Dealing with Mislabeling via Interactive Machine Learning

Wanyi Zhang, Andrea Passerini & Fausto Giunchiglia

KI - Künstliche Intelligenz
German Journal of Artificial Intelligence
- Organ des Fachbereichs "Künstliche Intelligenz" der Gesellschaft für Informatik e.V.

ISSN 0933-1875

Künstl Intell
DOI 10.1007/s13218-020-00630-5

Springer

KI - Künstliche Intelligenz
https://doi.org/10.1007/s13218-020-00630-5

DISSERTATION AND HABILITATION ABSTRACTS

Dealing with Mislabeling via Interactive Machine Learning

Wanyi Zhang[1] · **Andrea Passerini**[1] · **Fausto Giunchiglia**[1]

Abstract
We propose an interactive machine learning framework where the machine questions the user feedback when it realizes it is inconsistent with the knowledge previously accumulated. The key idea is that the machine uses its available knowledge to check the correctness of its own and the user labeling. The proposed architecture and algorithms run through a series of modes with progressively higher confidence and features a conflict resolution component. The proposed solution is tested in a project on university student life where the goal is to recognize tasks like user location and transportation mode from sensor data. The results highlight the unexpected extreme pervasiveness of annotation mistakes and the advantages provided by skeptical learning.

Keywords Interactive learning · Knowledge and learning · Managing annotator mistakes

1 Introduction

The performance of supervised learning algorithms crucially depends on the quality of the labeling of the data they are trained on. A perfectly labeled training set is a condition rarely met in real-world scenarios. Most modern supervised learning approaches can tolerate a small fraction of mislabelled training instances. The implicit assumption made in (even recent) mainstream machine learning and also in [7] is that annotators are experts. However, with the use of machine learning becoming viral, more and more applications are being developed where the tagging is provided by non-expert users. The research in social sciences provides evidence of the unreliability of people in providing correct answers when asked [17, 18]. The main motivations for this phenomenon relate to the users' response biases, e.g., conditioning, memory bias, and sometimes also unwillingness to report. The work in [9] provides evidence of the fact that these phenomena also apply when users are asked to tag sensor values in a pervasive system scenario. This work relates the response quality to cognitive bias, e.g., the inadequate recall of respondents when annotating and to carelessness, namely not putting enough attention to providing the answer, e.g., because of hurriedness.

Our work presented in this paper is part of a long-term series of experiments aimed at studying the University student life. As detailed in the evaluation section, since the beginning, it was clear that an unexpectedly high percentage of the labels provided by students was unreliable. The goal of the research described here is to minimize the effects of this overwhelming amount of mislabeling. The key idea is to design a model that allows machines to interact with user and use all its available knowledge to check the correctness of its own prediction and of the label provided by the user. By keeping track of the sequence of wrong and right answers, the machine builds a measure of confidence towards itself and the user, which is then used, in the case of a contradiction, to decide what is actually the case. In this context, by available knowledge we mean both the knowledge inductively built out of the previous learning activity and the knowledge which may come from third parties or may be built-in as a priori knowledge.

✉ Wanyi Zhang
wanyi.zhang@unitn.it

Andrea Passerini
andrea.passerini@unitn.it

Fausto Giunchiglia
fausto.giunchiglia@unitn.it

[1] DISI, University of Trento, Trento, Italy

Published online: 11 January 2020

2 Related Work

While traditional approaches to concept learning assume perfectly labeled training sets, most recent supervised learning techniques can tolerate a small fraction of mislabelled training instances (see for instance [7]). A common solution consists in designing learning models which are robust to (some) label noise [6]. In particular, by averaging predictions of multiple learners, ensemble methods usually perform well in terms of noise robustness [3, 12]. In this line of thought, the robustness of random forests, the ensemble method used in this paper, has recently been shown both theoretically and empirically [8]. Nonetheless, label noise badly affects the performance of learning algorithms [11]. Our approach diverges from existing solutions since it involves an interactive error correction phase. This process allows tolerating a much larger amount of noise, achieving substantial improvements over the previous work.

The field of statistical relational learning [2] deals with the integration of symbolic and sub-symbolic approaches to learning. Frameworks like Markov Logics [14], Semantic-Based Regularization [4] or Learning Modulo Theories [16] combine logical rules or other types of constraints with learnable weights to encourage predictions consistent with the available knowledge. Our main difference is that we use knowledge in an interactive way to identify potential errors in user feeback, and activate a conflict resolution phase to solve such controversies.

While many machine learning approaches assume an expert user, it is not the case in other areas of research, e.g., mobile crowdsensing [10], where users collect and share sensed data and their annotations via their smartphones. A relevant issue here is assessing the quality of users' annotations (see [13] for a comprehensive review). However, the focus in these works is on gathering reliable information about locations or events of common interest among a set of users. The key difference from this work is that we focus on personal data, e.g., personal context and activities, to be used by the user herself. Thus, the quality of a user's annotation cannot be evaluated by comparing it with other annotations from the crowd (which would be very hard if not impossible, since we deal with personal data) but, rather, by comparing it with the machine's knowledge.

3 The SkeL Main Algorithm

We propose Skeptical Learning (SkeL) and implement it as a multi-layer architecture [20]. The key intuition is that the human annotator(s) and the machine learning algorithm(s) are considered as interpretation channels that provide their own fallible perspective on what the case is in the real world.

In this section, without loss of generality, we assume that there is a single property of interest P, e.g., the location of the user at a certain time. We represent by $\mathcal{Y}$ the set of possible values for this property.

Algorithm 1 Skeptical Learning (SkeL)

```
1:  procedure SkeL(θ)
2:      init c^u = 1, c^p = 0
3:      while trainMode(c^p, c^u, θ) do
4:          x_t = sensorReading()
5:          y_t = askUser()
6:          ŷ_t = Pred(x_t)
7:          train(x_t, y_t)
8:          update(c^p, ŷ_t, y_t)
9:      while refineMode(c^p, c^u, θ) do
10:         x_t = sensorReading()
11:         y_t = askUser()
12:         ŷ_t = Pred(x_t)
13:         solveConflict(c^p, c^u, x_t, ŷ_t, y_t)
14:     while True do
15:         x_t = sensorReading()
16:         ŷ_t = Pred(x_t)
17:         if min_{ŷ'_t ∈ SMers(ŷ_t)} Conf(x_t, ŷ'_t, c^p_{ŷ'_t}) ≤ θ then
18:             y_t = askUser()
19:             solveConflict(c^p, c^u, x_t, ŷ_t, y_t)
```

We model SkeL as an algorithm which takes input a continuous stream of sensor data as they are stored in Stream Data Storage. The pseudocode of SkeL is reported in Algorithm 1. The algorithm can be in one of three modalities which, for simplicity, we assume are activated sequentially, namely: Train mode performed in usual supervised learning, Refine mode where it checks the quality of the user answers and, under certain conditions, it challenges them, and the Regime mode where it starts being autonomous and only queries the user for particularly ambiguous instances. The algorithm takes as input a confidence threshold of θ. It starts by initializing the user confidence to one and the predictor confidence to zero for all classes ($|\mathbf{c}^{\mathrm{u}}| = |\mathbf{c}^{\mathrm{p}}| = |\mathcal{Y}|$). Then the training phase begins. The algorithms collect sensor readings ($\mathbf{x}_t$) to be used as input for the predictor.

The prediction procedure Pred is implemented as a hierarchy of classifiers matching the schematic knowledge (SK) ontology, which contains general knowledge about the world. There is one multiclass classifier for each internal node in the ontology (bold contour nodes in Fig. 1), discriminating between its children. Prediction starts from the root classifier and progresses down in the hierarchy following the highest scoring class at each node, until a leaf node is reached which is the class eventually predicted.

The system then asks the user for a label (y_t) to be used as ground truth. The input–output pair trains the predictor by following the train procedure. Note that thanks to the SK, the system can infer all the labels implied by that provided by the user, i.e., all those from the root to the user label. Each

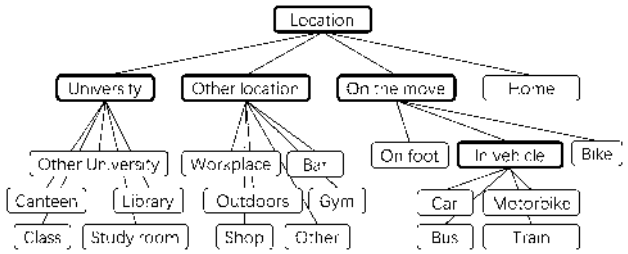


Fig. 1 Ontology of the labels used in the experiment. Bold contours correspond to classifiers in the PRED procedure

classifier in the path is thus retrained with the addition of its corresponding input–output pair during a TRAIN procedure.

After training, the confidence of the predictor is updated using the UPDATE procedure, receiving as input the ground truth label and the predicted one before the training step. The UPDATE procedure takes as input a confidence vector, a tentative label ($\hat{y}_t$) and a ground truth label (y_t), and updates the confidence vector according to the relationship between the two labels. The new confidence vector is as a label-wise running average accuracy over the current and past predictions, for a certain window size d. Confidence updates are applied to all implied label pairs according to the SK, i.e., those from the root to the predicted (respectively) ground truth label.

The confidence in a prediction y for an input $\mathbf{x}$ is the product of the score of the prediction times the confidence c^{p}_y the predictor has in predicting that label:

$$\textsc{Conf}(\mathbf{x}, y, c^{\mathrm{p}}_y) := c^{\mathrm{p}}_y \cdot f_{\textsc{parent}(y)}(\mathbf{x}, y) \tag{1}$$

When a predicted label is compared with a user provided label, care must be taken in making a sensible comparison. The system recovers all the labels in the hierarchy up to the first common root, i.e., the least common subsumer [1] and compares them instead of the original ones. Thus, for instance, in the previous example, Train implies in vehicle which is then compared to on foot as they are both children of on the move.

The system remains in training mode as long as the expected probability of contradicting the user does not exceed the threshold:

$$\begin{aligned}&\textsc{trainMode}(\mathbf{c}^{\mathrm{p}}, \mathbf{c}^{\mathrm{u}}, \theta)\\ &\quad := E[\mathbb{1}(\textsc{Conf}(\mathbf{x}, \hat{y}, c^{\mathrm{p}}_{\hat{y}}) > c^{\mathrm{u}}_y \cdot \theta)] \leq \theta\end{aligned} \tag{2}$$

where $\mathbb{1}(\varphi)$ evaluates to one if φ is true and zero otherwise, and the expectation is taken over all inputs seen so far. The labels to be compared are obtained as $(\hat{y}, y) = \textsc{LCS\_Children}(\textsc{Pred}(\mathbf{x}), y_u)$, where $\textsc{Pred}(\mathbf{x})$, the predicted label for input $\mathbf{x}$, y_u is the label provided by the user for that input, and the LCS_CHILDREN procedure outputs a pair of implied predicted/user labels which are children of the least common subsumer. The user is contradicted when the confidence in the predicted label exceeds a factor θ of the confidence of the user in her own label.

When the system enters the refine mode, it keeps asking the user for labels, but it starts to compare them with its own predictions. The SOLVECONFLICT procedure deals with this comparison, and will be described in detail later. The refinement stage stops when the predictor is confident enough to stop asking for user feedback on every input, but selectively query the user on "difficult" cases. In general, it should be the user who decides when to switch modes, thus trading off system maturity and cognitive load. A simple fully automated option, similar to the one used for the train mode consists in staying in refine mode as long as the expected probability of querying the user exceeding the threshold:

$$\begin{aligned}&\textsc{refineMode}(\mathbf{c}^{\mathrm{p}}, \mathbf{c}^{\mathrm{u}}, \theta)\\ &\quad := E[\mathbb{1}(\textsc{Conf}(\mathbf{x}, \hat{y}, c^{\mathrm{p}}_{\hat{y}}) \leq \theta)] \geq \theta\end{aligned} \tag{3}$$

again with expectation taken over all inputs are seen so far. Note that given that the system has no access to the user label here, it takes a conservative approach and considers the smallest confidence among the ones of the subsumers (SMERS) of its (leaf) prediction, see line 17 in Algorithm 1.

When leaving the refine mode, the system enters the regime, where it stays indefinitely. Here, the system stops asking feedback for all inputs, and an active learning strategy [15] begins. The system queries the user only if the (minimal) confidence in a certain prediction is below the "safety" threshold θ. If the system decides to query the user, it includes the tentative label in the query, and then behaves as in refinement mode, calling SOLVECONFLICT to deal with the comparison between the predicted and the user labels.

4 The Conflict Management Algorithm

Algorithm 2 Procedure for solving labeling conflicts.

```
1:  procedure SOLVECONFLICT(c^p, c^u, x, ŷ, y)
2:      if ISCOMPATIBLE(ŷ, y) then
3:          y* = CONSENSUS(ŷ, y)
4:          UPDATE(c^p, ŷ, y*)
5:          UPDATE(c^u, y, y*)
6:      else
7:          (ŷ', y') = LCS_CHILDREN(ŷ, y)
8:          if CONF(x, ŷ', c^p_ŷ') ≤ c^u_y' · θ then
9:              TRAIN(f, x, y)
10:             UPDATE(c^p, ŷ, y)
11:         else
12:             y* = CHALLENGEUSER(ŷ)
13:             if not ISCOMPATIBLE(ŷ, y*) then
14:                 TRAIN(x_t, y*)
15:             UPDATE(c^p, ŷ, y*)
16:             UPDATE(c^u, y, y*)
```

The SOLVECONFLICT procedure is described in Algorithm 2. It takes as input the predictor and user confidence vectors $\mathbf{c}^{\mathrm{p}}$ and $\mathbf{c}^{\mathrm{u}}$, an input $\mathbf{x}$ with its predicted label ($\hat{y}$) and the label

given by the user (y). It first compares the two labels according to the isCompatible procedure. As the SK encodes a subsumption hierarchy for the property of interest, the procedure returns true if the two labels are the same or if one subsumes the other. In case the labels are compatible, a consensus label is taken as the ground truth, and the predictor and user confidences are updated accordingly. A natural choice for the consensus (the one used in our experiments) is choosing the more general among the two labels.

If the two labels are not compatible, a labeling conflict arises. In case the confidence of the prediction is not large enough, the user label is taken as ground truth, the predictor is retrained with this additional feedback, and its confidence is updated accordingly. Otherwise, the system contradicts the user, advocating its own prediction as the right one[1]. The user is now responsible for solving the conflict. She can decide to stick to her own label, realize that the machine is right and converge on the predicted one, or provide a third label as a compromise. Note that the user can, and often will because of imperfect memories, make a prudent choice and return an intermediate node of the label hierarchy rather than a leaf. As we are assuming a non-adversarial setting, and we aim at providing a support to the user rather than a replacement for her, the system eventually trusts the newly provided label (even if unchanged), which becomes the ground truth. At this point, a compatibility check is made to verify whether a retrain step is needed, and the predictor and user confidences are updated.

5 Dataset

The evaluation of the SkeL architecture presented in this paper is based on a dataset collected in an experiment which main objective was to understand the empirical gap concerning students' time allocation and academic performance. The data was collected using the i-Log mobile application [19] that can simultaneously acquire data from up to thirty sensors on the smarpthone, both hardware (e.g., GPS) and software (e.g., running applications). The i-Log also allows to administer time diaries to the participants asking about their activities, location and social relations at fixed time intervals. The collected answers are then used as the user's labels in the machine learning algorithms of the SkeL architecture.

Initially, 312 students who enrolled in the first academic year of the bachelor courses active in 2016 at our University were asked to participate in this experiment. 104 students from this initial population fulfilled the three specific criteria that were defined: (1) to agree on sharing their socio-demographics, as shown in Table 1, and other characteristics, e.g., psychological and time use related; (2) to attend lessons during the period the experiment took place in order to describe their daily behavior during a normal day at the university, and (3) to have an Android smartphone (iOS was not supported at the time the experiment was done). In the end, the final sample consisted of 72 students that reflected the general population of freshman year students of our University in terms of gender and departments.

Table 1 Socio-demographics of students from the experiment

Gender	Male	61.1%
	Female	39.9%
Departments	Scientific	56.9%
	Humanities	43.1%
Scholarship	True	37.5%
	False	62.5%
Age	Min	19
	Max	22

6 Results and Evaluation

In order to have a ground-truth independent of both the predictor and the user annotation, we focused on predicting the locations participants visited during the two weeks of the experiment, as these were easier to verify with respect to, e.g., activities. We used a hierarchy of labels from SK that accounts for both the user location and the user transportation means, as reported in Fig. 1. We computed ground-truth labels for University by using maps of University buildings. The ground-truth for user's home was identified by clustering the locations she labels as home via DBSCAN [5] and choosing the cluster where she spends most of the time during the night. By using Google data (for users that agreed to provide them, i.e. 32 out of 72), we could also detect if the user was on the move, and whether she was on foot, by bike or in vehicle. Finally, the other location label was assigned to the cases in which none of the other three main locations was detected. Note that SkeL has no access to the information used to compute the ground truth.

The classifier for each internal node in the ontology was implemented as a random forest classifier, which is known to be robust to labeling noise. The window size for confidence computation was set to be infinite ($d = \infty$). In order to achieve a reasonable trade-off between accurate training and cognitive effort for the user, the confidence parameter θ was set to 0.2.

[1] In order to support its argument, the machine could provide some sort of explainable critique to the user feedback, in terms of counter-examples or evidence of inconsistencies with respect to the SK. This is a promising direction for future research.

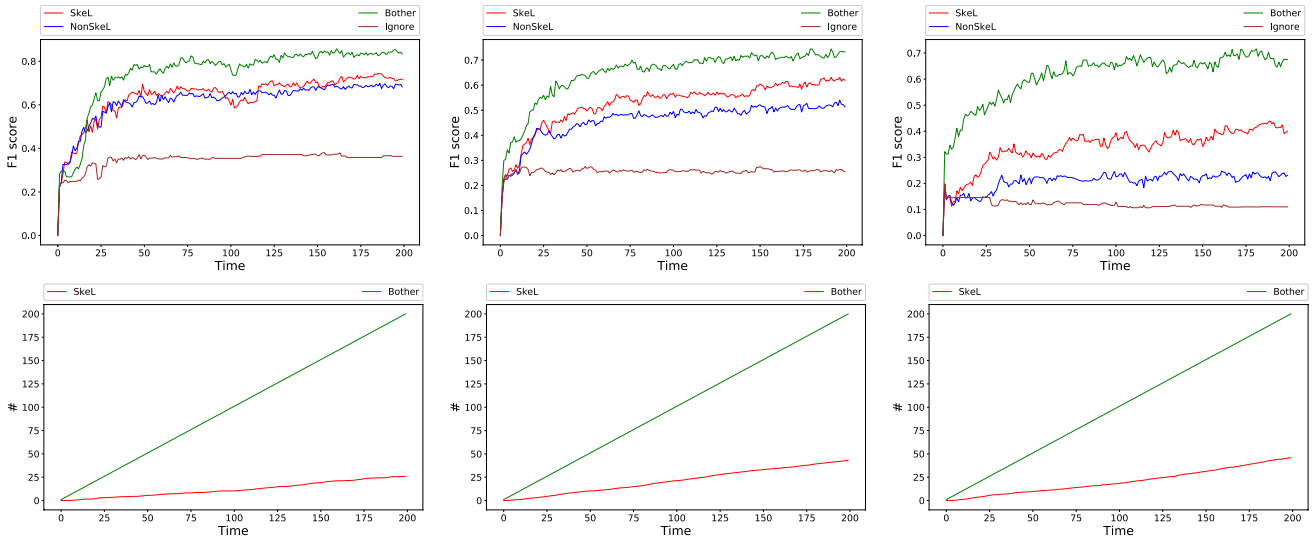


Fig. 2 Results averaged over users with at most 10% (left column), from 10 to 25% (middle column) and more than 25% (right column) labelling errors. First row: F_1 scores on left-out data. Second row: number of times user is contradicted. The Time axes represent the number of iterations the algorithm is going through

6.1 Comparing SkeL with Three Alternative Strategies

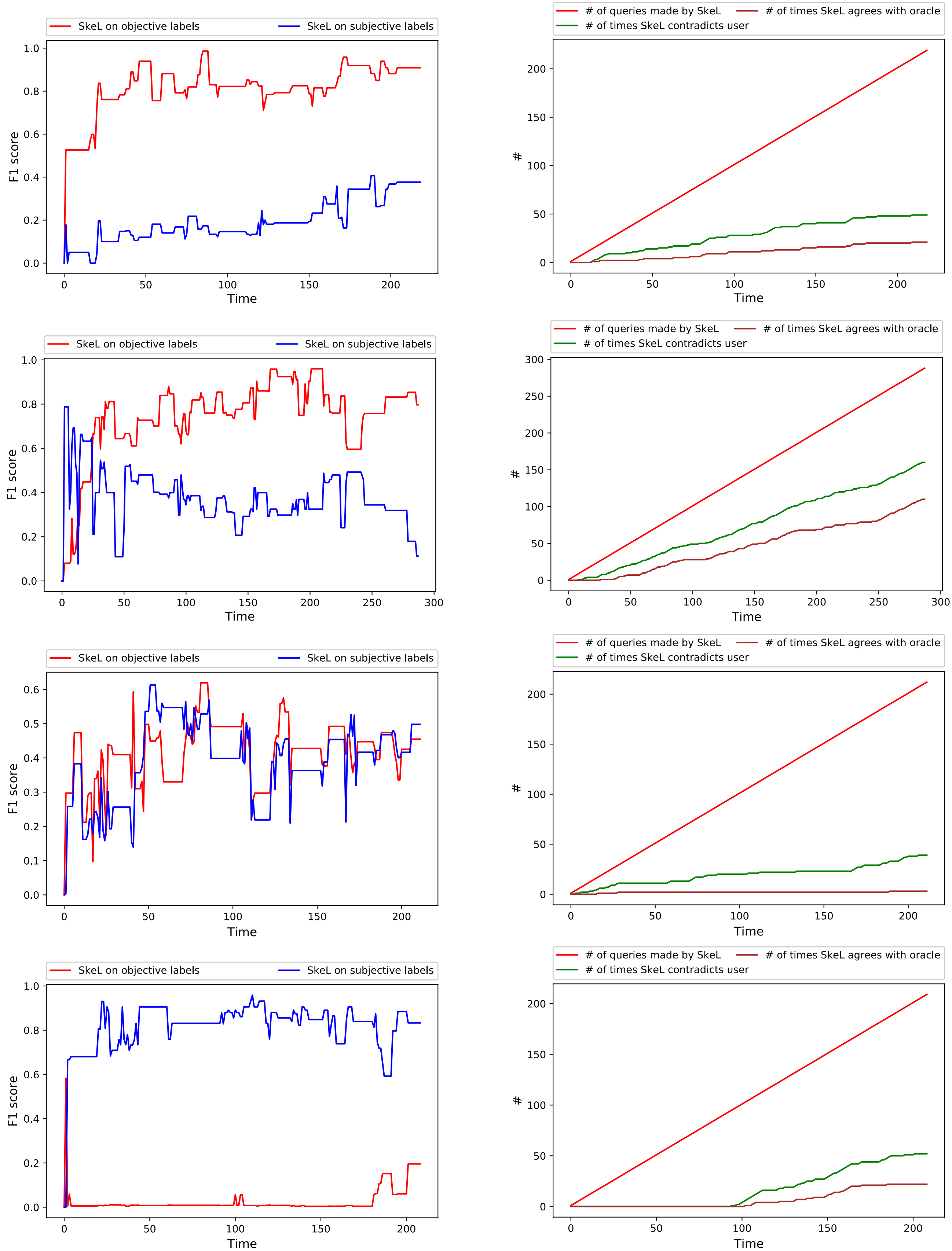
We compared SkeL with three alternatives:

- NonSkeL, that never contradicts the user (obtained by replacing SOLVECONFLICT with a train and update step, as happens in the training phase);
- Ignore, that simply ignores any example for which a conflict arises (obtained by removing everything from the ELSE onwards in Algorithm 2);
- Bother, that always contradicts the user (obtained by calling CHALLENGEUSER after all ASKUSER calls, and removing SOLVECONFLICT).

When comparing user annotations with the ground truth, we found a surprisingly high proportion of inconsistencies. Table 2 shows some statistics on the percentage of users with different amounts of labeling noise. In order to estimate the effect of this large and very diverse proportion of labelling errors on the performance of the system, we divided the set of users in the three groups reported in the table. Figure 2 reports the results of SkeL and of the three alternatives for an increasing number of iterations. Each column represents the results for a different group of users: at most 10% labelling errors (left), 10–25% (middle), more than 25% (right). The first row reports f_1 scores averaged over all users in the corresponding group with a number of training samples greater than 200. The score for each user is computed on a fixed test set, namely the latest 15% of the all data available for that user, which was not used for training. This score provides an estimate of the performance of the algorithms when making predictions on future data. Note that we consider a label as correctly predicted if it is compatible with the ground-truth label, because this is the only type of reliable supervision we have access to. Results clearly indicate that our skeptical algorithm (red curve) consistently outperforms a non-skeptical alternative (blue curve). As expected, the advantage is moderate for users with a relatively small fraction of labelling errors (left figure), and grows with the unreliability of the users, reaching a gap of 0.20 for users with more than 25% labelling errors (right figure). Ignoring conflicting cases (brown curve) is clearly not an option, as it achieves the worst performance in all cases. On the other hand, having always access to correct supervision, Bother (green curve) clearly achieves the highest performance. However SkeL is capable of getting reasonably close to this upper bound when enough iterations are provided, at a fraction of the cost in terms of user effort. The second row reports the number of times the user is contradicted for SkeL (red curve) and for Bother (green curve), for which they are simply the number of iterations. SkeL clearly contradicts more when facing increasingly unreliable users. However, the cost remains substantially lower than the one of Bother, going from 13 (left figure) to 23% (right figure).

Table 2 Percentage of users with each label noise level

Label noise level	$\leq 10\%$	10–25%	$\geq 25\%$
Users	21.6%	51.4%	27.0%



◂**Fig. 3** Results for four different prototypical users, namely an inattentive user, a predictable user, a reliable user, and a tricksy user (from the top to the bottom). The images on the left report the f_1 scores with respect to different labels, and the ones on the right report the information about the number of queries and the agreement with the user and the oracle

6.2 Variability of Users

The objective labels are the labels provided by the oracle and the subjective labels are the ones provided by the user. In this section, we investigate the performance of the SkeL algorithm with respect to these two types of labels. By analysing performance graphs of every single user, we can identify four different patterns that are related to distinct behavior patterns. Figure 3 shows the results for these four prototypical users. Each row refers to a specific user. Left figures report f_1 scores with respect to the objective labels and the subjective labels. Figures on the right column report the number of queries by SkeL, the number of times when SkeL challenges the user, and the number of times when SkeL agrees with the oracle label.

- Inattentive user: The results of the first row in Fig. 3 show that the highest score is achieved by the SkeL algorithm evaluated on objective labels. This behavior can be explained in terms of an inattentive user, who often provides subjective labels that are different from the objective ones (difference between red and blue curves). The inconsistency of the user is also reflected in the right graph of the first row of Fig. 3, showing that half of times when the user is contradicted (because there is no agreement) the predictor agrees with the oracle. This is the type of user benefits the most from SkeL algorithm. Note that the fact that SkeL manages to correct user inconsistencies indicates that the system reaches a sufficient confidence to start challenging the user, i.e., the user is a "detectable" inconsistent one.
- Predictable user: The second case is a particularly interesting one. Initially, the algorithm learns to predict subjective labels with a high accuracy (blue curve is higher than the red curve). This happens because the user is consistent in providing feedback, but her subjective labels are largely different from the objective ones. At a certain point, the system starts challenging the user and soon afterward (around iteration 40), the system learns to predict objective labels with an higher accuracy with respect to subjective ones. We refer to this user as "predictable". When the system receives the appropriate feedback, objective labels can be predicted with high accuracy. This is confirmed by the high number of times when the predictor and the oracle agree (brown curve with high value in the right figure). A predictable user is thus another case in which the benefits of SkeL are substantial, even if it takes some time for the system to figure out the discrepancy between subjective and objective labels.
- Reliable user: The third row of Fig. 3 shows that, for of this user, the performance of the SkeL on objective and subjective labels are roughly the same and have similar trend. This is because the user is already reliable in providing initial feedback, as can be seen by the substantial overlap between the red and blue curves. Indeed, the user is contradicted only occasionally (green curve in the right figure), and even rarer are the cases in which the oracle agrees with the predictor against the subjective label of the user (brown curve). This is a user for whom SkeL is not helpful, but also not harmful.
- Tricksy user: The last one is a case in which the SkeL algorithm completely fails to predict user actual behavior. The big gap between between blue and red curves shows the difference performance between subjective and objective labels. Note that the user will be contradicted only when it goes into refine or regime mode. The first part of the green curve in the right figure stays in zero, which means the machine stays in the train mode in the first 90 iterations. The algorithm keeps learning from subjective labels, even when it goes into the next mode and is given the chance to question user labeling. The right figure shows that this chance is rarely taken by the algorithm, and in few cases it discovering that prediction agrees with oracle. The user here succeeds in fooling the system by convincing it with the correctness of her own feedback.

7 Conclusion

In this paper we introduced Skeptical Learning as a paradigm for dealing with the unreliability of users when providing labels that describe their personal context. The fundamental idea is to use the available knowledge when deciding what is more reliable between the output of the machine learning algorithms and the user input, and to engage in a conflict resolution phase when a controversy arises. Experimental results show the pervasiveness of mislabelling when dealing with feedback from non-expert users, and the effectiveness of Skeptical Learning in addressing the problem as compared to existing approaches to deal with noisy labels.

Acknowledgements This research has received funding from the European Union's Horizon 2020 FET Proactive project "WeNet–The Internet of us", Grant Agreement No: 823783.

References

1. Baader F, Calvanese D, McGuinness D, Patel-Schneider P, Nardi D (2003) The description logic handbook: theory, implementation and applications. Cambridge University Press, Cambridge
2. Bakir GH, Hofmann T, Schölkopf B, Smola AJ, Taskar B, Vishwanathan SVN (2007) Predicting structured data (neural nnformation processing). The MIT Press, Cambridge
3. Dietterich TG (2000) An experimental comparison of three methods for constructing ensembles of decision trees: bagging, boosting, and randomization. Mach Learn 40(2):139–157. https://doi.org/10.1023/A:1007607513941
4. Diligenti M, Gori M, Saccà C (2017) Semantic-based regularization for learning and inference. Artif Intell 244:143–165. https://doi.org/10.1016/j.artint.2015.08.011. http://www.sciencedirect.com/science/article/pii/S0004370215001344. (Combining constraint solving with mining and learning)
5. Ester M, Kriegel HP, Sander J, Xu X et al (1996) A density-based algorithm for discovering clusters in large spatial databases with noise. Kdd 96:226–231
6. Folleco A, Khoshgoftaar T.M, Hulse J.V, Bullard L (2008) Identifying learners robust to low quality data. In: 2008 IEEE International Conference on Information Reuse and Integration, pp 190–195. https://doi.org/10.1109/IRI.2008.4583028
7. Frénay B, Kabán A, et al (2014) A comprehensive introduction to label noise. In: ESANN
8. Ghosh A, Manwani N, Sastry PS (2017) On the robustness of decision tree learning under label noise. In: Kim J, Shim K, Cao L, Lee JG, Lin X, Moon YS (eds) Advances in knowledge discovery and data mining. Springer International Publishing, Cham, pp 685–697
9. Giunchiglia F, Zeni M, Bignotti E (2018) Personal context recognition via reliable human-machine collaboration. In: Pervasive Computing and Communications Workshops (PerCom Workshops), 2018 IEEE International Conference on, in print
10. Guo B, Yu Z, Zhou X, Zhang D (2014) From participatory sensing to mobile crowd sensing. In: Pervasive Computing and Communications Workshops (PERCOM Workshops), 2014 IEEE International Conference on, pp 593–598
11. Nettleton DF, Orriols-Puig A, Fornells A (2010) A study of the effect of different types of noise on the precision of supervised learning techniques. Artif Intell Rev 33(4):275–306. https://doi.org/10.1007/s10462-010-9156-z
12. Rätsch G, Schölkopf B, Smola AJ, Mika S, Onoda T, Müller KR (2000) Robust ensemble learning for data mining. In: Terano T, Liu H, Chen ALP (eds) Knowledge discovery and data mining. Current issues and new applications. Springer, Berlin, pp 341–344
13. Restuccia F, Ghosh N, Bhattacharjee S, Das SK, Melodia T (2017) Quality of information in mobile crowdsensing: survey and research challenges. ACM Trans Sens Netw (TOSN) 13(4):34
14. Richardson M, Domingos P (2006) Markov logic networks. Mach Learn 62(1):107–136. https://doi.org/10.1007/s10994-006-5833-1
15. Settles B (2009) Active learning literature survey. Computer Sciences Technical Report 1648, University of Wisconsin–Madison. http://axon.cs.byu.edu/~martinez/classes/778/Papers/settles.activelearning.pdf
16. Teso S, Sebastiani R, Passerini A (2017) Structured learning modulo theories. Artif Intell 244:166–187. https://doi.org/10.1016/j.artint.2015.04.002. http://www.sciencedirect.com/science/article/pii/S0004370215000648. (Combining constraint solving with mining and learning)
17. Tourangeau R, Rips LJ, Rasinski K (2000) The psychology of survey response. Cambridge University Press, Cambridge
18. West BT, Sinibaldi J (2013) The quality of paradata: a literature review. Improving surveys with paradata. Wiley Online Library, pp 339–359
19. Zeni M, Zaihrayeu I, Giunchiglia F (2014) Multi-device activity logging. In: Proceedings of the 2014 ACM International Joint Conference on Pervasive and Ubiquitous Computing: Adjunct Publication, pp 299–302
20. Zeni M, Zhang W, Bignotti E, Passerini A, Giunchiglia F (2019) Fixing mislabeling by human annotators leveraging conflict resolution and prior knowledge. Proc ACM Interact Mob Wearable Ubiquitous Technol 3(1):32