Spotting Spurious Data with Neural Networks

Hadi Amiri, Timothy A. Miller, Guergana Savova
Boston Children's Hospital Informatics Program, Harvard Medical School
{firstname.lastname}@childrens.harvard.edu

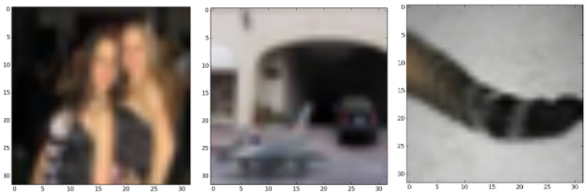
Abstract

Automatic identification of spurious instances (those with potentially wrong labels in datasets) can improve the quality of existing language resources, especially when annotations are obtained through crowdsourcing or automatically generated based on coded rankings. In this paper, we present an effective approach inspired by queueing theory and psychology of learning to automatically identify spurious instances in datasets. Our approach discriminates instances based on their "difficulty to learn," determined by a downstream learner. Our method can be applied to *any* dataset assuming the existence of a neural network model for the target task of the dataset. Our best approach outperforms competing state-of-the-art baselines and has a MAP of 0.85 and 0.22 in identifying spurious instances in synthetic and carefully-crowdsourced real-world datasets respectively.

1 Introduction

The importance of error-free language resources cannot be overstated as errors can inversely affect interpretations of the data, models developed from the data, and decisions made based on the data. Although the quality of language resources can be improved through good annotation guidelines, test questions, etc., annotation noise still exists (Gupta et al., 2012; Lasecki et al., 2013). For example, Figure 1 shows sample spurious instances (those with potentially wrong labels) in CIFAR-10 (Krizhevsky, 2009) which is a benchmark dataset for object classification. Spurious instances can mislead systems, and, if available in test data, lead to unrealistic comparison among competing systems.

Previous works either directly identify noise in datasets (Hovy et al., 2013; Dickinson and Meurers, 2003; Eskin, 2000; Loftsson, 2009), or develop models that are more robust against noise (Guan et al., 2017; Natarajan et al., 2013; Zhu et al., 2003; Zhu and Wu, 2004). Furthermore, recent works on adversarial perturbation have tackled this problem (Goodfellow et al., 2015; Feinman et al., 2017). However, most previous approaches require either annotations generated by each individual annotator (Guan et al., 2017), or both task-specific and instance-type (genuine or adversarial) labels for training (Hendrik Metzen et al., 2017; Zheng et al., 2016), or noise-free data (Xiao et al., 2015). Such information is often not available in the final release of most datasets.



(a) Truck (b) Airplane (c) Cat

Figure 1: Spurious instances in CIFAR-10. (a) not a truck, (b) missing annotation for car as another object category, (c) incomplete image of a cat.

Current approaches utilize prediction probability/loss of instances to tackle the above challenges in identifying spurious instances. This is because prediction probability/loss of spurious instances tend to be lower than that of genuine instances (He and Garcia, 2009). In particular, the Bayesian Uncertainty model (Feinman et al., 2017) defines spurious instances as those that have greater uncertainty (variance) in their stochastic predictions, and the Variational Inference model (Rehbein and Ruppenhofer, 2017; Hovy et al., 2013) expects greater posterior entropy in predictions made for spurious instances.

In this paper, our hypothesis is that spurious instances are *frequently* found to be difficult to

Proceedings of NAACL-HLT 2018, pages 2006–2016
New Orleans, Louisiana, June 1 - 6, 2018.

learn during training process. This difficulty in learning stems from the intrinsic discrepancy between spurious and the cohort of genuine instances which frequently makes a learner less confident in predicting the *wrong* labels of spurious instances. Based on this hypothesis, we present two frameworks which are inspired by findings in queueing theory and psychology, namely Leitner queue network (Leitner, 1974) and Curriculum Learning (Bengio et al., 2009). Our frameworks can be considered as schedulers that schedule instances to train a downstream learner (e.g. a neural network) with respect to "easiness"/"difficulty" of instances - determined by the extent to which the learner can correctly label (e.g. classify) instances during the training process. The two frameworks, however, differ in their views on the theory of learning as we describe below:

Curriculum learning is inspired by the learning principle that humans can learn more effectively when training starts with *easier* concepts and gradually proceeds with more difficult ones (Bengio et al., 2009). On the other hand, Leitner system is inspired by *spaced repetition* (Dempster, 1989; Cepeda et al., 2006), the learning principle that effective and efficient learning can be achieved by working more on difficult concepts and less on easier ones. Both frameworks are effective, conceptually simple, and easy to implement.

The contributions of this paper are as follows: (a) we develop a cognitively-motivated and effective algorithm for identifying spurious instances in datasets, (b) our approach can be applied to *any* dataset without modification if there exists a neural network architecture for the target task of the dataset, and (c) we release a tool that can be easily used to generate a ranked list of spurious instances in datasets.[1] Our tool requires a dataset and its corresponding network architecture to generate a ranked list of spurious instances in the dataset.

Our best approach (Leitner model) has a mean average precision (MAP) of 0.85 and 0.22 in identifying spurious instances on real-world and synthetic datasets and outperforms competing state-of-the-art baselines.

2 Method

We assume that our learner is a neural network which trains for k iterations until convergence. Furthermore, we assume that spurious and genuine instances are mixed at training time and the network is only provided with task-specific but not genuine/spurious labels for the instances.

[1] https://scholar.harvard.edu/hadi/spot

Algorithm 1. Curriculum Spotter

Input: $\mathbf{H}$: training data, $\mathbf{V}$: validation data, k : number of iterations
Output: Ranked list of spurious instances

$batch = \mathbf{H}$
$S^0[h_j] = 0$ for $h_j \in \mathbf{H}$
For $epoch = 1$ to k:
  $model = \mathbf{train}(batch, \mathbf{V})$
  $loss = \mathbf{Loss}(model, \mathbf{H})$
  $\lambda = \mathbf{compute\_lambda}(model, loss, \mathbf{H})$
  $easy\_batch = \mathbf{sample\_easy}(\lambda, loss, \mathbf{H})$
  $hard\_batch = \mathbf{H} - easy\_batch$
  $batch = easy\_batch + \mathbf{top}(\frac{epoch}{k}, hard\_batch)$
  $S^{epoch} = \mathbf{update\_stat}(S^{epoch-1}, hard\_batch, loss)$
End for
return $\mathbf{sort}(S^k, \mathbf{H}, loss)$

Figure 2: Curriculum Spotter. $\mathbf{Loss}(.)$ computes loss of a network with respect to given instances, $\mathbf{compute\_lambda}(.)$ computes the average loss of current model for correctly classified instances, $\mathbf{sample\_easy}(.)$ creates list of easy instances using current loss values, $\mathbf{top}(.)$ returns $epoch/k$ fraction of easiest hard instances, $\mathbf{update\_stat}(.)$ scores instances according to Eq. (1), and $\mathbf{sort}(.)$ ranks instances based on the resulting scores S updated by Eq. (2).

2.1 Curriculum Learning

Bengio et al. (2009) and Kumar et al. (2010) developed training paradigms which are inspired by the learning principle that humans can learn more effectively when training starts with easier concepts and gradually proceeds with more difficult ones. Since easiness of information is not readily available in most datasets, previous approaches used heuristic techniques (Spitkovsky et al., 2010; Basu and Christensen, 2013) or optimization algorithms (Jiang et al., 2015, 2014) to quantify easiness for instances. These approaches consider an instance as easy if its prediction loss is smaller than a threshold (λ). Given a neural network as the learner, we adopt curriculum learning to identify spurious instances as follows (see Figure 2):

At each iteration i, we divide all instances into *easy* and *hard* batches using the iteration-specific threshold λ_i and the loss values of instances at iteration i, obtained from the current partially-trained network. All instances with a loss smaller than λ_i are considered as easy and the rest are consid-

ered as hard. All easy instances in conjunction with $\delta_i \in [0, 1]$ fraction of easiest hard instances (those with smallest loss values greater than λ_i) are used for training at iteration i. We set each λ_i to the average[2] loss of training instances that are correctly classified by the current partially-trained network. Furthermore, at each iteration $i > 1$, we set $\delta_i = i/k$ where k is the total number of iterations. In this way, difficult instances are gradually introduced to the network at every new iteration.

The $\mathbf{update\_stat}(.)$ function in Figure 2 scores instances based on their frequency of occurrence in the hard batch. In particular, for each instance h_i:

$$S^e(h_i) = S^{e-1}(h_i) + \mathbb{1}_{hard\_batch^e}(h_i) \times \Big(\frac{1}{|hard\_batch^e|} + loss^e(h_i)\Big), \quad (1)$$

where $S^e(h_i)$ is the score of h_i at iteration e, $\mathbb{1}_Y(x)$ is an indicator function which is 1 when $x \in Y$ and otherwise 0, $hard\_batch^e$ indicates the set of hard instances at iteration e, and $loss^e(h_i)$ is the loss of the network for h_i at iteration e. The above function assigns higher scores to instances that are frequently considered as hard instances by the curriculum learning framework (such instances are ranked higher in the final ranked list of spurious instances). It also assigns a final score of $S^k(h_i) = 0$ to instances that are treated as easy instances throughout the training process, i.e. those that have a loss smaller than the iteration-specific threshold λ_i at each iteration i and, therefore, are always placed in the $easy\_batch$. To break the tie for these instances in the final ranking, we resort to their final loss values as follows:

$$S^k(h_i) = loss^k(h_i), \text{if } S^k(h_i) = 0. \quad (2)$$

2.2 Leitner System

The Leitner System is inspired by the broad evidence in psychology that shows human ability to retain information improves with repeated exposure and exponentially decays with delay since last exposure (Cepeda et al., 2006). Spaced repetition forms the building block of many educational devices, such as flashcards, in which small pieces of information are repeatedly presented to a learner on a schedule determined by a spaced repetition algorithm. Such algorithms show that human learners can learn efficiently and effectively by increasing intervals of time between subsequent reviews of previously learned materials (Dempster, 1989; Novikoff et al., 2012). We adopt the Leitner system to identify spurious instances as follows:

Algorithm 2. Leitner Spotter

Input: $\mathbf{H}$: training data, $\mathbf{V}$: validation data, k : number of iterations, n : number of queues
Output: Ranked list of spurious instances

$Q = [q_0, q_1, \ldots, q_{n-1}]$
$q_0 = [\mathbf{H}]$, $q_i = []$ for $i \in [1, n-1]$
$S^0[h_j] = 0$ for $h_j \in \mathbf{H}$
For $epoch = 1$ to k:
  $batch = []$
  For $i = 0$ to $n - 1$:
    If $epoch\%2^i == 0$:
      $batch = batch + q_i$
  End For
  $promos, demos, loss = \mathbf{train}(batch, \mathbf{V})$
  $\mathbf{update\_queue}(Q, promos, demos)$
  $S^{epoch} = \mathbf{update\_stat}(S^{epoch-1}, Q, loss)$
End for
return $\mathbf{sort}(S^k, \mathbf{H}, loss)$

q_0 epochs = $\{1, 2, 3, 4, 5, \ldots\}$
q_1 epochs = $\{2, 4, 6, 8, 10, \ldots\}$
q_2 epochs = $\{4, 8, 12, 16, 20, \ldots\}$
...

Figure 3: Leitner Spotter. The $\mathbf{train}(.)$ function trains the network using instances in the current $batch$, $\mathbf{update\_queue}(.)$ promotes the correctly classified instances–$promos$–to their next queues and demotes the wrongly classified ones–$demos$–to q_0, $\mathbf{update\_stat}(.)$ scores instances according to Eq. (3), and $\mathbf{sort}(.)$ ranks instances based on resulting scores S updated by Eq. (4).

Suppose we have n queues $\{q_0, q_1, \ldots, q_{n-1}\}$. The Leitner system initially places all instances in the first queue, q_0. As Figure 3 shows, the system trains with instances of q_i at every 2^i iterations. At each iteration, only instances in the selected queues will be used for training the network. During training, if an instance from q_i is correctly classified by the network, the instance will be "promoted" to q_{i+1}, otherwise it will be "demoted" to the first queue, q_0. Therefore, as the network trains through time, higher queues will accumulate easier instances which the network is most accurate about, while lower queues carry either hard or potentially spurious instances. This is because of the intrinsic discrepancy between spurious instances and the cohort of genuine instances which makes the network less confident in predicting the wrong labels of spurious instances. Figure 3 (bot-

[2] We also considered maximum and median loss, but average loss led to greater training gain in terms of effectiveness.

tom) provides examples of queues and their corresponding processing epochs.

The **update_stat**(.) function in Figure 3 scores instances based on their occurrence in q_0. In particular, for each instance h_i:

$$S^e(h_i) = S^{e-1}(h_i) + \mathbb{1}_{q_0^e}(h_i) \times \Big(\frac{1}{|q_0^e|} + loss^e(h_i)\Big), \quad (3)$$

where $|q_0^e|$ indicates the number of instance in q_0 at iteration e. The above function assigns higher scores to instances that frequently occur in q_0. It also assigns a final score of $S^k(h_i) = 0$ (at the last iteration) to instances that have never been demoted to q_0. To break the tie for such instances, we use their final loss value as follows:

$$S^k(h_i) = loss^k(h_i), \text{if } S^k(h_i) = 0. \quad (4)$$

3 Experiments

3.1 Evaluation Metrics

We employ a TREC-like evaluation setting to compare models against each other. For this, we create a pool of K most spurious instances identified by different models. If needed, e.g. in case of real-world datasets, we manually label all instances in the pool and come to agreement about their labels. Then, we compare the resulting labels with the original labels in the dataset to determine spurious/genuine instances. We compare models based on the standard TREC evaluation measures, namely mean average precision (MAP), precision after r instances are retrieved (P@r), and, only for synthetic data, precision after all spurious instances are retrieved (Rprec). We use the `trec-eval` toolkit to compute performance of different models.[3]

3.2 Datasets

We develop synthetic and real-world datasets for our experiments. Since, in contrast to real-world datasets, (most[4]) synthetic datasets do not contain any noisy instances, we can conduct large-scale evaluation by injecting spurious instances into such datasets. Table 1 shows detail information about our datasets.

[3] http://trec.nist.gov/trec_eval/

[4] Some synthetic datasets may contain noise, see sample inconsistencies that our model identified in the bAbi dataset (Weston et al., 2016) in Table 3.

Dataset	Train/Val	SpRatio	Input	Output
Synthetic Dataset				
Addition	10K/2K	$\alpha \in (0, 0.5]$	$(x,y \geq 0)$	$x+y$
Real-world Datasets				
Twitter	10K/1K	0.30	brand tweet	pos/neg
Reddit	4K/400	0.23	cancer post	rel/irel

Table 1: Dataset Information. "SpRatio" shows the fraction of spurious instances in the TREC pool; size of the pool for Addition is 10K instances with no limit on the top K retrieved instances; the corresponding value for Twitter and Reddit datasets are 198 and 152 posts respectively for top $K = 50$.

3.2.1 Synthetic Dataset

The Addition dataset, initially developed by Zaremba and Sutskever (2014), is a synthetic dataset in which an input instance is a pair of non-negative integers smaller than 10^l and the corresponding output is the arithmetic sum of the input; we set $l = 4$ in our experiments. Since this dataset contains only genuine instances, we create noisy datasets by injecting $\alpha \times N$ spurious instances into $(1 - \alpha) \times N$ genuine instances, where $N = 10K$ is the total number of training instances and $\alpha \leq 0.5$ indicates the noise level in the dataset. We create spurious instances as follows: given three random numbers $x_i, x_j, x_k \in [0, 10^l)$ such that $x_j \neq x_k$, the wrong sum (output) for the pair (x_i, x_j) is computed as:

$$\max(0, x_i + (-1)^o \times x_k),$$

where o is a random variable that takes values from $O = \{1, 2\}$ with equal probability.

3.2.2 Real-world Datasets

We crowdsource annotations for two real-world datasets, namely Twitter and Reddit posts (see Table 1). For quality control, we carefully develop annotation schemas as well as high quality test questions (see below) to minimize the chances of spurious labels in the resulting annotations.

The Twitter dataset contains tweets about a telecommunication brand. Tweets contain brand name or its products and services. Annotators are instructed to label tweets as positive/negative if they describe positive/negative sentiment about the target brand. We use 500 labeled instances for annotation quality assurance and ignore data generated by annotators who have less than 80% accuracy on these instances. The resulting Fleiss' kappa (Fleiss, 1971) is $\kappa = 0.66$ on our Twitter dataset which indicates substantial agreement.

The Reddit dataset includes posts about colon, breast, or brain cancer. These posts contain phrases like *colon cancer*, *breast cancer*, or *brain cancer*. Annotators are instructed to label a post as "relevant" if it describes a patient's experience (including sign and symptoms, treatments, etc.,) with respect to the cancer. In contrast, "irrelevant" posts are defined as generic texts (such as scientific papers, news, etc.,) that discuss cancer in general without describing a real patient experience. We use 300 labeled instances for annotation quality assurance and ignore annotations generated by users who have less than 80% accuracy on these instances. The resulting Fleiss' kappa is $\kappa = 0.48$ for the Reddit dataset which indicates moderate agreement.

3.3 Settings

For the synthetic Addition dataset, we set the size of the TREC pool to $K = 10,000$ (size of training data) which indicates there is no limitation on the number of spurious instances that a model can retrieve; note that we have a spurious/genuine label for each instance in the Addition dataset and therefore do not need to label the resulting TREC pool manually. Furthermore, we consider the LSTM network developed by Sutskever et al. (2014) as the downstream learner.[5] Without noise in data, this network obtains a high accuracy of 99.7% on the Addition task.

For the real-world datasets, we allow each model to submits its top 50 most spurious instances to the TREC pool (we have five models including our baselines). As mentioned before, we manually label these instances to determine their spurious/genuine labels. This leads to TREC pools of size 198 and 152 posts (with 59 and 35 spurious instances) for the Twitter and Reddit datasets respectively.

We use the MLP network `fastText` (Joulin et al., 2017) as the downstream learner - for more effective prediction, we add a `Dense` layer of size 512 before the last layer of `fastText`. This network obtains accuracy of 74.6% and 70.2% on Twitter and Reddit datasets respectively.

Finally, for the Leitner system, we experiment with different queue lengths, $n = \{3, 5, 7\}$, and set $n = 5$ in the experiments as this value leads to slightly better performance in our experiments.

[5] http://github.com/fchollet/keras/blob/master/examples/addition_rnn.py

3.4 Baselines

We consider the following baselines; each baseline takes a dataset and a model as input and generates a ranked list of spurious instances in the dataset:

• **Prediction Probability (PP)**: Since prediction loss of spurious instances tend to be higher than that of genuine ones (He and Garcia, 2009; Hendrycks and Gimpel, 2016), this baseline ranks instances in descending order of their prediction loss after networks are trained through standard (rote) training.

• **Variational inference (VI)** (Hovy et al., 2013; Rehbein and Ruppenhofer, 2017): This model approximates posterior entropy from several predictions made for each individual instance (see below).[6]

• **Bayesian Uncertainty (BU)** (Feinman et al., 2017): This model ranks instances with respect to the uncertainty (variance) in their stochastic predictions.[7]

BU estimates an uncertainty score for each individual instance by generating $T = 50$ predictions for the instance from a distribution of network configurations. The prediction disagreement tends to be common among spurious instances (high uncertainty) but rare among genuine instances (low uncertainty). Uncertainty of instance x with predictions $\{\mathbf{y}_1, \ldots, \mathbf{y}_T\}$ is computed as follows:

$$\frac{1}{T}\sum_{i=1}^{T}\mathbf{y}_i^\top\mathbf{y}_i - \Big(\frac{1}{T}\sum_{i=1}^{T}\mathbf{y}_i\Big)^\top\Big(\frac{1}{T}\sum_{i=1}^{T}\mathbf{y}_i\Big).$$

Variational inference (VI) (Rehbein and Ruppenhofer, 2017; Hovy et al., 2013) detects spurious instances by approximating the posterior $p(y|x)$ with a simpler distribution $q(y)$ (called variational approximation to the posterior) which models the prediction for each instance. The model jointly optimizes the two distributions through EM: in the E-step, q is updated to minimize the divergence between the two distributions, $D(q||p)$; in the M-step, q is kept fixed while p is adjusted. The two steps are repeated until convergence. Instances are then ranked based on their posterior entropies. Similar to BU, we generate $T = 50$ predictions for each instance.

For both BU and VI baselines, we apply a dropout rate of 0.5 after the first and last hidden

[6] http://isi.edu/publications/licensed-sw/mace/

[7] http://github.com/rfeinman/detecting-adversarial-samples

layers of our downstream networks to generate predictions. See (Gal and Ghahramani, 2016) for the ability of dropout neural networks in representing model uncertainty.

3.5 Experimental Results

The overall mean average precisions (MAPs) of different models on synthetic and real-world datasets are reported in Table 2. For the synthetic dataset (Addition), we report average MAP across all noise levels, and for real-world datasets (Twitter and Reddit), we report average MAP at their corresponding noise levels obtained from corresponding TREC pools. We use t-test for significance testing and asterisk mark (*) to indicate significant difference at $\rho = 0.05$ between top two competing systems.

The results show that Leitner (Lit) and Bayesian uncertainty (BU) models considerably outperform prediction probability (PP) and curriculum learning (CL) on both synthetic and real-world datasets. In case of real-world datasets, we didn't find significant difference between top two models perhaps because of the small size of corresponding TREC pools (198 Twitter posts and 152 Reddit posts, see Table 1). Overall, BU and Lit show average MAP of 0.81, and 0.85 on the synthetic dataset and 0.15, 0.22 on real-world datasets respectively. The higher performance of Lit indicates that spurious instances often appear in q_0. The lower performance of CL, however, can be attributed to its training strategy which may label spurious instances as easy instances if their loss values are smaller than the loss threshold (section 2.1). The large difference between the performances of Lit and CL (two methods based on repeated scoring across training epochs) shows that the way that repetition is utilized by different methods largely affects their final performance in spotting spurious instances. In addition, VI shows lower performance than BU and Lit on synthetic data, but comparable performance to BU on real-world datasets.

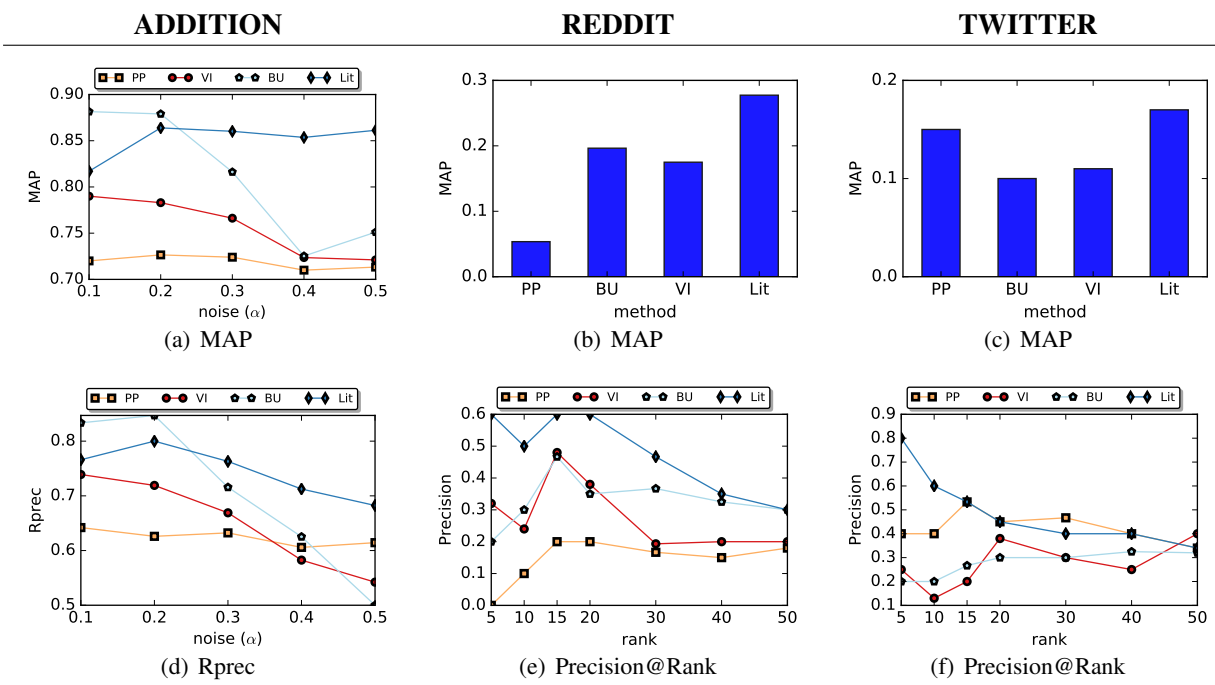
Furthermore, the results show that the performance of all models are considerably lower on real-world datasets than the synthetic dataset. This could be attributed to the more complex nature of our real-world datasets which leads to weaker generalizability of downstream learners on these datasets (see next section for discussion on training performance). This can in turn inversely affect the performance of different spotters, e.g. by encouraging most instances to be considered as hard and thus placed in lower queues of Lit or in the hard batch of CL, or by increasing the prediction uncertainty and entropy in case of BU and VI respectively. In addition, as we mentioned before, we carefully setup the annotation task to minimize the chances of spurious labels in the resulting annotations. Therefore, we expect a considerably smaller fraction of spurious instances in our real-world datasets.

	Synthetic noise = $[0.1, 0.5]$	**Real-world** noise = $\{0.28, 0.35\}$
PP	0.719	0.100
CL	0.718	0.067
VI	0.757	0.142
BU	0.811	0.148
Lit	**0.851***	**0.225**

Table 2: Average overall MAP performance across datasets and noise levels.

Figures 4(a) and 4(d) report MAP and precision after all spurious instances have been retrieved (Rprec) on Addition at different noise levels respectively; note that $\alpha = 0.5$ means equal number of spurious and genuine instances in training data (here, we do not report the performance of CL due to its lower performance and for better presentation). First, the results show that Lit and BU considerably outperform PP and VI. Furthermore, BU shows considerably high performance at lower noise levels, $\alpha \leq 0.2$, while Lit considerably outperforms BU at greater noise levels, $\alpha > 0.2$. The lower performance of BU at higher noise levels might be because of the poor generalizability of LSTM in the context of greater noise which may increase the variance in the prediction probabilities of most instances (see section 3.6 for our note on training performance). In terms of average Rprec, the overall performance of PP, CL, VI, BU, and Lit models is 0.62, 0.57, 0.65, 0.70, and 0.74 respectively on the Addition dataset across all noise levels (see the corresponding values for MAP in Table 2). The lower Rprec values than MAP indicate that some spurious instances are ranked very low by models. These are perhaps the most difficult spurious instances to identify.

For the real-world datasets, we only report MAP and P@r (precision at rank r) as spurious/genuine labels are only available for those instances that make it to the TREC pool but not for all instances. The results on Reddit, Figures 4(b) and 4(e) respectively, show that Lit outperforms other mod-



(a) MAP (b) MAP (c) MAP

(d) Rprec (e) Precision@Rank (f) Precision@Rank

Figure 4: Models performance across datasets. MAP indicates mean average precision, Rprec indicates precision after all spurious instances are retrieved, and P@r indicates precision after r instances are retrieved. In (a) and (b) performance values are reported at different noise levels $\alpha \in (0, 0.5]$.

els, but VI and BU show comparable MAP (in contrast to their performance on Addition). Furthermore, Figure 4(e) shows that Lit generates a more accurate ranked list of spurious instances and consistently outperforms other models at almost all ranks. In particular, it maintains a MAP of around 60% at rank 20, while other models have consistently lower MAP than 50% at all ranks.

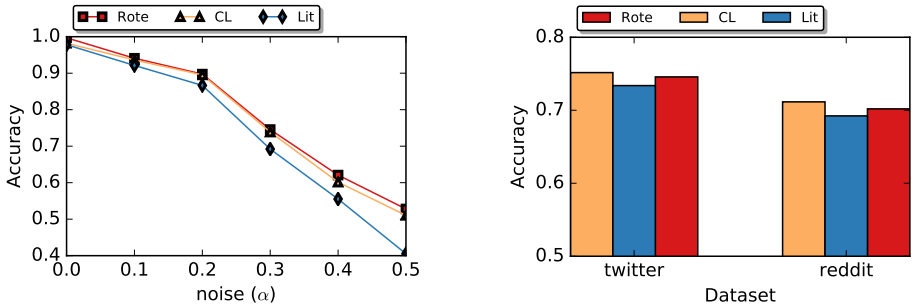
The results on the Twitter dataset, Figures 4(c) and 4(f), show that Lit outperforms other models. However, interestingly, PP outperforms BU in terms of both MAP and P@r across almost all ranks. This result could be attributed to the substantial annotation agreement on Twitter dataset (Fleiss' $\kappa = 0.66$) which could make network predictions/loss values more representative of gold labels. Figure 4(f) also shows that Lit is the most precise model in identifying spurious instances. Note that P@5 is an important metric in search applications and as Figures 4(e) and 4(f) show, at rank 5, Lit is 2-3 times more precise than the best-performing baseline on our real-world datasets.

Given *any* dataset and its corresponding neural network, our Leitner model simultaneously trains the network and generates a ranked list of spurious instances in the dataset. For this purpose, the model tracks loss values and occurrences of instances in the lower Leitner queue during training.

3.6 Notes on Training Performance

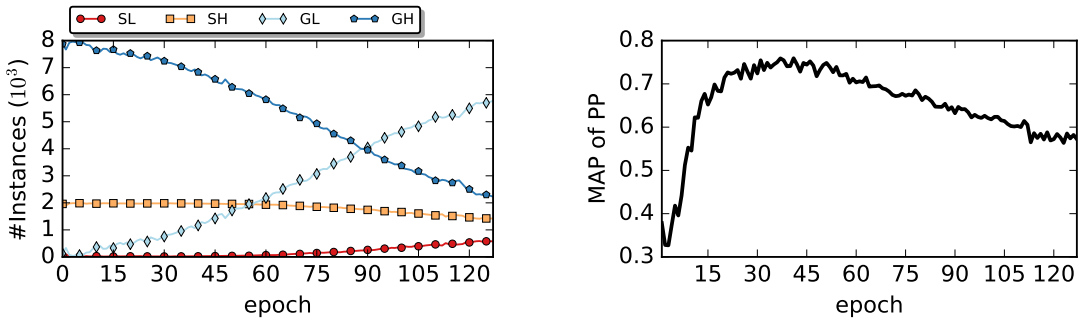
Figure 5(a) shows the accuracy of the LSTM network (Sutskever et al., 2014) trained with different training regimes on the validation data of Addition with different noise levels; note that Rote represents standard training where at each iteration all instances are used to train the network. As the results show, at lower noise levels, the training performance (i.e. the generalizability/accuracy of the LSTM network) is generally high and comparable across different training regimes, e.g. close to 100% at $\alpha = 0$. However, Lit leads to a slightly weaker training performance than CL and Rote as the noise level increases. This is because Lit learns from spurious instances more frequently than genuine ones. This may decrease the training performance of Lit, especially with greater amount of noise in data. However, this training strategy increases the spotting performance of Lit as spurious instances seem to occur in lower queues of Leitner more frequently, see Figure 4.

In addition, the accuracy of `fastText` (Joulin et al., 2017) is reported in Figure 5(b). The results show that different training regimes lead to comparable performance on both datasets (accuracy of around 75% and 70% on Twitter and Reddit respectively). The relatively lower training per-



(a) Addition/LSTM (b) Twitter and Reddit/`fastText`

Figure 5: Accuracy of LSTM/`fastText` trained with different schedulers. Rote: standard training.



(a) Spurious/genuine instances with low/high loss (b) MAP of prediction loss (PP)

Figure 6: Behavior of prediction loss across training epochs. (a) S: spurious, G: genuine; L: low, H: high.

formance on these datasets can contribute to the weaker performance of spotters on these datasets.

3.7 Discussion

We first report insights on why prediction loss alone is not enough to identify spurious instances. For this analysis, we track the loss of spurious and genuine instances at each training iteration.[8] Figure 6(a) shows the number of spurious/genuine instances with low/high loss at each epoch; where, we use the average loss of *correctly* classified training instances at each epoch as a pivot value to determine the low and high loss values for that epoch. Initially, almost all spurious and genuine instances have high loss values (see SH and GH in Figure 6(a)). However, the sheer imbalance of genuine instances relative to spurious instances means that there will still be a relatively large number of genuine instances with large loss - these are simply difficult instances. Furthermore, the number of spurious instances with lower loss values (SL) slowly increases as the network gradually learns the wrong labels of some spurious instances; this, in turn, decreases the expected loss of such instances. Since PP merely ranks instances based on loss values, the above two factors may cause some spurious instances to be ranked *lower* than genuine ones by PP; see Figure 6(b) for MAP of PP in detecting spurious instances at every iteration. Using queue information from the Leitner system adds information that loss alone does not; we suspect that the learner can find principled solutions that trade off losses between one difficult genuine instance and another (causing them to bounce between q_0 and higher queues) without harming total loss, but that the more random nature of spurious instances means that they are consistently misclassified, staying in q_0. Verifying this hypothesis will be the subject of future work.

For our second analysis, we manually inspect highly ranked instances in q_0 of Lit. We use the synthetic dataset bAbi (Weston et al., 2016) which is a systematically generated QA dataset for which the task is to generate an answer given a question and its corresponding story. As the learner, we use an effective LSTM network specifically developed for this task.[9] Table 3 shows sample instances from bAbi which are highly ranked by

[8] Here we use Addition with $N = 10$K training instances and noise level of $\alpha = 0.2$.

[9] https://github.com/fchollet/keras/blob/master/examples/babi_rnn.py

Story: Mary traveled to the garden. Daniel went to the garden. Mary journeyed to the kitchen. Mary went **back to the hallway**. Daniel traveled to the office. Daniel moved to the garden. Sandra went back to the kitchen. John traveled to the bathroom. **Question**: Where is Mary? **Answer**: hallway
Story: John went to the office. Daniel journeyed to the office. Sandra picked up the football **there**. Sandra went to the bedroom. Sandra left the football there. Sandra went **back to the kitchen**. Sandra traveled to the hallway. Sandra moved to the garden. **Question**: Where is the football? **Answer**: bedroom

Table 3: Sample inconsistencies in bAbi dataset.

Lit. We observe inconsistencies in the given stories. In the first case, the story contains the sentence "Mary went *back* to the hallway," while the previous sentences indicate that Mary was in the "garden/kitchen" but not "hallway" before. In the second case, the sentence "Sandra picked up the football *there*" is inconsistent with story because the word "there" doesn't refer to any specific location. We conjecture that these inconsistencies can mislead the learner or at least make the learning task more complex. Our model can be used to explore language resources for such inconsistencies.

4 Related Work

There is broad evidence in psychology that shows human ability to retain information improves with repeated exposure and exponentially decays with delay since last exposure. Ebbinghaus (1913, 2013), and recently Murre and Dros (2015), studied the hypothesis of the exponential nature of forgetting in humans. Three major indicators were identified that affect memory retention in humans: delay since last review of learning materials and strength of human memory (Ebbinghaus, 1913; Dempster, 1989; Wixted, 1990; Cepeda et al., 2006; Novikoff et al., 2012), and, more recently, difficulty of learning materials (Reddy et al., 2016).

The above findings show that human learners can learn efficiently and effectively by increasing intervals of time between subsequent reviews of previously learned materials (spaced repetition). In (Amiri et al., 2017), we built on these findings to develop efficient and effective training paradigms for neural networks. Previous research also investigated the development of cognitively-motivated training paradigms named curriculum learning for artificial neural networks (Bengio et al., 2009; Kumar et al., 2010). The difference between the above models is in their views to learning: curriculum learning is inspired by the learning principle that training starts with *easier* concepts and gradually proceeds with more difficult ones (Bengio et al., 2009). On the other hand, spaced repetition models are inspired by the learning principle that effective and efficient learning can be achieved by working more on difficult concepts and less on easier ones.

In this research, we extend our spaced repetition training paradigms to simultaneously train artificial neural networks and identify training instances with potentially wrong labels (spurious instances) in datasets. Our work is important because spurious instances may inversely affect interpretations of the data, models developed from the data, and decisions made based on the data. Furthermore, spurious instances lead to unrealistic comparison among competing systems if they exist in test data.

5 Conclusion and Future Work

We present a novel approach based on queueing theory and psychology of learning to identify spurious instances in datasets. Our approach can be considered as a scheduler that iteratively trains a downstream learner (e.g. a neural network) and detects spurious instances with respect to their difficulty to learn during the training process. Our approach is robust and can be applied to any dataset without modification given a neural network designed for the target task of the dataset.

Our work can be extended by: (a) utilizing several predictions for each training instance, (b) investigating the extent to which a more sophisticated and effective downstream learner can affect the performance of different spotters, (c) developing models to better distinguish hard genuine instances from spurious ones, and (d) developing ranking algorithms to improve the performance of models on real-world datasets.

Acknowledgments

We thank anonymous reviewers for their thoughtful comments. This work was supported by National Institutes of Health (NIH) grant R01GM114355 from the National Institute of General Medical Sciences (NIGMS). The content is solely the responsibility of the authors and does not represent the official views of the NIH.

References

Hadi Amiri, Timothy Miller, and Guergana Savova. 2017. Repeat before forgetting: Spaced repetition for efficient and effective training of neural networks. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, Copenhagen, Denmark, pages 2401–2410.

Sumit Basu and Janara Christensen. 2013. Teaching classification boundaries to humans. In *Proceedings of the Twenty-Seventh AAAI Conference on Artificial Intelligence*. AAAI Press, pages 109–115.

Yoshua Bengio, Jérôme Louradour, Ronan Collobert, and Jason Weston. 2009. Curriculum learning. In *Proceedings of the 26th annual international conference on machine learning*. pages 41–48.

Nicholas J Cepeda, Harold Pashler, Edward Vul, John T Wixted, and Doug Rohrer. 2006. Distributed practice in verbal recall tasks: A review and quantitative synthesis. *Psychological bulletin* 132(3):354–380.

Frank N Dempster. 1989. Spacing effects and their implications for theory and practice. *Educational Psychology Review* 1(4):309–330.

Markus Dickinson and W Detmar Meurers. 2003. Detecting errors in part-of-speech annotation. In *Proceedings of the tenth conference on European chapter of the Association for Computational Linguistics-Volume 1*. pages 107–114.

Hermann Ebbinghaus. 1913. *Memory: A contribution to experimental psychology*. 3. University Microfilms.

Hermann Ebbinghaus. 2013. Memory: A contribution to experimental psychology. *Annals of neurosciences* 20(4):155.

Eleazar Eskin. 2000. Detecting errors within a corpus using anomaly detection. In *Proceedings of North American chapter of the Association for Computational Linguistics conference*. pages 148–153.

Reuben Feinman, Ryan R Curtin, Saurabh Shintre, and Andrew B Gardner. 2017. Detecting adversarial samples from artifacts. *arXiv preprint arXiv:1703.00410* .

Joseph L Fleiss. 1971. Measuring nominal scale agreement among many raters. *Psychological bulletin* 76(5):378.

Yarin Gal and Zoubin Ghahramani. 2016. Dropout as a bayesian approximation: Representing model uncertainty in deep learning. In *international conference on machine learning*. pages 1050–1059.

Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. 2015. Explaining and harnessing adversarial examples. *International Conference on Learning Representations* .

Melody Y. Guan, Varun Gulshan, Andrew M. Dai, and Geoffrey E. Hinton. 2017. Who said what: Modeling individual labelers improves classification. *CoRR* abs/1703.08774.

Aakar Gupta, William Thies, Edward Cutrell, and Ravin Balakrishnan. 2012. mclerk: enabling mobile crowdsourcing in developing regions. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. ACM, pages 1843–1852.

Haibo He and Edwardo A Garcia. 2009. Learning from imbalanced data. *IEEE Transactions on knowledge and data engineering* 21(9):1263–1284.

Jan Hendrik Metzen, Tim Genewein, Volker Fischer, and Bastian Bischoff. 2017. On detecting adversarial perturbations. *International Conference on Learning Representations* .

Dan Hendrycks and Kevin Gimpel. 2016. A baseline for detecting misclassified and out-of-distribution examples in neural networks. *International Conference on Learning Representations* .

Dirk Hovy, Taylor Berg-Kirkpatrick, Ashish Vaswani, and Eduard Hovy. 2013. 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*. Association for Computational Linguistics, Atlanta, Georgia, pages 1120–1130.

Lu Jiang, Deyu Meng, Shoou-I Yu, Zhenzhong Lan, Shiguang Shan, and Alexander Hauptmann. 2014. Self-paced learning with diversity. In *Advances in Neural Information Processing Systems 27*, pages 2078–2086.

Lu Jiang, Deyu Meng, Qian Zhao, Shiguang Shan, and Alexander G. Hauptmann. 2015. Self-paced curriculum learning. In *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence*. AAAI'15, pages 2694–2700.

Armand Joulin, Edouard Grave, and Piotr Bojanowski Tomas Mikolov. 2017. Bag of tricks for efficient text classification. *Proceedings of the 15th Eropean Chapter of the Association for Computational Linguistics* page 427.

Alex Krizhevsky. 2009. Learning multiple layers of features from tiny images. *Technical Report, University of Toronto* .

M Pawan Kumar, Benjamin Packer, and Daphne Koller. 2010. Self-paced learning for latent variable models. In *Advances in Neural Information Processing Systems*. pages 1189–1197.

Walter S Lasecki, Christopher D Miller, and Jeffrey P Bigham. 2013. Warping time for more effective real-time crowdsourcing. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. ACM, pages 2033–2036.

Sebastian Leitner. 1974. *So lernt man lernen: der Weg zum Erfolg (How to learn to learn)*. Herder.

Hrafn Loftsson. 2009. Correcting a pos-tagged corpus using three complementary methods. In *Proceedings of the 12th Conference of the European Chapter of the Association for Computational Linguistics*. pages 523–531.

Jaap MJ Murre and Joeri Dros. 2015. Replication and analysis of ebbinghaus' forgetting curve. *PloS one* 10(7):e0120644.

Nagarajan Natarajan, Inderjit S Dhillon, Pradeep K Ravikumar, and Ambuj Tewari. 2013. Learning with noisy labels. In *Advances in neural information processing systems*. pages 1196–1204.

Timothy P Novikoff, Jon M Kleinberg, and Steven H Strogatz. 2012. Education of a model student. *Proceedings of the National Academy of Sciences* 109(6):1868–1873.

Siddharth Reddy, Igor Labutov, Siddhartha Banerjee, and Thorsten Joachims. 2016. Unbounded human learning: Optimal scheduling for spaced repetition. In *Proceedings of SIGKDD*. pages 1815–1824.

Ines Rehbein and Josef Ruppenhofer. 2017. Detecting annotation noise in automatically labelled data. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, pages 1160–1170.

Valentin I Spitkovsky, Hiyan Alshawi, and Daniel Jurafsky. 2010. From baby steps to leapfrog: How less is more in unsupervised dependency parsing. In *North American Chapter of the Association for Computational Linguistics*. pages 751–759.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. In *Proceedings of Advances in neural information processing systems*. pages 3104–3112.

Jason Weston, Antoine Bordes, Sumit Chopra, Alexander M Rush, Bart van Merriënboer, Armand Joulin, and Tomas Mikolov. 2016. Towards ai-complete question answering: A set of prerequisite toy tasks. *International Conference on Learning Representations* .

John T Wixted. 1990. Analyzing the empirical course of forgetting. *Journal of Experimental Psychology: Learning, Memory, and Cognition* 16(5):927.

Tong Xiao, Tian Xia, Yi Yang, Chang Huang, and Xiaogang Wang. 2015. Learning from massive noisy labeled data for image classification. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. pages 2691–2699.

Wojciech Zaremba and Ilya Sutskever. 2014. Learning to execute. *arXiv preprint arXiv:1410.4615* .

Stephan Zheng, Yang Song, Thomas Leung, and Ian Goodfellow. 2016. Improving the robustness of deep neural networks via stability training. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. pages 4480–4488.

Xingquan Zhu and Xindong Wu. 2004. Class noise vs. attribute noise: A quantitative study. *Artificial Intelligence Review* 22(3):177–210.

Xingquan Zhu, Xindong Wu, and Qijun Chen. 2003. Eliminating class noise in large datasets. In *Proceedings of the Twentieth International Conference on International Conference on Machine Learning*. AAAI Press, ICML'03, pages 920–927.