ClusterClean: a Weak Semi-Supervised Approach for Cleaning Data Labels

Kyriaki Dimitriadou
Amazon
kikid@amazon.com

Rahul Manghwani
Amazon
rmanghwa@amazon.com

Timothy C. Hoad
Amazon
hoad@amazon.com

***Abstract*—Every day, tens of thousands of Amazon customers message Amazon's selling partners inquiring about orders, products and services. Unfortunately, we have observed over time that buyers may send unsolicited and often times mal intended messages to sellers through the buyer-seller messaging (BSM) service. Although the BSM service gives the ability to sellers to report such messages, most of them do not make use of this feature. Hence, collecting training and testing data with clean labels to build machine learning models in order to proactively block unsolicited messages and help prevent and mitigate losses for Amazon is extremely challenging. To address this problem we propose ClusterClean, an algorithm that automatically cleans data labels with little to no human effort. ClusterClean has the ability to a) accurately infer the labels of unlabeled data points based on an initial small labeled set and b) detect new data patterns such as incoming unobserved spam attacks by soliciting feedback from users to decide on their label. Experiments on approximately 150,000 real messages from Amazon customers showed that ClusterClean can accurately clean the labels of these messages in a few minutes, drastically reducing the human effort and time spent on this task.**

I. Introduction

In the last six months millions of messages have been sent from buyers to sellers in the US. Based on our investigations, we observed many different types of unsolicited messages coming through the service, varying from messages with malware attachments to phishing messages asking sellers for their credit card details. Figure 1 shows examples of real messages sent to sellers in January 2019. In order to help shield sellers from such messages and protect Amazon, we built a proactive automated system based on machine learning models that blocks malicious messages and prevents such attacks from happening before they begin. By blocking malicious messages, we improve the sellers' experience and help them focus on their communication with legitimate buyers who are contacting them about their orders, products or services.

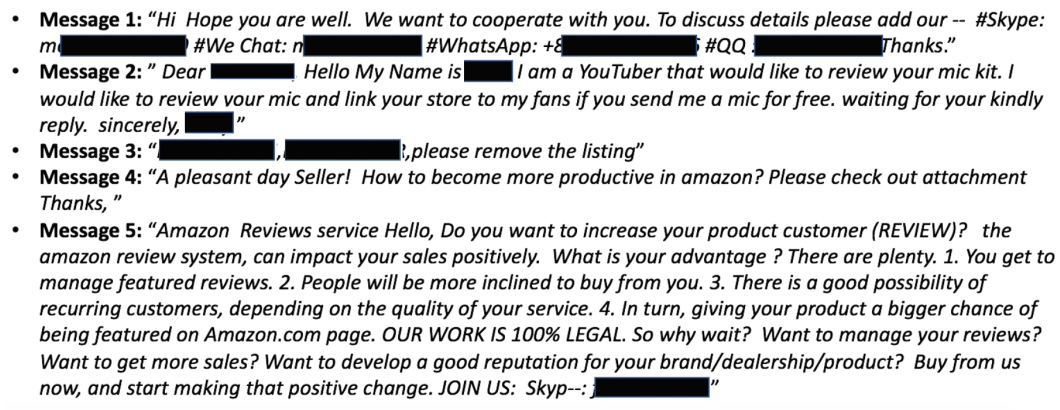
The greatest challenge we faced during the development of ML models that identify malicious messages was gathering a large amount of training data with clean labels. Currently, when a seller receives a message she can report it in one of several abuse categories: spam, fraud, phishing, inappropriate content, asking for free products, taking communication outside of Amazon and other. However, most of the abusive messages do not get reported resulting in a large number of false negatives. Even when sellers went to the trouble of reporting a message, most of the times they reported it in the wrong category (e.g. reporting a spam message as fraud). Furthermore, to top it off, we observed a few cases where sellers reported messages that were actually legitimate. Therefore, using the reported messages as positive and non-reported ones as negative to form our training set in order to build a model was not optimal because of how noisy the labels provided by the sellers were.

To address this problem and gather a labeled set for training, we initially used a manual-based approach. We gathered a team of 10 people and asked them to read messages and label them as abusive or non-abusive. By using human annotators, we were able to collect approximately 50K labeled messages. Not only was this process labor intensive and time consuming but it also resulted in inconsistencies between the labeled messages (for example one annotator could label an out of the office message as abusive and another one as non-abusive).

As the next step, in order to increase the amount of labeled data for our models, we followed a heuristic-based approach. Initially, a human observed different patterns in the messages. Then she designed a heuristic based on those patterns in order to clean and give a label to the unlabeled data such as: label all messages that contain the phrase *"remove this listing"* as abusive. Although this process helped us obtain a large number of labels, there are several drawbacks associated with it: 1) the process is completely subjective to the human that is designing the heuristics: two humans might come up with completely different heuristics to clean the data, 2) it is labor intensive since the human must observe many messages in order to design the heuristics and 3) there might be abusive or non-abusive patterns that a human might completely miss.

The problem of obtaining clean data labels is not unique to this domain or user case and is increasingly becoming one of the biggest pain points in building machine learning models [1], [2]. Inspired by our heuristic-based approach, we designed *ClusterClean*, an algorithm that automates this ad-hoc process and solves the problem of obtaining large amounts of data labels quickly with only a small effort from the users. Given a small training set of clean data labels, ClusterClean works in two steps: a) it automatically finds patterns in the data and b) gives a label to each unlabeled point based on the pattern it corresponds to, requiring minimal labeling effort from the user.

To the best of our knowledge ClusterClean is the first approach that combines aspects of weak [3] and semi-supervised



- **Message 1:** "*Hi Hope you are well. We want to cooperate with you. To discuss details please add our -- #Skype: m[redacted] #We Chat: n[redacted] #WhatsApp: +8[redacted] #QQ [redacted]Thanks.*"
- **Message 2:** "*Dear [redacted] Hello My Name is [redacted] I am a YouTuber that would like to review your mic kit. I would like to review vour mic and link your store to my fans if you send me a mic for free. waiting for your kindly reply. sincerely, [redacted]*"
- **Message 3:** "*[redacted],please remove the listing*"
- **Message 4:** "*A pleasant day Seller! How to become more productive in amazon? Please check out attachment Thanks, *"
- **Message 5:** "*Amazon Reviews service Hello, Do you want to increase your product customer (REVIEW)? the amazon review system, can impact your sales positively. What is your advantage ? There are plenty. 1. You get to manage featured reviews. 2. People will be more inclined to buy from you. 3. There is a good possibility of recurring customers, depending on the quality of your service. 4. In turn, giving your product a bigger chance of being featured on Amazon.com page. OUR WORK IS 100% LEGAL. So why wait? Want to manage your reviews? Want to get more sales? Want to develop a good reputation for your brand/dealership/product? Buy from us now, and start making that positive change. JOIN US: Skyp--: [redacted]*"

Fig. 1: Sample of unsolicited messages sent through the BSM service in January 2019.[1]

learning [4]. It discovers patterns in the data and labels them based on their proximity to the labeled points (similar to semi-supervised algorithms) but it also solicits feedback from users when the labeled points are dissimilar to new patterns that we observe in the data (similar to weakly supervised approaches). Specifically, with ClusterClean, we provide a method which is:

- **Effective** in detecting and labeling new unobserved patterns such as new incoming spam attacks, by asking for the users feedback on the label of sample data points corresponding to the new patterns.
- **Efficient** in labeling large amounts of data in short time given a small initial labeled set of data. Our experiments showed that ClusterClean is able to clean the labels of approximately 150K messages with a high precision and recall in under 2.5 minutes.
- **Extensible** to other domains beyond Buyer Seller Messaging where it is expensive to retrieve a large number of data labels manually and a small initial set of clean labels is available. Furthermore, ClusterClean can be applied to multi-label classification and not just binary problems.

The rest of this paper is organized as follows: we discuss ClusterClean in detail in Section II, we present our experiments in Section III, give a brief overview of related work in Section IV and finally conclude in Section V.

II. Proposed method: ClusterClean

In this section we present our algorithm. ClusterClean is motivated by and tries to automate the heuristic-based approach described above. Initially, it discovers patterns in the data space using a clustering algorithm. Conceptually, we assume that a pattern in the data corresponds to a specific label or class. Once ClusterClean has clustered the data, it labels all unlabeled points within a cluster either based on their proximity to some already existing labeled points or by asking for the user's feedback. The complete ClusterClean algorithm is presented in figure 3. Next, we describe it in detail.

[1]We have redacted the messages to protect the identity of the message senders and receivers.

The algorithm accepts as input a small set of labeled data points $S_l = \{l_1, l_2, ..., l_x\}$ as well as a set of unlabeled points $S_u = \{u_1, u_2, ..., u_n\}$ that it will try to give a label to. Every data point s consists of a set of features $F = \{f_1, f_2, ..., f_v\}$ and a label y; for a point $s \in S_u$ the correct label s_y is unknown. At the first step, ClusterClean discovers patterns using both the labeled and unlabeled data. In order to discover these patterns, it uses a clustering algorithm, specifically k-means, and clusters all available data points. For the purposes of this clustering we use all features F for the data except for their label (if a label exists).

ClusterClean is not aware of how many patterns lie in the data and thus what is the number of clusters k that should be used for clustering. For this reason, it follows an iterative approach. Initially, it starts with a minimum k_{min} and increases it in every iteration until one of two stop conditions are true: a) a purity threshold P is met or b) the threshold of the user effort E is met. We use the purity threshold as a stopping criterion since it is a measure of how well the different classes have been distributed in every cluster; ideally every class of data will be placed in its own cluster. On the other hand, we use the user effort threshold to counter balance the effect of creating too many clusters that the user will have to manually label. We discuss these stopping conditions and the trade-off between them in more detail in the next section.

When one or both of these two stopping conditions have been met, we compute the *majority label* within every cluster based on the labeled points and label every unlabeled data point within that cluster with the majority label. The majority label is the class to which most data points belong to within the cluster. In the case of a tie between two or more classes for the majority label, we randomly pick one of the majority classes and assign the unlabeled points in the cluster to that class. Figure 2 shows an example in a two dimensional space containing two classes of data points. As we see in this example, cluster $C1$ contains more positive data points than negative and therefore we will label every unlabeled data point in $C1$ with a positive label. Similarly, every unlabeled point in $C2$ will be given a negative label.

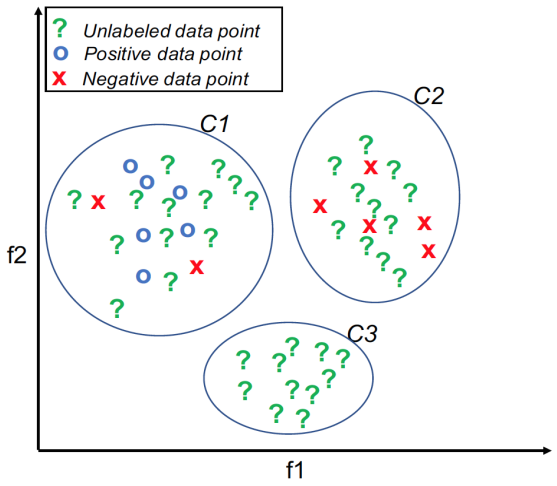


Fig. 2: Cluster assignments using ClusterClean.

When clustering the labeled and unlabeled points together, there can be cases where clusters that have been formed in the data space do not contain any labeled points; this is the case for cluster $C3$ in figure 2. Such cases can occur when new patterns that are observed in the unlabeled data did not exist in the initial set of labeled data provided to ClusterClean. To decide on the label of such new patterns, we pick a random sample point from within the cluster and ask the user to provide us with a label for that point. Based on the label given as feedback, ClusterClean labels every point in the cluster. Once ClusterClean has iterated through all clusters in the dataspace and given a label to all points in every cluster it finally comes to an end.

A. Stopping conditions

As stopping criteria for ClusterClean we use the purity p of the clusters as well as the user effort e measured by the number of labels a user is willing to manually provide. These conditions complement each other. We use purity as a measure of the clustering performance and use the user effort to make sure that we do not create too many clusters that the user will have to manually provide labels for. Next, we give the formal definitions for each of these conditions and we discuss how one affects the other.

Purity Intuitively, purity is a measure of the homogeneity of clusters; the more points that belong to one class within a cluster, the higher the purity value and the better our clustering is. Formally, purity is defined as:

$$p = \frac{1}{x}\sum_{i=1}^{k} max_j|c_i \cap t_j| \tag{1}$$

where x is the total number of labeled points we have in our initial labeled set, k is the number of clusters, c_i is cluster i and t_j is class j. Obviously, $p = 1$ when our labeled points have been ideally distributed in the clusters and every cluster contains points that belong only to one class (or in the degenerate case when every point is its own cluster).

Algorithm 1: ClusterClean

Input: $S_l = \{l_1, l_2, ..., l_x\}$ (set of labeled points)
$S_u = \{u_1, u_2, ..., u_n\}$ (set of unlabeled points)
$T = \{t_1, t_2, ..., t_j\}$ (set of classes)
E (user effort threshold)
P (purity threshold)
k_{min} (minimum number of clusters)
Set $k = k_{min}$
while $p < P$ & $e < E$ **do**
  $C^* \leftarrow$ k-means($S_l \cup S_u$, k)
  $p = \frac{1}{x}\sum_{i=1}^{k} max_j|c_i \cap t_j|$
  $e = \sum_{i=1}^{k} \mathbb{1}(|c_{i,u}{}^{**}| == |c_i|)$
  $k+ = 1$
end
for c_i *in* C **do**
  $label_i = argmax_j|c_i^j|$
  if $label_i == null$ **then**
    u_r = randomly pick a point in c_i
    $label_i$ = get user feedback on u_r
  end
  for u *in* $c_{i,u}$ **do**
    $u_y = label_i$
  end
end
* *where* $C = \{c_{k_{min}}, c_{k_{min}+1}, ..., c_k\}$ *is a set of clusters.*
** *where* $|c_{i,u}|$ *is the number of unlabeled points in* c_i.

Fig. 3: The ClusterClean algorithm.

User Effort We measure user effort as the number of labels the user has to manually provide. Formally, user effort is measured as:

$$e = \sum_{i=1}^{k} \mathbb{1}(|c_{i,u}| == |c_i|) \tag{2}$$

where k is the number of clusters, $|c_{i,u}|$ is the number of unlabeled points in cluster c_i and $|c_i|$ is the size of cluster c_i. We assume a user would have to label only one data point per unlabeled cluster. In the future we plan to explore whether the number of user labeled data points should differ based on cluster characteristics like its size or intra-cluster distance of points within the cluster.

The need for manual labels exists because after we have clustered the data, there will possibly be clusters in the data space that do not contain any labeled data points. These clusters indicate new patterns found in the unlabeled data that did not exist in the initial labeled set provided to ClusterClean and therefore ClusterClean cannot automatically select a label for them. We strive to keep the user effort as low as possible and therefore we set a threshold to the number of labels the user should provide.

Purity & user feedback trade-off As ClusterClean increases the value of k in every iteration and more clusters are created in the data space, the purity of the clustering naturally increases. This is because labeled data points get distributed

to more clusters. However, as we create more clusters the probability increases that we will have clusters that contain only unlabeled points and no labeled points within them and thus the user effort increases. As a result, we use a threshold on the user effort to counter balance the effect of creating a large number of clusters in order to achieve a high purity score.

Class Imbalance Purity does not work well for imbalanced data. For a heavily imbalanced dataset, no matter how bad k-means performs purity will always have a high value. This can affect how ClusterClean performs. For example, consider a scenario in which ClusterClean starts with $k_{min} = 1$ in the first iteration of the algorithm and all data points are placed in one cluster. In this case, purity is equal to the size of the majority class divided by the total number of data points in the dataset. If the purity threshold is below this number then all data points will get the label of the majority class and the algorithm will come to a halt. Therefore, to avoid such scenarios we set the purity threshold to be greater than the size of the majority class divided by the total number of points in the dataset.

III. Experiments

In this section, we present the experimental results of our approach. We implemented ClusterClean in Python using less than 50 lines of code. All of our experiments were run on a machine with 2.5GHz Intel Core i7 processor and 16 GB 2133 MHz LPDDR3 memory. For our initial small set of labeled data, we used a set of manually labeled messages sent from buyers to sellers in April 2018. In our experiments we varied the size of this initial labeled set from 50 messages, up to 1000 messages. For the unlabeled data (test set) that we would like to find the true labels for, we used a set of approximately 150K unreported messages from buyers to sellers sent in September 2018. Both our training set and test set have two classes of data: abusive (1) and non-abusive (0).

To evaluate our results we used the standard metrics of Precision ($precision = \frac{TP}{TP+FP}$) and Recall ($recall = \frac{TP}{TP+FN}$). As our ground truth we manually examined the 150K unreported messages and labeled them as abusive or non-abusive. The task for ClusterClean is to correctly clean the labels of these 150K unreported messages and match the labels we get from our manual labeling.

Table I lists the features we used to cluster the data and discover patterns in the incoming messages from buyers to sellers. Besides the features presented in table I, we used a doc2vec model [5] pre-trained on English Wikipedia texts to extract a vector from all the message bodies and create a numeric representation of the messages for our clustering purposes. We selected this model because it is straightforward to use and it has been known to produce great results for other document related tasks such as sentiment analysis. Using doc2vec we added 300 more features in our data set. In total, our feature set consisted of 308 features.

For our first experiment, we compared ClusterClean with two well-known techniques used in practice for getting clean data labels: kNN [6] and label propagation [7]. For both we use the scikit-learn implementation [8] in Python. kNN labels every unlabeled point with the majority label of its k nearest neighbors. Label propagation uses a similar technique but forwards labels with a "strength" based on the proximity (distance) of the points to the labeled ones. To tune these algorithms we used a small manually labeled set of messages (300 messages) from September 2018 and picked the set of parameters that yielded the best precision and recall on this validation set. Furthermore, we used an initial set of 500 labeled points for all three approaches (ClusterClean, kNN and Label propagation). For ClusterClean, we set the purity threshold to 90%, k_{min} to 2 and the user effort threshold to 10 manual labels. Finally, to remove the effects of randomness in our results we re-run the experiments for all three algorithms 10 times (with 10 different initial labeled sets of 500 points) and report the average numbers for these runs.

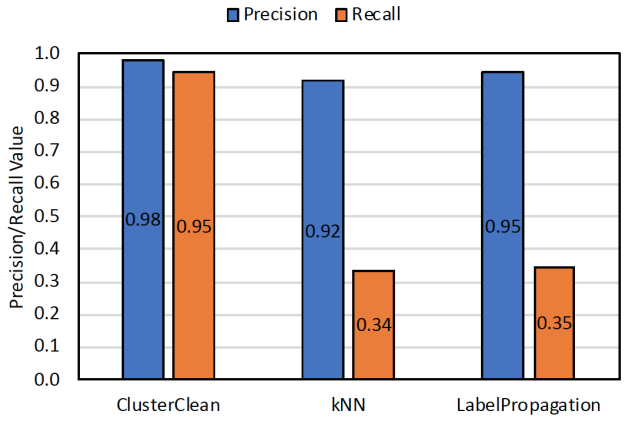
Figure 4a presents the results of this experiment. As the figure shows, ClusterClean outperforms the other two approaches both in precision and recall. Specifically, it outperforms kNN by 64.5% and Label Propagation by 63.4% in recall and while all three algorithms have high precision (more than 92%) ClusterClean performs the best with 98% precision. *The results indicate that ClusterClean is able to detect more abusive messages with a higher precision in the unlabeled data than kNN and Label Propagation.* The reason behind this is that ClusterClean accepts feedback from users (average user effort was 3.6 manually labeled messages over the 10 ClusterClean runs) and therefore is able to detect abusive patterns that do not exist in the initial labeled set and the other two approaches were not able to find.

Figure 4b shows the impact of the initial labeled size given as input to ClusterClean on its effectiveness. For this experiment, we vary the size of the initial labeled set from 100, 250, 500 up to 1000. We selected these sets randomly out of the approximately 16K manually labeled data from April 2018; we run every experiment with 10 different initial labeled sets and report average numbers for these runs and keep the 150K messages from September 2018 as our test set. We set the purity threshold to 90%, k_{min} to 2 and the user feedback threshold to 10. As the results show, even with only 100 messages in our initial labeled set ClusterClean is able to achieve a precision of 98.7% at a 92.5% recall. *Naturally, as the size of the initial labeled set increases and ClusterClean has more examples in order to label messages in the test set, the recall increases.* Specifically, when increasing the initial labeled set size from 100 data points to 1000 we see an increase of 3.5% in recall. Furthermore, as the recall increases we notice that there is an increase in the number of feedback the user has to provide: the user had to provide feedback for 1 label for an initial size of 100 data points and feedback for 4 data points for an initial set of 1000 messages. This is due to the larger number of clusters that have to formulate in the data space to reach a purity of 90%.

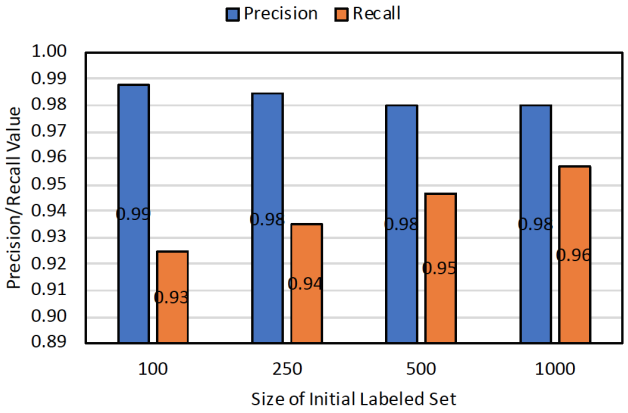
Next, we experiment with the purity threshold and its effect on ClusterCleans performance. For this experiment, we set

Feature	Description	Type
acount_age	Age of buyer account.	Numeric
has_order_1_year	Customer has an order in previous 365 days.	Boolean
count_words_in_message	Count of words in the message.	Numeric
msg_is_question	Message ends with question mark.	Boolean
msg_has_nonEng_characters	Message contains non english character.	Boolean
buyer_initiated_thread	Buyer sent the first message in the thread.	Boolean
count_messages_in_thread	Count of messages in the thread.	Numeric
seller_sent_message	Seller has sent a message in the thread.	Boolean

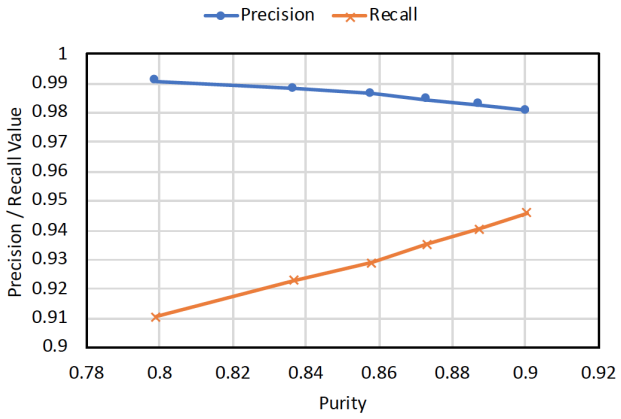
TABLE I: Set of features used in our experiments.



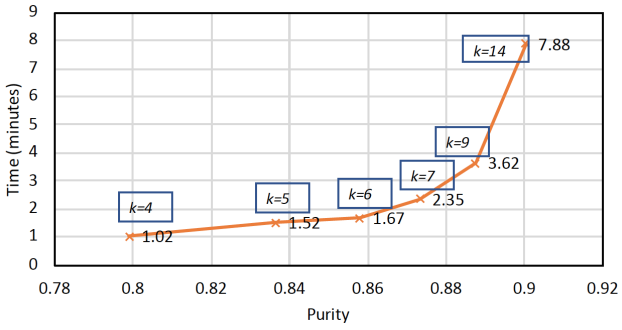
(a) Comparison of ClusterClean with kNN and Label Propagation.



(b) Effect of the size of initial labeled set on ClusterClean.



(c) Effect of the purity threshold on ClusterClean.



(d) ClusterClean's running time for different purity thresholds.

Fig. 4: ClusterClean effectiveness & efficiency experiments.

the initial labeled set size to 500 data points, we set k_{min} to 2 clusters and the user effort threshold to 10 manual labels. We run every experiment with 10 different initial labeled sets and report average numbers while we keep the 150K messages from September 2018 as our test set. We vary the purity threshold from 75% up to 90%; the natural event rate (abusive messages) was 71.7% in our dataset so setting the purity less than that would stop clustering at a very early stage were all messages would be clustered together as explained in section II-A. Figure 4c shows that *as we increase the purity threshold, more patterns are discovered in the data space but at a cost of extra running time* as Figure 4d shows. Specifically, recall increases from 91% when purity is 79.8% up to 94.6% for a purity of 90%. This is due to more clusters getting created in the data space and more patterns being recognized as abusive or non-abusive. At the same time, precision slightly declines because when more clusters are formulated a few non-abusive messages get mislabeled as abusive; however precision always stays at a high level (above 98% for all experiments).

Figure 4d shows the running time of ClusterClean for the above experiment. For this experiment, we do not include the time it takes a user to review and label a message; we estimate this time to be an average of 30 seconds per message. As the results show, as the purity increases and more clusters are formulated in the dataspace, the running time required by ClusterClean increases. Specifically, when we set the purity

threshold to 90%, ClusterClean formulated 14 clusters in the dataspace and it took approximately 7.8 minutes to clean the entire dataset. However, even for a purity of 88% that resulted in a very high precision of 98.5% with a recall of 93.7% it only took ClusterClean under 2.5 minutes to clean the dataset. *ClusterClean offers a major improvement over manual cleaning or designing heuristics since it can automatically clean a large amount of data with a high precision and recall in a very short time instead of requiring days or even weeks of human effort.*

A. Production Results

To build a machine learning model that blocks malicious messages from buyers to sellers and launch it in production we cleaned a sample of 5.15 million messages sent from June to September 2018 and used them as our training set. Furthermore, we cleaned 3.76 million messages from September to October 2018 and used them to test the performance of the model. In order to clean these messages, we manually labeled thousands of messages and we used heuristics to identify abusive patterns and automatically label subsets of the messages. This process took several weeks and it required a lot of human effort which inspired us to design ClusterClean. As part of our future work, we plan to re-train our models using training and testing data cleaned using ClusterClean. Next, we briefly present the results of our model currently in production in the US. Based on our experiments, ClusterClean matches the labels of the manual investigation with a 99% precision and a 95% recall and we expect our next model (using messages cleaned from ClusterClean) to be equivalent in performance.

To build the training and test data sets we considered all reported messages as abusive, since we noticed that the vast majority were indeed malicious after we audited a large sample and we cleaned all unreported messages to label them as abusive or not. For our feature set we used all features presented in table I and added approximately 2,200 unigram and bigram bag of words features. We identified the top unigrams and bigrams by transforming the message bodies into vector space using TFIDF (Term Frequency - Inverse Document Frequency) transformation and selected the k best features using the Chi Square test, a statistical test used to weed out the features that are most likely independent of the class and therefore irrelevant for classification. For the model, we trained a Gradient Boosted Tree model using XGBoost [9], a popular boosting tree library.

Our offline evaluation showed that the model is able to hit a precision of 99% percent with a 96.8% recall. The model is currently running in production blocking malicious messages from buyers to sellers with an online performance of a 99% precision and 78% recall. Approximately 40K messages flow on average per day through the model and it is blocking on average 7,093 malicious messages per day. i.e. 17.4% of the total messages, having a large impact on the experience of the sellers by protecting them from these messages.

IV. Related Work

The absence of large amounts of clean labeled data is one of the biggest bottlenecks in training and evaluating machine learning models. Therefore, a lot of attention has been given to gathering clean labels from the machine learning community. Next, we discuss related work in this area and how it differs from our approach.

Semi-supervised learning Semi-supervised learning algorithms [4], try to learn by both labeled and unlabeled data by using assumptions about label distributions, the structure of the data or the distance between data. Perhaps the most prominent algorithm is label propagation [7] where a points label propagates to neighboring points based on their proximity. The authors in [10] develop a semi-supervised clustering technique that uses both labeled and unlabeled data by using a genetic algorithm to optimize the objective function of k-means and produce clusters. In general, semi-supervised algorithms such as the ones mentioned here a) make assumptions about the structure of the data, b) do not solicit feedback from users and c) are agnostic to the domain and task they are being applied to. For a scenario such as in the BSM world where new spam attacks with new patterns happen often and the incoming data do not necessarily fit the previously labeled data structures, semi-supervised algorithms do not perform well because of the aforementioned reasons.

Weak supervision Weak supervision focuses on getting lower-quality data labels cheaply and efficiently. There are multiple ways this is achieved in weakly supervised scenarios such as using crowdsourcing, heuristic rules, leveraging algorithms that make assumptions about the expected label distributions or taking opportunistic advantage of existing resources (e.g. knowledge bases, pre-trained models). In [11] they present a system that automatically labels unlabeled data points based on labeling functions that express arbitrary heuristics written by the users. In contrast to this system, with our approach we discover the patterns automatically and we do not require the user to write any labeling functions. In [12] they propose a technique for labeling data points by taking advantage of a small, labeled dataset and automatically generating heuristics based on this set to be used for labelling unlabeled data. Contrary to our approach, this technique for labelling new data points is only based on the initial small labeled set and will miss all new patterns in the unlabeled data that do not exist in the labeled set.

Active learning Active learning studies the problem of label acquisition [13]: having a training set that consists of both labeled and unlabeled data, active learning algorithms are used in order to select the most informative points and query the user to label them. The size of labeled sets that are created using active learning techniques is limited to the labeling effort of the user. For complex models such as the one we built for the BSM service (2000+ features) we generally need a training set too large to conveniently label by hand. Thus, such problems are a better fit for weak supervision or semi-supervised learning such as the algorithm we are presenting

here that can generate a large number of labeled data. An interesting approach is presented in [14] where they build a model on both clean and dirty data and iteratively ask the user to clean those records that are most likely to affect the classification results. However, this technique is tied to a specific type of models: only linear regression and SVMs and cannot be applied without significant modifications and development cost to our XGBoost model.

V. Conclusions

We presented ClusterClean, an algorithm which is able to a) to automatically clean the labels of a large number of training data points based on an initial small labeled set and b) detect new data patterns such as incoming new spam attacks and solicit feedback from users to decide on their label. We highlighted the advantages of the method: 1) it is effective in identifying new spam patterns not observed in the already existing labeled data set with minimal feedback from a user, 2) it is efficient in cleaning large amounts of data accurately in a short time and 3) it is applicable to other domains, as well as multi-classification problems. Finally, we also presented results from applying ClusterClean on approximately 150K real messages from buyers to sellers, where ClusterClean took less than 2.4 minutes to clean the entire dataset and identified 92.7% of the abusive messages with a 98.5% precision, outperforming in effectiveness other popular techniques such as kNN and label propagation.

References

[1] Yuji Roh, Geon Heo, and Steven Euijong Whang. A Survey on Data Collection for Machine Learning: a Big Data - AI Integration Perspective. CoRR, abs/1811.03402, 2018.

[2] Alexander J. Ratner, Christopher De Sa, Sen Wu, Daniel Selsam, and Christopher Ré. Data Programming: Creating Large Training Sets, Quickly. In NIPS, 2016.

[3] Zhi-Hua Zhou. A Brief Introduction to Weakly Supervised Learning. In National Science Review, volume 5, pages 4453, 2018.

[4] Xiaojin Zhu. Semi-Supervised Learning Literature Survey. Technical report, Department of Computer Science, University of Wisconsin, Madison, 2006.

[5] Jey Han Lau and Timothy Baldwin. An Empirical Evaluation of doc2vec with Practical Insights into Document Embedding Generation. In Proceedings of the 1st Workshop on Representation Learning for NLP, pages 7886, 2016.

[6] N. S. Altman. An Introduction to Kernel and Nearest-Neighbor Nonparametric Regression. The American Statistician, 46(3):175185, 1992.

[7] Xiaojin Zhu and Zoubin Ghahramani. Learning from Labeled and Unlabeled Data with Label Propagation. Technical report, CMU-CALD 02-107, 2002.

[8] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine Learning in Python. Journal of Machine Learning Research, 12:28252830, 2011.

[9] Tianqi Chen and Carlos Guestrin. Xgboost: A scalable tree boosting system. In KDD, 2016.

[10] Ayhan Demiriz, Kristin Bennett, and Mark J. Embrechts. Semi-Supervised Clustering Using Genetic Algorithms. In Artificial Neural Networks in Engineering, pages 809814, 1999.

[11] Alexander Ratner, Stephen H. Bach, Henry R. Ehrenberg, Jason Alan Fries, Sen Wu, and Christopher R. Snorkel: Rapid Training Data Creation with Weak Supervision. PVLDB, 11(3):269282, 2017.

[12] Paroma Varma and Christopher Ré. Snuba: Automating Weak Supervision to Label Training Data. PVLDB, 12(3):223236, 2018.

[13] Burr Settles. Active Learning Literature Survey. Technical report, University of Wisconsin, Madison, 2010.

[14] Sanjay Krishnan, Jiannan Wang, Eugene Wu, Michael J. Franklin, and Ken Goldberg. Active- Clean: Interactive Data Cleaning for Statistical Modeling. PVLDB, 9(12):948959, 2016.