.e. do not know what norms were violated to have caused these sanctions).

The starting point for the algorithm is sanction recognition. Once sanctions are recognized, the reasons for these sanctions are investigated (i.e. norm violations are reasons for sanctions to occur). As noted earlier, the prohibition norm identification algorithm of [10] identifies those actions that occur $x\%$ of the time before the sanctions, but that does not take into account issues two to five as discussed in Section 2 and for which we now propose a revision (see Algorithm 1).

We start by computing the punishment probability through examination of episodes where sanctions follow and do not follow an action. An agent uses a fixed length history of event sequences, over which it runs a variable window size (WS). The agent looks into its history for a certain number of recent events that precede a sanction. For example, if the WS is set to 3, an agent creates an event episode with the last three events that were observed before the sanction. The sequence of events in this window is called the punishment event sequence (PES). The agent does not know the right window size, so it lets WS range over $[1, \mathrm{HL})$ and calculates the punishment probabilities for all the actions. The actions are then sorted by PP, highest first, in order to identify a set of actions that precede the sanctions in all cases and may therefore be attributed with causing the punishment. If the WS range is $[1, 10)$ (i.e. HL=11), the agent has 10 sets of actions that circumstantially have a causal relationship with the sanction. Then, the agent computes the minimum set of actions from these 10 sets to identify actions that are most likely – by virtue of when the actions took place – to constitute the norms. We call this minimum set the candidate norms and it is the intersection of the action sets. The candidate norms thus identified become the input to the norm classification system.

In order to understand the norm identification process in detail, let us consider the event sequences shown in Figure 2. An event sequence (ES) is a sequence of actions that an agent observes another agent to be performing. The first line in the event sequences box on the left is the observation of another agent's actions and their consequences

Algorithm 1: Pseudocode to identify a minimum set of actions that are prohibited

Input: Event Sequences (ES) with punishments
Output: Minimum action set (a minimum set of actions that are prohibited)

```
1  List actionSetForEachWindowSize ⟵ ∅ ;            /* Contains a list of
   actionSets */
2  Set minimumActionSet ⟵ ∅
3  for each window 1 to i in Window Size (WS) do
4      foreach action (a_j) ∈ ES do
5          Calculate Punishment Probability (PP_aj);
6      end
7      Arrange actions by descending order of PP_aj
8      boolean actionSetForCurrentWSFound ⟵ false;
9      Set actionSetForCurrentWS ⟵ ∅;
10     for each action (a_j) ∈ ES do
11         actionSetForCurrentWS.add(a_j);
12         actionSetForCurrentWSFound ⟵
           checkIfActionSetResponsibleForSanctions(actionSetForCurrentWS) ;
13         if actionSetForCurrentWSFound then
14             actionSetForEachWindowSize.add(actionSetForCurrentWS);
15             break;
16         end
17     end
18 end
19 minimumActionSet = findIntersectingActions(actionSetForEachWindowSize)
   ;                          /* intersection of all action sets */
```

(HL=20 and WS=4). The observer sees an agent perform actions ANJI followed by a sanction (denoted by #). Subsequently, the observee performs ARBP and is sanctioned (@) and then it performs FAAF and is sanctioned again ($). The punishment event sequences (PES) occurring in a given history are listed in the middle, and are highlighted in the ES box on the left. There are three PES in the first observation. The unpunished event sequence (UES) in this case is the negation of the PES from the ES, yielding DONLJ. The input to Algorithm 1 is a ES list.

The agent can recognise a sanction. However, it does not know which action (or a set of actions in the case of multiple co-existing norms) has caused the sanction. To find out, it examines WSs 1 to HL-1, since it does know how long it is between violation and sanction (i.e. the delay between the two). For each WS the agent does the following: (i) extract the PES and UES from the history, (ii) calculate the punishment probabilities for all the unique actions in PES (line 5), (iii) order the actions in descending order of punishment probabilities (line 7). (iv) identify a set of actions that can account for all violations (lines 10-18) by adding one action at a time starting with the one with the highest PP_{a_j}, (v) compute the intersection of the action sets (line 19). This intersection is the minimum set of actions that may account for the violations that have resulted in the sanctions.

Table 1: Action sets

Window Size	Action Set	Size of action set
1	[P, D, I, K, E, R, L, Q, G, M, J, H, A, C, S, B, F, N, O, T]	20
2	[D, I, P, S, R, B, T, C, M, G, F, K, L, J, O, N, H, Q, A, E]	20
3	[I, P, D, H, J, S, Q, O, N, T, B, A, K, E, G, F, R, M, L]	19
4	[P, D, I, O, T, M, F, K, G, A, J, B, N, E, L, H, C, Q]	18
5	[P, D, I, H, M, Q, O, R, C, F, J, T, K, N, G, E]	16
6	[P, I, D, N, K, R, F, J, O, B, S, G, T, M, Q, L]	16
7	[P, I, D, L, E, K, J, T, A, N, G, H, M, O, S]	15
8	[I, P, D, S, K, A, M, H, E, B, O, G, R]	13
9	[P, D, I, O, Q, G, E, K, F, A, T, M, S, R]	14
10	[P, D, I, O, K, H, J, B, G, R]	10
11	[P, D, I, J, E, S, O, K, F, Q]	10
12	[P, D, I, J, K, E, Q, A, O, N, F]	11
13	[P, D, I, Q, A, E, O, K]	8
14	[P, D, I, N, E, Q, J]	7
15	[P, D, I, E, R, S, F, K]	8
16	[P, D, I, N, E, J, R]	7
17	[P, D, I, O, E, R]	6
18	[P, D, I, Q, T]	5
19	[P, D, I]	3

Table 1 shows the actions sets obtained by ranging WS from 1 to 19 (i.e. HL=20) for an ES list containing 50000 entries. It shows three columns, the WS in column 1, the action set that was identified for the corresponding WS in column 2 and size of the action set in column 3. The minimum set of actions out of these 19 action sets is identified as the set of candidate norms. The minimum action set (i.e. the minimum set of prohibited actions) here is P, D and I identified at WS=19.

Note that this approach only identifies *potential* norms as an aggregate: that is, P, D and I *together* are considered responsible for the occurrence of the sanctions. It does not identify the salience of the individual norm violations. For example, it does not answer the question of whether P is punished more often than D. Addressing this question forms the focus of the next phase discussed in Section 3.3.

Results We first compare the new algorithm against that is reported in [10]. For these experiments we use three data generation schemes: uniform, random and pre-determined. The last was also used in [10], and thus provides us with a baseline reference against which to evaluate the new approach. A set of 20 actions are considered, with HL=20. At each time step, an agent picks an action to perform, governed by the underlying distribution.

Each experiment comprises three rounds in which the system is configured with one, two or three norms to be discovered. The number of norms in the system is known to the experimenter, but unknown to the norm identification system whose goal is to identify those norms. For each distribution, there are 100 runs each with low (p=0.1), medium (p=0.5) and high (p=0.9) punishment probability, making 300 runs in all. Finally, each run has 50,000 iterations, for each of which one event sequence of length HL is created.

Table 2: Comparison of two algorithms under different distributions

		Uniform		Gaussian		Pre-determined	
No.of. norms	FP or FN	Old	New	Old	New	Old	New
1	FP	0.003	0.000	5.623	0.130	3.033	0.000
1	FN	0.013	0.000	0.057	0.117	0.013	0.000
2	FP	0.003	0.000	5.570	0.043	3.967	0.000
2	FN	0.023	0.000	0.873	0.390	0.023	0.000
3	FP	0.000	0.000	5.187	0.053	1.877	0.000
3	FN	0.030	0.000	1.673	0.580	0.030	0.000

This process depends upon the punishment probability derived from the data set, to identify potential norms, in contrast to that of [10], which depends on the experimenter setting a hard Norm Inference Threshold (which is set to 0.25 here).

We compare the number of false positives (FP) and false negatives (FN) generated by the new and the published prohibition norm identification algorithm in experiments where the system is configured with one, two and three norms. The Generation Window Size (GWS) is set to 19. The results are shown in Table 2.

It can be observed that the new algorithm generally performs better, in that it significantly reduces the number of false positives and false negatives, except in the case of FNs in the one norm configuration under Gaussian distribution (an artifact of the behaviour of the Gaussian model that we explain below). The new algorithm does not produce any FPs or FNs under uniform and pre-determined distributions, while there are some under the normal (Gaussian) distribution. We explain this as follows: under normal distribution, there will be some actions (αs) that happen rarely and that get punished only a few times. Due to randomness there will be other actions (βs) that occur in the same windows as the αs (particularly when window sizes are large). Consequently, the βs can be identified as a reason for the sanction, rather than the αs.

For example, let us assume that there exists an action (γ) whose occurrence probability is low (say 0.0001) and when that action occurs it gets punished. In a run with 50000 iterations and generation window size of 19, this action would occur only 100 times[2]. When the WS is large (say 19), it could happen that another action (say ω) might co-occur with γ just by chance (e.g. ω could have a high occurrence probability). Hence, ω and α might be identified as two actions that are responsible for the sanctions to occur even though ω has nothing to do with sanctions (an error of commission – a false positive). On the other hand if ω is also a reason for a sanction and assuming that its PP is higher than that of γ, then, only ω could be identified as the norm (an error of omission – a false negative). Nevertheless, under Gaussian distribution, the new algorithm performs significantly better.

An agent inferring norms can also predict the delay between the violations and sanctions (i.e. how long after a violation does the sanction occur). In other words, the agent can identify the right WS for identifying the norms, simply by noting the WS at which the minimum action set is normally – because of the small probability of false positives/negatives – found. To illustrate the prediction of the correct WS consider

[2] 50000*19*0.0001=95.

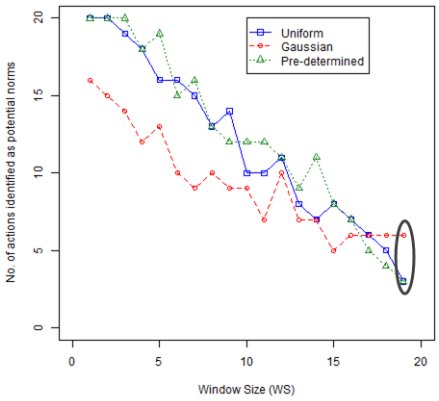


Fig. 3: Generation window size = 19

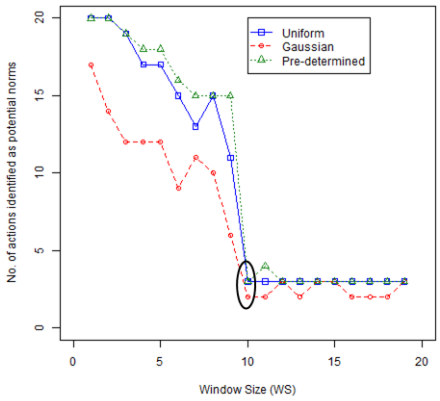


Fig. 4: Generation window size = 10

Figures 3 and 4. They show the size of the action sets identified for WS=19 and 10 respectively. Each of the figures also show the comparisons of the sizes of norm sets that are identified when the underlying action generation distribution is varied (i.e. uniform, Gaussian and pre-determined).

The system was configured with three norms for all the three experiments and the HL was set to 20. Figure 3 shows that when the generation window size[3] was 19 the correct number of norms (i.e. three) is identified at WS=19 (x-axis) for uniform and pre-determined distributions, but under Gaussian there are two FPs (i.e. a total of five norms). Figure 4 shows the correct number of norms with WS=10 for uniform and pre-determined distributions, but Gaussian only identifies two norms (i.e. one FN). It should be noted that in all the three cases, the dip (i.e. the least number of action sets (norms) is found for the first time (highlighted using ellipses in the figures) – 3 in this experiment), occurs at the correct generation window size (19 and 10, respectively in Figures 3 and 4) for uniform and pre-determined distributions. It should be noted that for the Gaussian distribution the dip occurs at the correct generation window sizes (19 and 10). However, the sets of norms identified contain false positives in one case (Figure 3) and a false negative in another (Figure 4). Nevertheless, independent of the distributions, the window size at which the dip occurs has consequences for an agent. In all the three distributions an agent is able to determine the delay between an action and a sanction (i.e. the right WS). This has implications for the amount of processing an agent undertakes in order to identify norms. For example, upon discovering that all norms are punished within a window size of 10, an agent only needs to consider the 10 actions that precede a sanction for further processing (i.e. to identify norms). It does not have to consider window sizes 11 to 19, which it would otherwise for HL=20.

[3] To reiterate, the generation window size (GWS) is the delay between an action and the sanction that follows it. This delay is set by the experimenter.

3.3 Norm classification

Norm classification is based on norm salience. Norm salience is computed based on two axes: frequency of an action in the x-axis and the punishment probability in the y-axis. We propose a simple salience categorization containing four zones of salience (very high, high, low and very low). The graphical representation of the norm classification scheme is given in Figure 5(a).

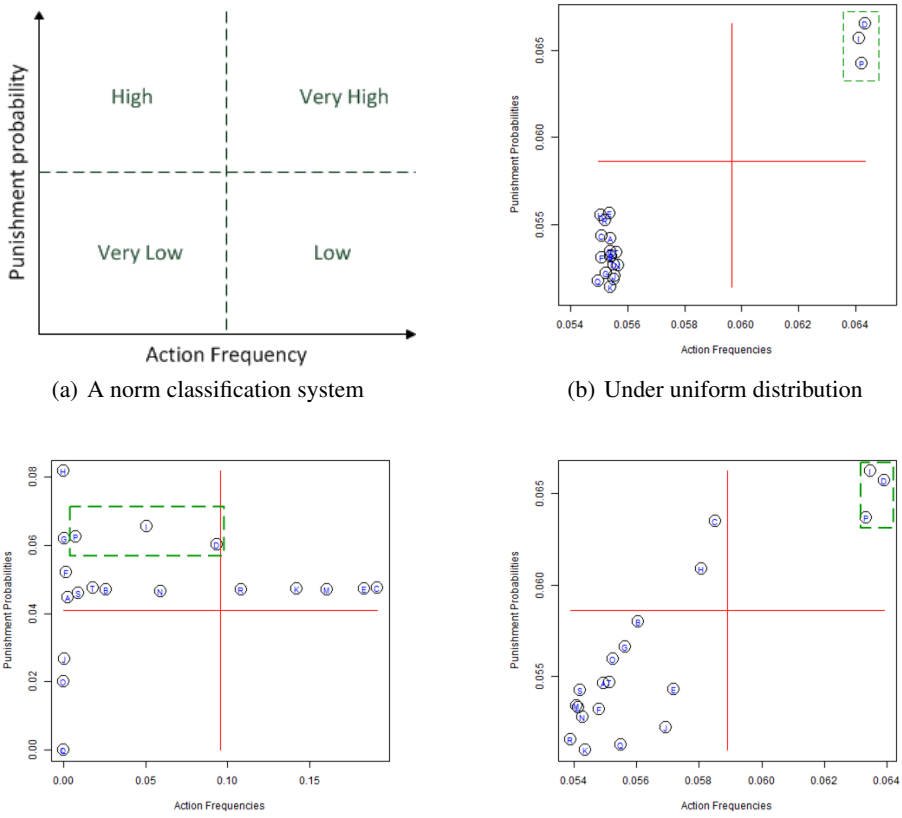
The agent classifies the candidate norms identified in the previous stage. Figures 5(b), 5(c) and 5(d) show example norm classifications for the three distributions considered in this work. It can be observed in Figures 5(b) and 5(d) that all the three norms with which the system is configured (P, D and I) are classified as *very highly* salient norms (highlighted by dashed rectangles). However, under Gaussian distribution, P, D and I are *highly* salient norms. In order to provide a comparison of how normative actions (i.e. P, D and I) and non-normative actions fare on the norm classification scale, we have also presented all the other non-normative actions. It can be observed that there is a clear separation of normative and non-normative actions for uniform and pre-determined distributions, while for Gaussian distribution, the difference isn't clear. There are some actions that occur rarely, but appear along the sanctioned action more often (just by chance) in the gaussian distribution (e.g. action H in Figure 5(c)). Since Algorithm 1 arranges actions based on descending order of the punishment probability the action H will appear in the norms set in this case. This is an example of how false positives are included in the norms set in the case of Gaussian distribution.

Norm salience is a metric for the short term (i.e. how salient is the norm now) and does not account for whether a norm is stable or emerging etc. The purpose of the next step is to identify the life-stage of a norm.

3.4 Norm life-stage detection

We propose that every potential norm should be considered in a historical context. Assuming that we have enough long-term information available, norm life-stage detection can be enabled. Based on historical data, an agent can identify whether a norm is emerging, growing, maturing, declining or decaying. The inspiration for this terminology comes from the life cycle stages of a product [13].

We define the life-stage function below. The function takes three inputs, the cumulative PP of a norm at regular time intervals, a threshold for norm emergence (α) and a threshold for norm growth (β). We specify the decision function in terms of whether: (i) the PP of a norm is higher, the same or lower than its value at the previous census point, (ii) the value of PP in relation to α and β (noting that $0 \leq \alpha < \beta \leq 1$): The life-stage labels are assigned according to the scheme outlined below, with the additional distinction between the start of the decline of a mature norm and the latter stages when its influence is greatly reduced, but not yet decaying. For each label, we also give an



(a) A norm classification system

(b) Under uniform distribution

(c) Under Gaussian distribution

(d) Under pre-determined distribution

Fig. 5: Salience based norm classification across different probability distributions

abbreviation that is used in Figure 6.

$$\text{life\_stage}(PP, \alpha, \beta) = \begin{cases} PP_t > PP_{t-1} \wedge\ PP_t \in [0, \alpha] \rightarrow \text{emerging: E} \\ PP_t > PP_{t-1} \wedge PP_t \in (\alpha, \beta] \rightarrow \text{growing: G} \\ PP_t > PP_{t-1} \wedge\ PP_t \in (\beta, 1] \rightarrow \text{maturing: M} \\ PP_t = PP_{t-1} \rightarrow \text{previous value} \\ PP_t > PP_{t-1} \wedge\ PP_t \in (\beta, 1] \rightarrow \text{declining but mature: D1} \\ PP_t > PP_{t-1} \wedge PP_t \in (\alpha, \beta] \rightarrow \text{declining and weak: D2} \\ PP_t < PP_{t-1} \wedge\ PP_t \in [0, \alpha] \rightarrow \text{decaying: D3} \end{cases}$$

Figure 6 shows how life-stages are identified for a sample set of datapoints (being the cumulative punishment probabilities for the violation of a norm (y-axis) over time (x-axis)) using the life-stage function. The values of α and β are set at 0.25 and 0.75, respectively. At time t_1 the punishment probability is 0.96. At t_2 the value falls to 0.17

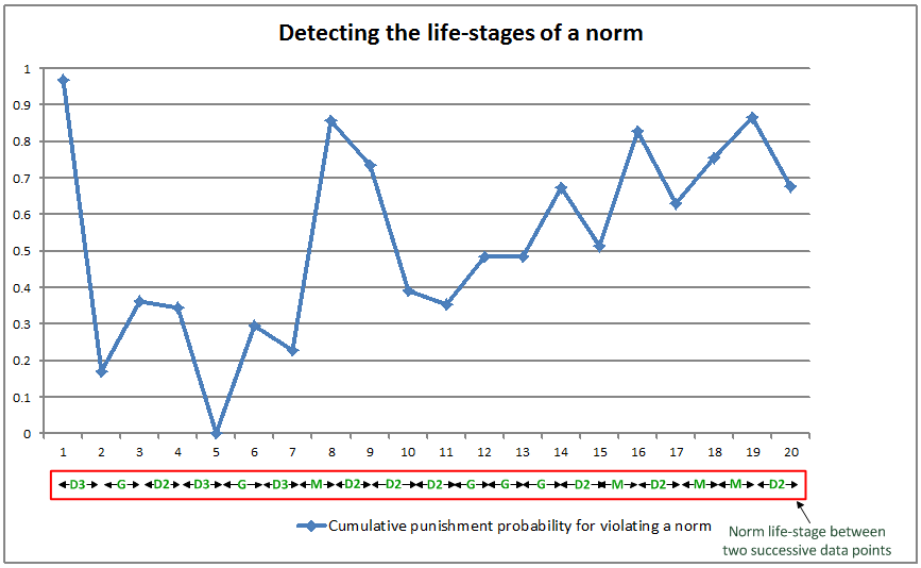


Fig. 6: An example of life stage detection for a sample norm

which is less than α, giving a life-stage of decaying. The life-stage at time t_2 (between data points 2 and 3) is growing and the life-stage at time t_3 is declining. Using the life-stage function, we can identify the life-stages between the pairs of consecutive data points. The norm life-stages at various time points are shown at the bottom of Figure 6.

3.5 Norm recommendation

Using the results of the previous two stages, this stage can provide norm recommendation to software agents or to human users. A simple heuristic based recommendation is set out in Table 3. The tick marks represent occasions where a norm is recommended to an agent and the crosses represent occasions where a norm is not recommended to an agent. Two sample recommendations are given in Table 4. The first column corresponds to a situation where an agent has been advised to adopt the norm *prohibit(X)* (i.e. the agent has been advised to abstain from performing action X) whose salience is high and the norm is growing. The second column corresponds to a situation where an agent has been recommended to ignore the norm proscribing action Y (represented by *ignore(prohibit(Y))*) since the norm salience is low and the norm is decaying.

It should be noted that the heuristics used in this phase will be configured by the designer of the system. Also, the norm recommendations are provided to human users, who may or may not decide to honour these norms. The approach proposed here is a 'non-intrusive' approach where recommendation is provided to human agents without forcing them to abide to the norms (i.e. autonomous decision making is honoured).

Table 3: Heuristics for norm recommendation

	Emergent	Growing	Maturing	Declining	Decaying
Very High (VH)	✓	✓	✓	✓	✓
High (H)	✓	✓	✓	✓	✗
Low (L)	✓	✓	✓	✗	✗
Very Low (VL)	✗	✓	✓	✗	✗

Table 4: A sample of recommended norms

Sample 1	**Sample 2**
Norm: prohibit(X)	Norm: prohibit(Y)
Salience: High	Salience: Low
Life-stage:Growing	Life-stage: Decaying
Recommendation: adopt(prohibit(X))	Recommendation: ignore(prohibit(Y))

4 Discussion and conclusion

The main contribution of the paper is the proposal and the evaluation of what we regard as a more realistic model for a norm recommendation system. This paper has presented a comprehensive four-phase approach for recommending norms in software agent societies. Based on the data observed by an agent, it first identifies candidate norms. Second, these norms are then classified based on their salience (i.e. the extent of punishments imposed for violations). Third, the life-stage of the norm is identified. Finally, based on salience and life-stage detection, a heuristic method is used to recommend norms. The recommended norms can then be used by human or software agents for choosing actions. The operationalization of the four phases of the recommendation system is evaluated using simulated data. Additionally, the paper also proposes a new algorithm for norm identification which overcomes the limitations of published work.

There are some shortcomings to the current work that we intend to address in the future. First, there could be a combination of actions that are the reason for a sanction (as opposed to just one action as assumed here). Second, the subject of the method described here is prohibition norms: it is technically fairly straightforward to apply this approach also to obligation norms. Third, the demonstration of the architecture is based on simulations. A concrete implementation in the context of an agent-based application is a desirable next step (e.g. in conjunction with the revision mechanism of Li et al. [5]). Fourth, different models for some of the phases of norm recommendation can be considered. For example, in the norm identification phase, norms are ordered based on the descending order of punishment probability. A weighted approach that ranks candidate norms not only based on punishment probability, but also the action frequency can be considered. Fifth, enabling agents to auto-tune parameters in certain scenarios will be beneficial. For example, an agent can decide that it is only interested in highly salient norms and may decide to ignore all the other norms. Hence, the agent will only identify the life-stages of those norms that are highly salient, thus eliminating additional computation required for evaluating other norms. Sixth, the norm life-cycle model presented in the paper is coarse grained using a small number of discrete stages. The discrete model could be replaced with a continuous model that the designer might specify by means of

a malleable graph interface. What these points serve to highlight is that the contribution here is not just in the various decision procedures and the plausibility of the results they together deliver, but also in form of a framework into which alternative procedures can be plugged to explore a rich landscape of norm discovery.

References

1. R. Axelrod. *The complexity of cooperation: Agent-based models of competition and collaboration*. Princeton University Press, 1997.
2. D. Corapi, M. De Vos, J. Padget, A. Russo, and K. Satoh. Norm refinement and design through inductive learning. *Coordination, Organizations, Institutions, and Norms in Agent Systems VI*, pages 77–94, 2011.
3. N. Criado, J. M. Such, and V. Botti. Providing agents with norm reasoning services. In *Third International Workshop on Infrastructures and Tools for Multiagent Systems (ITMAS - 2012)*, 2013.
4. J. M. Epstein. Learning to be thoughtless: Social norms and individual computation. *Computational Economics*, 18(1):9–24, 2001.
5. T. Li, T. Balke, M. D. Vos, J. Padget, and K. Satoh. A model-based approach to the automatic revision of secondary legislation. In *The 14th International Conference on Artificial Intelligence and Law, Proceedings of the Conference, June 10-14, 2013, Rome, Italy*, pages 202–206, 2013.
6. J. Morales, M. López-Sánchez, and M. Esteva. Using experience to generate new regulations. In *Proceedings of the Twenty-Second international joint conference on Artificial Intelligence-Volume Volume One*, pages 307–312. AAAI Press, 2011.
7. N. Oren and F. Meneguzzi. Norm identification through plan recognition. In *Coordination, Organization, Institutions and Norms in Agent Systems (COIN 2013@AAMAS)*, 2013.
8. B. T. R. Savarimuthu and S. Cranefield. Norm creation, spreading and emergence: A survey of simulation models of norms in multi-agent systems. *Multiagent and Grid Systems*, 7(1):21–54, 2011.
9. B. T. R. Savarimuthu, S. Cranefield, M. A. Purvis, and M. K. Purvis. Obligation norm identification in agent societies. *Journal of Artificial Societies and Social Simulation*, 13(4):3, 2010.
10. B. T. R. Savarimuthu, S. Cranefield, M. A. Purvis, and M. K. Purvis. Identifying prohibition norms in agent societies. *Artificial Intelligence and Law*, pages 1–46, 2012.
11. B. T. R. Savarimuthu, S. Cranefield, M. K. Purvis, and M. A. Purvis. Norm emergence in agent societies formed by dynamically changing networks. *Web Intelligence and Agent Systems*, 7(3):223–232, 2009.
12. S. Sen and S. Airiau. Emergence of norms through social learning. In *Proceedings of the Twentieth International Joint Conference on Artificial Intelligence (IJCAI)*, pages 1507–1512. AAAI Press, 2007.
13. L. Theodore. Exploit the product life cycle. *Harvard business review*, 43:81–94, 1965.
14. D. Villatoro, S. Sen, and J. Sabater-Mir. Topology and memory effect on convention emergence. In *WI-IAT '09: Proceedings of the 2009 IEEE/WIC/ACM International Joint Conference on Web Intelligence and Intelligent Agent Technology*, pages 233–240, Washington, DC, USA, 2009. IEEE Computer Society.
15. C.-C. Wang and C.-H. Wang. Helping others in online games: Prosocial behavior in cyberspace. *Cyberpschology, Behavior, and Social Networking*, 11(3):344–346, 2008.