Some people aren't worth listening to: periodically retraining classifiers with feedback from a team of end users

Joshua Lockhart [1] **Samuel Assefa** [1] **Tucker Balch** [1] **Manuela Veloso** [1]

Abstract

Document classification is ubiquitous in a business setting, but often the end users of a classifier are engaged in an ongoing *feedback-retrain* loop with the team that maintain it. We consider this feedback-retrain loop from a multi-agent point of view, considering the end users as autonomous agents that provide feedback on the labelled data provided by the classifier. This allows us to examine the effect on the classifier's performance of unreliable end users who provide incorrect feedback. We demonstrate a classifier that can learn which users tend to be unreliable, filtering their feedback out of the loop, thus improving performance in subsequent iterations.

1. Introduction

Financial institutions must process huge amounts of textual data as part of their daily business and operations. When implemented and maintained to a high standard, document classification systems can increase productivity of teams throughout such an organisation, from the front office to the back, from sales through HR, and everywhere in between.

Consider a team of analysts assigned to monitor news articles that may have a bearing on the fate of publicly traded securities. To cope with such a vast stream of information, the analysts may ask a data science team to build a supervised or semi-supervised machine learning classifier to categorize the news articles as they arrive. News articles could be detected as containing information about company earnings releases, announcements about central bank policy changes, or even environmental events that would affect commodities prices.

Consider also a team of support staff who share an email mailbox. If a member of the business needs the support team to do something, they send an email to the shared mailbox. The request is then assigned to one of the support staff to work on. To aid the team in processing the stream of support requests, a classifier could be deployed to categorise each one as it arrives, *e.g.* an email could be labelled 'Tech Failure' if it contains technical details related to I.T. infrastructure, 'Wrong Mailbox' if it is off topic, or even 'Severe' if the email body mentions lost revenue or contains profanity, *etc*.

In both of these cases the business teams have identified the need for a purpose built document classifier. A data science team works with them to train and deploy such a classifier, and it is deployed in a production setting in an effort to reduce workload of the business team.

Unfortunately this is rarely the end of the story. No machine learning classifier will be in production for very long before it provides an incorrect classification of a document. This could be for any number of reasons, including but not limited to out of date or insufficient training data, changing business requirements, or even a change in the format of the data to be classified.

Inevitably the data science team will be contacted by the business team if the classifier stops performing well. The data science team can use this feedback to improve the classifier, perhaps retraining the model and redeploying. It is this *feedback-retrain* loop that we focus on in this paper: we consider feedback given by the end users of a classifier (*agents* in a *feedback pool*), and investigate schemes for using this agent feedback to retrain the classifier.

An immediate issue that arises in this setting is the following: which agents in the feedback pool provide reliable feedback? Is it wise to blindly incorporate all agent feedback, or should we become more selective? It seems reasonable that more experienced agents in the feedback pool will provide better feedback than less experienced agents. For instance, perhaps a subset of the agents in the pool confuse two categories, and consistently swap one for the other when asked for feedback on a particular labelling of a document. Retraining the classifier on documents with labels provided by such an agent will cause issues with performance.

[1]JP Morgan Artificial Intelligence Research. Correspondence to: Joshua Lockhart <joshua.lockhart@jpmchase.com>.

Proceedings of the 36^{th} International Conference on Machine Learning, Workshop on AI in Finance: Applications and Infrastructure for Multi-Agent Learning, Long Beach, CA, 2019.

Explicitly, we consider how a classifier's accuracy can be improved over time by engaging in a feedback loop with a pool of agents. The agents receive labelled documents from the classifier. If an agent thinks their document has been mis-labelled, they provide the classifier with feedback: an alternate label that they believe is correct. These 're-labelled' documents are collected and used to retrain the classifier, in the hope that this new iteration will provide better performance than the current one.

The goal of the classifier is to identify and filter out agents whose feedback degrades rather than improves performance. Note that our motivating scenarios are those in which each discrete document is sent to one agent to work on. Hence, in our setting each document will receive one piece of feedback: a single alternative label provided by an agent in the pool. This complicates matters because the classifier is unable to consolidate multiple pieces of feedback on a single data point.

1.1. Related work

(Hovy et al., 2013) consider how trustworthy users can be identified from non-expert annotation services, *e.g.* Amazon Turk. They identify what we would refer to as noisy agents by comparing multiple user's responses to a particular task, which is in contrast to the problem we are trying to solve where at most one user will give feedback on a data point they receive. (Vaughan, 2017) provides a comprehensive survey on how machine learning models can be trained from crowdsourced labelled data. Concrete examples of work on the issues that arise in this setting are (Dalvi et al., 2013), who consider how multiple inputs from users can be aggregated to learn the ground truth of a binary classification task. (Zhang et al., 2014), (Zhu et al., 2015) and (Sinha et al., 2018) all build on the seminal Dawid-Skene algorithm (Dawid & Skene, 1979) for aggregating multiple, often differing opinions on a data point into a classification close to some ground truth.

This existing literature tends to come from the perspective of *obtaining* labels for data that is intended to be used as training set for a classifier. The problem we consider is an online version of this problem, where instead of relying on a crowd of unknown participants (*c.f. Amazon Turk*) to give us some semblance of a ground truth, we are interested in taking labels from a team of end users with a vested interest in the classifier performing well. In this setting it is appropriate to keep track of the performance of individual users, with the aim of giving them direct feedback on how helpful their feedback is for the classifier. Furthermore, we are interested in the case where there is only one piece of feedback on a labelled item, from a single end user, meaning we are not able to make use of any sort of majority vote based algorithm.

2. Approach

2.1. Training and feedback loop

We consider an iterative learning scenario over a dataset X, Y partitioned into N equal chunks $(X^{(1)}, Y^{(1)}), \ldots, (X^{(N)}, Y^{(N)})$. Each data point $x \in X$ has a *true label* y that belongs to the fixed set of target labels $\mathcal{L}$. There is assumed to be a fixed pool of *agents* $A = \{A_1, \ldots, A_M\}$. Upon receiving a label $l \in \mathcal{L}$, an agent $A_i \in A$ will respond with another label $l' \in \mathcal{L}$, which we refer to as that agent's *feedback*. Note that for the purposes of our simulation, agents simply act as functions $A_i : \mathcal{L} \to \mathcal{L}$. To elicit 'feedback' on a classification, we as the experimenter will simply pass the agent the ground truth for the data point under consideration and use their response to retrain the classifier. As we will define in Section 2.2, agents can be reliable (act as identity function), noisy (respond with random selection from $\mathcal{L}$), or confused (act as non-identity function on $\mathcal{L}$).

The iterative learning scenario for a classifier C_θ over a partitioned data set $(X^{(1)}, Y^{(1)}), \ldots, (X^{(N)}, Y^{(N)})$ with agent feedback pool $A = \{A_1, \ldots, A_M\}$ is as follows

1. Receive $X^{(i)} = x_1^{(i)}, \ldots, x_k^{(i)}$. Provide labels $L = \{C(x_1^{(i)}), \ldots, C(x_k^{(i)})\}$.
2. Receive agent feedback $A(X^{(i)}) := \left(j_1, A_{j_1}\left(y_1^{(i)}\right)\right), \ldots, \left(j_k, A_{j_k}\left(y_k^{(i)}\right)\right)$, where each j_u is selected uniformly at random from $\{1, \ldots, M\}$.
3. Based on agent feedback $A(X^{(i)})$ and classifier labels L, adjust classifier parameters θ.
4. Set $i = i + 1$. Go to step 1.

To elucidate Step 2: for each datapoint x_u we randomly select a user A_{j_u}, recording the index of the user j_u and their response to the ground truth labelling of that datapoint $A_{j_u}(x_u)$. This is the information the training scheme will use to determine if each piece of feedback is worth using in subsequent training stages.

At each iteration, the classifier's accuracy is measured by comparing the classifier's labels $x_1^{(i)}, \ldots, x_k^{(i)}$ with the ground truth labels $y_1^{(i)}, \ldots, y_k^{(i)}$ by means of the F_1 metric with micro averaging (F1M).

Note that the classifier taking part in this learning scenario receives only the data points and the feedback provided by the pool of agents. The ground truth labels Y are provided to the agents only. This allows us to allow us to gain control over the 'personality' of an agent by treating them as functions $A : \mathcal{L} \to \mathcal{L}$. Consider for example a reliable

Algorithm 1 UPDATETRUSTTABLE

Input: Labels $l_1, \ldots, l_k$, a list of agent IDs and their feedback labels $\{(j_1, l'_1), \ldots, (j_k, l'_k)\}$
for $i = 1$ **to** k **do**
  if $l_i = l'_i$ **then**
    Update agent trust score $\mathcal{T}(j_i)$: *the agent has provided another label, and we agree on it*
  else
    Update agent trust score $\mathcal{T}(j_i)$: *the agent has provided another label, but we disagree with it*
  end if
end for

agent that always provides the ground truth. This can be modelled as the identity function. At the other extreme, an agent that acts as permutation on $\mathcal{L}$ with no fixed points will be completely useless as a source of training examples: it always gives the wrong class labels.

The main contribution of this work is a means of identifying unreliable agents during the training of the classifier, then disregarding the feedback. We demonstrate an algorithm, PRUNENB that acts as a filter over the feedback provided to the classifier at each iteration. We demonstrate that unreliable agents can be identified from their behaviour in previous iterations and filtered out from the feedback loop. We call such algorithms *training schemes*.

2.2. Modelling agent feedback

In our experiments the pool of agents that give feedback to the classifier at each step will consist of agents of the following kinds:

- *Reliable Agents*: respond with the true label for any data point they receive; explicitly $A(l) = l$ for all $l \in \mathcal{L}$.
- *Noisy Agents*: respond with a label selected uniformly at random from the set of target labels; explicitly $A(l) = l'$ where $l' \in_R \mathcal{L}$, for all $l \in \mathcal{L}$.
- *Confused Agents*: always respond according to some fixed non-identity function $A : \mathcal{L} \to \mathcal{L}$. Such agents are deterministic, but consistently incorrect about all or some of the classes in $\mathcal{L}$.

2.3. Algorithms

The general framework of our algorithms is laid out in Algorithm 2. For each agent in the feedback pool, the algorithm maintains a *trust score*. Explicitly, an agent's trust score is the quantity

$$\mathcal{T}(i) := \frac{\#\text{times my labels have agreed with agent } A_i\text{'s labels}}{\#\text{times agent } A_i \text{ has provided a label}}$$

Algorithm 2 PRUNE

Input: Stream of N chunks of datapoints, $X^{(1)}, \ldots, X^{(N)}$, access to feedback agent pool $A = \{A_1, \ldots, A_M\}$.
for $i = 1$ **to** N **do**
  Get labels from classifier $L := l_1, \ldots, l_k = C(x_1^{(i)}), \ldots, C(x_k^{(i)})$
  Get feedback from agent pool $A(X^{(i)}) = (j_1, l'_1), \ldots, (j_k, l'_k)$.
  Call UPDATETRUSTTABLE with arguments L and $A(X^{(i)})$.
  $R \leftarrow [\,]$
  for $u = 1$ **to** k **do**
    if $\mathcal{T}(j_u)$ satisfies $\langle$*TrustCondition*$\rangle$ **then**
      Append $(x_u^{(i)}, l'_u)$ to list R.
    end if
  end for
  Continue training classifier with labelled data points in R.
end for

which is updated at each iteration of the feedback loop. In Algorithm 2 we refer to a generic $\langle$*TrustCondition*$\rangle$, which can be one of two conditions

- MEANPRUNE: if $\mathcal{T}(i) > \sum_{j=1}^{M} \mathcal{T}(A_j)/M$ then accept, otherwise reject.
- THRESHOLDPRUNE: for some fixed threshold c, if $\mathcal{T}(i) > c$ then accept, otherwise reject.

In the next section we outline the experiments we perform to investigate the performance of these training schemes.

2.4. Experimental methodology

We call the following an N-stage *trial* of a classifier C on a set of data points X with ground truth labels Y, under feedback from a pool of agents A.

1. Shuffle X, Y. Partition X, Y into N chunks $(X^{(1)}, Y^{(1)}), \ldots, (X^{(N)}, Y^{(N)})$.
2. Pre-train classifier C on $X^{(1)}$ and $Y^{(1)}$.
3. Run PRUNE implementation on $X^{(2)}, \ldots, X^{(N)}$ with classifier C and feedback pool A. Record F_1 score at each iteration i of the outer loop by comparing the classifier's labels L with the true labels $Y^{(i)}$.

The classifier we use as C is a Multinomial Naive Bayes classifier. The partial_fit() function provided in the Scikit-Learn (Pedregosa et al., 2011) implementation of this classifier is used to continue training at each stage of the trial.

We henceforth refer to our training schemes as MEANPRUNENB and THRESHOLDPRUNENB respectively.

To judge the performance of MEANPRUNENB and THRESHOLDPRUNENB we run 100 repetitions of a 10-stage trial as described above. We are interested in the F_1 score at each stage of the feedback-retraining loop, so we save each 10-vector of F_1 scores after each trial $\vec{t_1}, \ldots, \vec{t_{100}}$, and take the mean over the 100 repetitions $\sum_{i=1}^{100} t_i/100$ to obtain an estimate of the performance at each stage.

2.4.1. DATASETS

We run our experiments on two datasets. The first dataset is the 20 Newsgroups Dataset (20N), obtained in a 'pre-vectorized' form using the Scikit-learn library (Pedregosa et al., 2011). We select all emails from the 'sci.med','comp.graphics', 'talk.politics.mideast', and 'sci.space' mailing lists, giving us a dataset with $\{594, 584, 564, 593\}$ data points in the respective classes. The classification task we consider is to take an email from this set, and label it with the newsgroup it is from.

The second data set is taken from the Reuters news document data set (reu). Specifically, we use the 90 class subset of the Reuters-21578 data set distributed as part of the nltk library (Bird et al.). We select the news articles that belong to the categories $\mathcal{C} = \{$'earn', 'acq', 'money-fx', 'grain', 'crude', 'trade', 'interest', 'sugar', 'corn', 'ship'$\}$. Articles in the Reuters dataset can belong to more than one category. To simplify matters we will not consider the problem of multi-label classification, so we remove all articles in this dataset that belong to more than one category in $\mathcal{C}$ (*i.e.* we filter out articles that aren't labelled with at least one category in $\mathcal{C}$. Then we filter out articles in that reduced set that belong to more than one category in $\mathcal{C}$). This leaves us with a dataset with $\{2847, 1609, 369, 217, 315, 318, 203, 104, 2, 119\}$ datapoints in each respective category. This dataset is then vectorised using the TfidfVectorizer class in Scikit-learn.

The next section contains details on the benchmarks we use to contextualise the performance of the training schemes we consider.

2.4.2. TRUSTING AND DISCERNING CLASSIFIERS

As a benchmark we consider two training schemes that represent different extremes: complete trust of all agents; and maximal discernment as to the nature of each agent.

At each round i, the TRUSTINGNB scheme accepts all feedback it receives from the users as truthful and useful, continuing the training of the classifier with the datapoints $x_1^{(i)}, \ldots, x_k^{(i)}$ and feedback labels it received from the agents $l'_1, \ldots, l'_k$. This is the lower bound benchmark that any training scheme should beat.

Conversely, the DISCERNINGNB scheme is imbued with perfect knowledge of the nature of each agent in the pool (reliable, noisy, or confused). This scheme will only train on feedback from agents it knows to be reliable (recall the reliable agents always give the ground truth labelling). Thus, the DISCERNINGNB scheme represents the best possible way in our setting of training the classifier at each stage of feedback: giving it the most possible ground truth available at each stage.

3. Experiments

3.1. Learning in the presence of noisy agents

First, we consider the case where the feedback pool is made up of reliable agents and noisy agents. In Figures 1 and 2 we compare the performance of TRUSTINGNB, DISCERNINGNB, and our algorithm THRESHOLDPRUNENB on a pool of five agents, with different proportions of noisy users.

During the research we found that performance improved if we made the PRUNENB schemes trust all agents until after stage 3 of the feedback-retrain loop. We call this the 'burn-in' period of the scheme. During the burn-in period we do no filtering of agents, and the scheme acts like TRUSTINGNB. After the burn-in period is over, the scheme filters agents according to its $\langle$*TrustCondition*$\rangle$.

The classifiers were tested simultaneously, each receiving the same chunk of data and agent feedback at each stage of the training. For each experiment we provide two plots, a lower plot which shows a line with error bars and an upper plot which shows data points marked with an 'x'. The shared X axis of the upper and lower plots contains the stage i of the feedback loop. The Y axis is the F_1 score at each stage. The lower plot contains the mean of this number over the 100 repetitions of the trial, with standard error bars. The upper plot attempts to provide a visualisation of the distribution of this number for each of the three classifiers at each stage. Each 'x' in the upper plot corresponds to the F_1 score obtained at each stage of one of the 100 trials. Each X position in the upper plot is split into three, one for each classifier. From left to right these are the DISCERNINGNB, THRESHOLDPRUNENB, and TRUSTINGNB classifiers. The THRESHOLDPRUNENB classifier was run with trust score threshold of 0.3 and a burn-in of 3 steps.

3.2. Learning in the presence of confused agents

Recall that a confused agent is one that answers according to some non-identity function on the labels representing a mistaken belief. We now consider the case where the feedback pool is made up of four reliable agents and a single confused agent. In the experiment on the Newsgroups dataset, the confused agent maps the labels 'sci.med' and 'comp.graphics' to 'comp.graphics', and the labels 'talk.politics.mideast' and

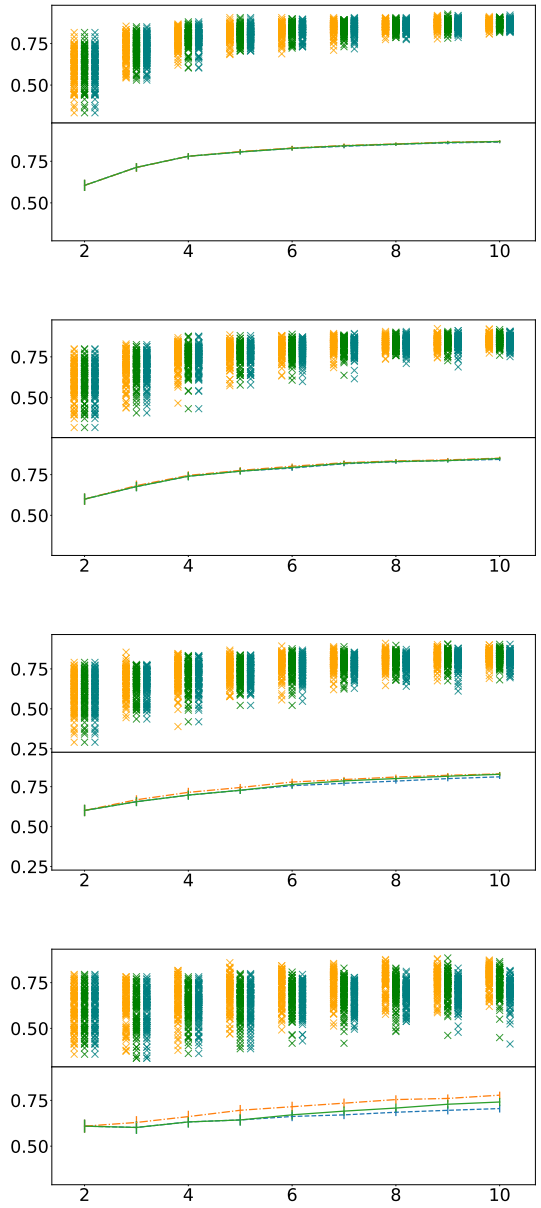

Figure 1. Performance of DISCERNINGNB (orange, left/dot+dash), THRESHOLDPRUNINGNB (green, middle/solid), and TRUSTINGNB (teal, right/dash) on 'drip fed' data from Newsgroups with noisy feedback from a pool of five agents. Error bars are standard error, sample size 100. X axis denotes the stage of the trial (i), Y axis is F_1 score of classifier at that stage. From top to bottom: four reliable agents, one noisy agent; three reliable agents, two noisy agents; two reliable agents, three noisy agents; one reliable agents, four noisy agents.

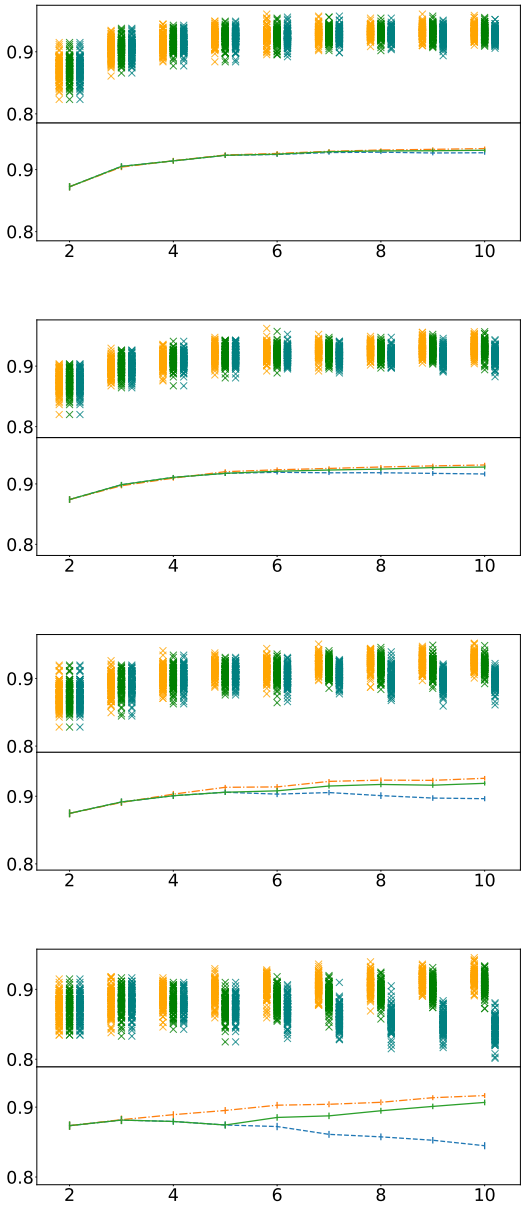

Figure 2. Performance of DISCERNINGNB (orange, left/dot+dash), THRESHOLDPRUNINGNB (green, middle/solid), and TRUSTINGNB (teal, right/dash) on 'drip fed' data from Reuters data set with noisy feedback from a pool of five agents. Error bars are standard error, sample size 100. X axis denotes the stage of the trial (i), Y axis is F_1 score of classifier at that stage. From top to bottom: four reliable agents, one noisy agent; three reliable agents, two noisy agents; two reliable agents, three noisy agents; one reliable agent, four noisy agents.

'sci.space' to 'talk.politics.mideast'. In the experiment on the Reuters dataset, the agent will map the labels 'earn', 'acq', 'money-fx', 'trade', and 'interest' to the single label 'money-fx', and all other labels to 'crude'.

We compare the performance on Newsgroups and Reuters datasets of TRUSTINGNB, DISCERNINGNB, and our algorithm MEANPRUNENB on a pool of four reliable agents and the confused agents described above. We plot the results for the Newsgroups dataset in Figure 3, and the results for the Reuters dataset in 4.

Again, schemes were tested simultaneously, each receiving the same chunk of data and agent feedback labels at each stage of the training. There was no burn in period used for these experiments. The lower plot shows the mean and standard error of the F_1 score at each stage of the trial, the upper plot shows a plot of the distribution F_1 scores at each stage for each classifier.

4. Discussion of results

4.1. Effectiveness in the presence of noisy agents

Figures 1 and 2 concern the performance of the THRESHOLDPRUNENB classifier in the presence of noisy agents. Remarkably, TRUSTINGNB is relatively resilient to noisy feedback from a small subset of the feedback pool, consider in particular the performance illustrated in the top plot of Figure 1: a single noisy agent has barely any effect on the F_1 score, almost matching the performance of the DISCERNINGNB.

The impact of noise becomes more apparent in the case where the majority of agents are providing noisy labels to the classifier, see the case where we have a pool consisting of a single reliable agent and four noisy agents. The impact of noise is most apparent in the Reuters dataset, illustrated in bottom plot of Figure 2, where we see the TRUSTINGNB lower benchmark classifier substantially underperform the DISCERNINGNB upper benchmark classifier.

Notice the overall performance impact of noisy agent feedback which manifests itself in a wider spread of F_1 values. See the widening of the F_1 distributions in all stages in the bottom two plots in both Figures 1: while the mean stays relatively similar, the F_1 distribution is noticeably wider, meaning that occasionally the classifier performs much worse than usual. We conjecture that this is due to the fact that even in the upper benchmark classifier DISCERNINGNB case, the classifier trains on substantially less training data, but the F_1 score is tested on the full chunk at each stage.

The performance increase afforded by the THRESHOLDPRUNENB scheme over the lower benchmark classifier TRUSTINGNB is very slight in the Newsgroups dataset. The effect manifests itself in the slight increase in stage 8 of the bottom plot in Figure 1, but the TRUSTINGNB value falls within the standard error bar of the THRESHOLDPRUNENB mean. Nevertheless, examining the F_1 distribution at stages 7 and 8 we see that the weight of the THRESHOLDPRUNENB distribution (middle, green) seems to fall slightly higher than the TRUSTINGNB distribution (right, teal). The effect is very slight.

More convincing evidence of the performance increase offered by the THRESHOLDPRUNENB scheme is illustrated in the Reuters dataset. Here we see an unambiguous boost in performance in the one reliable, four noisy agent trial (bottom plot, Figure 2) in stages 6, 7 and 8.

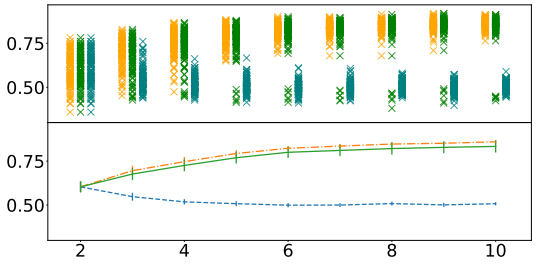

Figure 3. Performance of DISCERNINGNB (orange, left/dot+dash), THRESHOLDPRUNINGNB (green, middle/solid), and TRUSTINGNB (teal, right/dash) on 'drip fed' data from the Newsgroups dataset, with a pool of four reliable agents and a single confused agent. X axis denotes the stage of the trial (i), Y axis is F_1 score of classifier at that stage. Error bars are standard error, sample size 100.

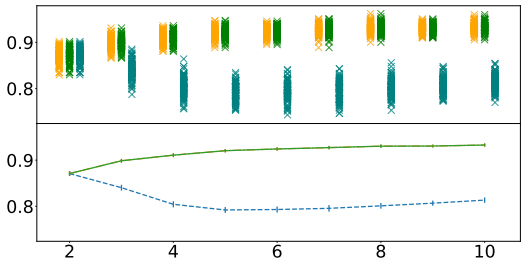

Figure 4. Performance of DISCERNINGNB (orange, left/dot+dash), THRESHOLDPRUNINGNB (green, middle/solid), and TRUSTINGNB (teal, right/dash) on 'drip fed' data from the Reuters dataset, with a pool of four reliable agents and a single confused agent. X axis denotes the stage of the trial (i), Y axis is F_1 score of classifier at that stage. Error bars are standard error, sample size 100.

4.2. Effectiveness in the presence of confused agents

Figures 3 and 4 illustrate how the three schemes perform on a feedback pool with four reliable users and a confused

agent. Unlike a single noisy agent, a single confused agent causes a substantial performance degradation in the TRUSTINGNB benchmark, so much so that we don't consider more than one.

In both datasets we see that our algorithm immediately starts to outperform the lower benchmark TRUSTINGNB. For the Reuters dataset it reaches the same level of performance as the upper benchmark, DISCERNINGNB. This level of performance is not reached on the Newsgroups dataset. From examining the execution traces of our experiments, we conjecture that this is because in some repetitions of the trial the trust score of some reliable agents can very occasionally dip below the mean, due to errors made by the classifier itself in the early stages of the training. The errors made by the classifier cause it to disagree with the reliable agents, which means they will be taken to have low trust scores. This can compound, and cause the classifier to erroneously distrust one or more of the reliable classifiers throughout all of the subsequent feedback-retrain stages.

Indeed the MEANPRUNENB very occasionally performs worse than the TRUSTINGNB scheme. This manifests itself in the wide spread in the F_1 distribution in the later stages, an effect most apparent in Figure 3. In stage 8 of this figure we note that the distribution has become bimodal. Subsequent work should address this issue, which we feel has arisen due to the 'misplaced trust' issue we discuss above.

5. Conclusion

We have considered a problem that faces any data science team that deploys a classifier in a production setting: just how do we retrain it when it (inevitably) provides the *users* of that classifier with labels they disagree with? The natural way of achieving this is to ask the users themselves to provide feedback on the outputs of the classifier. Either they agree with the labelling, in which case they send nothing back and the classifier takes this as an implicit confirmation of that label, or they disagree and choose the label from the list they consider more appropriate. An inevitable issue that will arise is that not all of the end users of a classifier will provide meaningful feedback: perhaps the user is disengaged with the process, misinterpets the meaning of some of the target labels, or even misunderstands what the feedback mechanism means. We have attempted to model these feedback scenarios in a multi-agent learning setting: in our imaginary scenario we have posited that there is a team of users (agents) who must process a set of documents according to some business need. These documents are labelled by the classifier, and distributed amongst the agents to work on. If an agent wants to provide feedback on one of the documents they have been given, they give this to the classifier. These feedback 'relabellings' are collected and used to periodically retrain the classifier. This multi-agent setting provides a compelling new consideration not present in the crowdsourced training data literature: if an agent keeps making mistakes then we'd like to identify them. Perhaps that agent is a new hire and thinks that one label means something else. A system that could identify their confusion could flag this, and the agent could be gently provided with documentation on what the labels mean. Alternatively, agents who provide the system with random feedback labels when they feel like it could be identified and suitable action could be taken.

We have presented a scheme for training a classifier in this iterative setting, the PRUNE wrapper around the Multinomial Naive Bayes classifier. We have shown how agents in the feedback pool can be assigned a *trust score*, that can be used to segment agents into those who give feedback we want to listen to (high trust score agents) and those whose feedback we have grown to distrust (low trust score agents). We have considered two different ways of segmenting users based on these scores: an explicit threshold cut off (THRESHOLDPRUNENB) and an arithmetic mean cut off (MEANPRUNENB). We have demonstrated that these two schemes can cope with noisy users, and confused users respectively.

Acknowledgements

J.L. thanks Andy Alexander, Ayham Alajdad, Angelos Filos, Mahmoud Mahfouz for their ongoing input to this work, and Greg Sidier for helpful discussions.

Disclaimer

used in any way for evaluating the merits of participating in any transaction. Past performance is not indicative of future results. Please consult your own advisors regarding legal, tax, accounting or any other aspects including suitability implications for your particular circumstances. J.P. Morgan disclaims any responsibility or liability whatsoever for the quality, accuracy or completeness of the information herein, and for any reliance on, or use of this material in any way.

References

The 20 newsgroups text dataset. https://scikit-learn.org/0.19/datasets/twenty_newsgroups.html. Accessed: 2019-04-29.

sklearn.metrics.f1_score. https://scikit-learn.org/stable/modules/generated/sklearn.metrics.f1_score.html.

Reuters-21578 text categorization collection data set. https://archive.ics.uci.edu/ml/datasets/reuters-21578+text+categorization+collection. Accessed: 2019-04-29.

Bird, S., Klein, E., and Loper, E. Natural language processing with python analyzing text with the natural language toolkit. ch. 2. accessing text corpora and lexical resources. https://www.nltk.org/book/ch02.html. Accessed 2019-04-29.

Dalvi, N., Dasgupta, A., Kumar, R., and Rastogi, V. Aggregating crowdsourced binary ratings. In *Proceedings of the 22nd international conference on World Wide Web*, pp. 285–294. ACM, 2013.

Dawid, A. P. and Skene, A. M. Maximum likelihood estimation of observer error-rates using the em algorithm. *Journal of the Royal Statistical Society: Series C (Applied Statistics)*, 28(1):20–28, 1979.

Hovy, D., Berg-Kirkpatrick, T., Vaswani, A., and Hovy, E. Learning whom to trust with mace. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pp. 1120–1130, 2013.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., and Duchesnay, E. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

Sinha, V. B., Rao, S., and Balasubramanian, V. N. Fast dawid-skene: A fast vote aggregation scheme for sentiment classification. *arXiv preprint arXiv:1803.02781*, 2018.

Vaughan, J. W. Making better use of the crowd: How crowdsourcing can advance machine learning research. *Journal of Machine Learning Research*, 18:193–1, 2017.

Zhang, Y., Chen, X., Zhou, D., and Jordan, M. I. Spectral methods meet em: A provably optimal algorithm for crowdsourcing. In *Advances in neural information processing systems*, pp. 1260–1268, 2014.

Zhu, C., Xu, H., and Yan, S. Online crowdsourcing. *CoRR*, abs/1512.02393, 2015.