Label Noise in Context

Michael Desmond[*]
mdesmond@us.ibm.com

Catherine Finegan-Dollak[*]
cfd@ibm.com

Jeff Boston
daddyb@us.ibm.com

Matthew Arnold
marnold@us.ibm.com

IBM Research AI
1101 Kitchawan Rd, Yorktown Heights, NY 10598, USA

Abstract

Label noise—incorrectly or ambiguously labeled training examples—can negatively impact model performance. Although noise detection techniques have been around for decades, practitioners rarely apply them, as manual noise remediation is a tedious process. Examples incorrectly flagged as noise waste reviewers' time, and correcting label noise without guidance can be difficult.

We propose LNIC, a noise-detection method that uses an example's neighborhood within the training set to (a) reduce false positives and (b) provide an explanation as to why the example was flagged as noise. We demonstrate on several short-text classification datasets that LNIC outperforms the state of the art on measures of precision and $F_{0.5}$-score. We also show how LNIC's training set context helps a reviewer to understand and correct label noise in a dataset. The LNIC tool lowers the barriers to label noise remediation, increasing its utility for NLP practitioners.

1 Introduction

Label noise—examples with incorrect or ambiguous labels in a training set—degrades the performance of the learned model, resulting in inaccurate predictions (Frénay and Verleysen, 2014). Automated data collection risks generating noisy datasets, and human annotators may introduce noise through a lack of attention or expertise.

Automatic noise-detection algorithms analyze a training set and flag "suspicious" examples that are likely mislabeled (Brodley and Friedl, 1999; Frénay and Verleysen, 2014). Suspicious examples can be deleted, automatically corrected by an algorithm, or reviewed by a human. Human review is the most effective of these mitigation options but is comparatively expensive.

[*]The first two authors contributed equally.

sports	fitness
⇒ **Unexpected increase in running ability** • Is it possible for the libero to score points in volleyball? • How counter-productive would having two coaches be?	• Why doesn't my stamina seem to improve? • Is there a rule of thumb for setting running goals?

Table 1: Training set context can help an annotator decide if the highlighted suspicious training example is correctly labeled *sports* or should be labeled *fitness*.

Two problems contribute to making human review time consuming: false positives and a lack of explanation. False positives are examples that are incorrectly flagged as noise; reviewing such examples wastes the annotator's time. Showing a reviewer a suspicious example without an explanation is effective in the simplest cases, but is likely to cause difficulty and frustration in the more common case of non-obvious noise that requires a deeper comprehension of the data.

To date, few noise-detection algorithms have been designed with human review in mind. Sluban et al. (2010) is the only work we are aware of that recognized that a noise-detection algorithm for use in a human review process should emphasize precision (*i.e.*, reduce the proportion of false positives). However, we are unaware of any existing work that addresses the explainability of detected label noise.

We propose the Label Noise in Context system, or LNIC, which uses the neighborhood surrounding a suspicious example in the training set to improve both precision and explainability. By calculating a similarity matrix for the dataset, we are able to identify a suspicious example's neighborhood and use a method similar to a nearest-neighbors classifier to filter out false positives. Applying a set of simple heuristics to the same similarity matrix allows us to construct a *training set context*, like that in Table 1. Seen in isolation, an example about

Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, pages 157–186
July 5 - July 10, 2020.

running ability labeled as belonging to the *sports* class is not obviously wrong; however, once the annotator understands that she is seeing it because there are more similar examples in the *fitness* class, it becomes apparent that there is a better label.

The main contributions of this work are

- We describe LNIC's nearest-neighbors-based algorithm to improve precision and explainability of automatically detected label noise (Sec. 3).
- We show that neighborhood-based filtering after noise-detection improves precision and $F_{0.5}$ over the state of the art for five short-text classification datasets (Sec. 4 and 5).
- We present the LNIC tool for reviewing noise in context, demonstrating the value of explanations for understanding and fixing label noise (Sec. 6).

A demo video is available at `https://www.youtube.com/watch?v=20cigQaCc_k`, and a live web demo is at `http://lnic.mybluemix.net/`

2 Related Work

Noise Detection. Frénay and Verleysen (2014) conducted a comprehensive survey of the various approaches to detecting and remediating label noise. Many works advocate removing label noise to improve model performance (Brodley and Friedl, 1999; Sánchez et al., 2003; Smith and Martinez, 2011). Teng (2000) advocates automatic relabeling, while others present the case for human-in-the-loop (Ekambaram et al., 2016; Fefilatyev et al., 2012; Matic et al., 1992; Sluban et al., 2010) and hybrid techniques (Miranda et al., 2009). In work contemporaneous with ours, Northcutt et al. (2019) remove examples where a classifier's confidence is low.

The most directly related work is Brodley and Friedl (1999), describing a noise detection method using predictions from an ensemble of classifiers, and Sluban et al. (2010), proposing the High Agreement Random Forest (HARF) system; both systems are described in detail in Section 3.1.

Brodley and Friedl (1999) dropped suspicious examples but propose correction instead as future work. Sluban et al. (2010) note that precision of noise-detection is important when a human will review all suspicious examples. Garcia et al. (2016)'s experiments show that HARF also achieved state-of-the-art F_1 scores on a variety of datasets.

Active Learning Similar to label noise remediation, active learning (Settles, 2014) seeks to minimize the effort a human needs to expend on data labeling activities in order to improve model performance. However, active learning aims to select the most informative *unlabeled* data to label next, while label noise detection identifies *already-labeled* data that may require additional labeling effort. We consider active learning and label noise detection as complimentary technologies, that might be woven together within a robust model improvement flow.

At a technical level, some active learning and label noise detection techniques are based on similar foundations. Query By Committee (QBC) (Seung et al., 1992) active learning uses an ensemble of classifiers, selecting examples on which the ensemble *disagrees* for labeling. Similarly ensemble-based noise detection algorithms select examples where the ensemble *agrees* (but disagrees with the given label). Model uncertainty, which underpins many effective active learning strategies such as *least confident*, *margin*, and *entropy*, is also the basis of label noise detection methods such as cleanlab (Northcutt et al., 2019).

Explainability. With the rise of increasingly complex classification models, explaining classifier predictions has received a great deal of attention. Perhaps the most well-known system is LIME (Ribeiro et al., 2016). The LIME authors noted that explaining classifier predictions increases human trust and provides insights that can be used to improve the model. To explain a classifier's prediction on a particular example, the algorithm collects nearby examples and the model's predictions for them. It trains a linear model on a simpler representation of this data, allowing it to indicate which words or super-pixels are important in the classifier's decision.

Numerous recent works in NLP and machine learning emphasize explainability. Dhurandhar et al. (2018) explained classifier predictions with positive features that push an example towards its assigned class and negative features whose absence prevent an example from being placed in a different class. Lei et al. (2016) jointly trained a generator and an encoder in order to generate rationales for sentiment prediction and a similar-question-retrieval task. Mullenbach et al. (2018) used a convolutional neural network to predict codes describing the diagnosis and treatment of patients given the clinical notes on the patent encounter. Their attention mechanism not only improved the system's precision and F_1, but also highlighted the text that

was most relevant to each code. Chiyah Garcia et al. (2018)'s system used an expert-generated decision tree and a set of templates to generate natural language explanations of what an autonomous underwater vehicle was doing and why.

Despite the interest in explainable models, no work that we are aware of has attempted to make detected label noise explainable.

3 Algorithms

LNIC uses a three-step process. First, a noise-detection algorithm flags suspicious examples. Second, a neighborhood-based filter decides which of these examples to ignore and which to flag for human review. Finally, we generate a context, using rules to select neighbors to present to the user.

3.1 Noise-Detection Algorithms

LNIC's noise-detection phase can use any noise-detection algorithm. Here, we report on three ensemble algorithms derived from the literature: consensus (Brodley and Friedl, 1999), agreed correction, and HARF (Sluban et al., 2010).[1]

Ensemble noise detection algorithms train several classifiers on cross-validation splits of the train set. Each classifier predicts labels for the left-out examples. The predicted label is the classifier's "vote" for that example. If it matches the current label, the classifier voted that the example is *not* suspicious; otherwise, the classifier voted that it is. In Brodley and Friedl (1999)'s consensus algorithm, if all votes agree that an example is suspicious, the algorithm flags that example as suspicious. Our agreed correction variant requires all votes from the ensemble to agree not only that an example is mislabeled, but also on what the correct label would be. HARF (Sluban et al., 2010) relies on the fact that a random forest is an ensemble of decision trees; it flags an example as suspicious if a super-majority of trees vote that it is.

3.2 Neighborhood Filtering

Neighborhood filtering reduces the number of examples that are incorrectly flagged as noise. If a majority of neighbors of an example have the same label as that example, it suggests that the example is correctly labeled, so LNIC filters it out of the list of suspicious examples.

The neighborhood filter calculates the pairwise cosine similarity of all examples in the training data, then finds the k neighbors closest to each suspicious example s, where k is a tunable hyperparameter. If s's current label y_c is also the most common among those neighbors, s is filtered from the pool of suspicious examples as a false positive, otherwise s is flagged for human review.[2]

LNIC supports filtering on the *feature neighborhood* or the *activation neighborhood*. The feature neighborhood represents each example using its original feature vector (here, USE embeddings (Cer et al., 2018)). The activation neighborhood represents each example in the training set using final layer activations from a neural classifier trained on the entire data set, the idea being to project training examples into a classification space.

3.3 Context Generation

The final step of the LNIC algorithm is to apply heuristics to the neighborhood to generate a training set context. This context acts as an explanation, showing (a) which classes the noise-detection ensemble proposed as a better label for the suspicious example, and (b) the most similar examples from the current class and those proposed classes.

The ensembles in the noise-detection algorithms generate a list of predicted labels for each suspicious example. These labels plus the example's current label comprise the *permitted labels* for that example. The heuristic selects the example from each permitted label that is closest to the suspicious example. If there are fewer than k permitted labels (where k is the desired context size), the balance of the context is filled out by selecting the remaining $k - n$ nearest neighbors from the permitted labels.

We build the explanation based on both the activation neighborhood and the feature neighborhood; an example that already appears in the activation context is omitted from the feature context and replaced by the next-nearest neighbor. Figures 4 and 5 show a examples of this contextual explanation.

4 Experiments

We hypothesize that adding a neighborhood-based filter after noise detection reduces the rates of false positives while retaining true noisy examples. We test this by injecting noise into datasets, running algorithms over them, and measuring the correctly and incorrectly flagged suspicious examples.

[1] Models and hyperparameters are listed in Appendix A

[2] When using raw features, this filter acts like a k-nearest neighbors classifier with veto power over the ensemble. Experiments with a vote by weighted cosine similarity correlated closely with this simpler technique, and we did not pursue it.

4.1 Datasets

We evaluate on the short-text classification datasets listed in Table 2.[3] Phase one of the evaluation introduces label noise—effectively "corrupting" the datasets. The amount of introduced label noise was controlled by an error-rate parameter, interpreted as the fraction of the training set to mislabel.

We used two strategies to introduce label noise: *random* and *next-best*. Both selected a random sample of the training data to mislabel. The random strategy assigned a random incorrect label to each selected example. The next-best strategy assigned the "next-best" incorrect label, as predicted by a classifier trained on the entire train set; this simulates a best effort but incorrect labeling, as might be performed by a confused human labeler.

4.2 Metrics

Because the goal of the algorithm is to avoid wasting human time, our evaluation should heavily punish false positives. We therefore measure the precision of each algorithm. We also follow Sluban et al. (2010) in reporting $F_{0.5}$, an F-score that values precision twice as much as recall.

$$F_{0.5} = (1 + 0.5^2)\frac{precision \cdot recall}{(0.5^2 \cdot precision) + recall} \quad (1)$$

Not every situation calls for precision to be valued twice as much as recall. Therefore, we also report F_β (Rijsbergen, 1979) for $\beta \in \{1.0, 0.2, 0.1\}$ to reflect the preferences of users who value precision and recall equally, precision five times more than recall, and precision ten times more.

5 Results

Figure 1 shows average precision and $F_{0.5}$ scores across the five datasets, and Table 3 further summarizes by averaging across error rates. Appendix B shows results split by dataset and error rate.

Table 3 shows that, averaged across datasets and error rates, adding neighborhood filtering of any kind improves precision of all of the underlying algorithms. For randomly generated noise, this is true for $F_{0.5}$ as well. Figure 1a also shows that the neighborhood activation filter gives a large boost to precision over all three noise-detection algorithms, and the feature neighborhood filter gives a smaller but still observable benefit. For next-best noise, adding the feature neighborhood filtering improves $F_{0.5}$, but activation neighborhood filtering slightly worsens $F_{0.5}$. From the graph in Figure 1d, it is apparent that activation neighborhood filtering has a benefit to $F_{0.5}$ at low error rates but declines relative to the other systems as the error rate increases, crossing at error rates near 15%. Addition of too much next-best noise negatively impacts the neural network trained on the uncorrected data, distorting the activation space. While this distortion does not harm precision, it is detrimental to recall.

For both random and next-best noise, agreed correction with activation neighborhood filtering achieves the best average precision. For random noise, HARF with activation-neighborhood filtering gives the best $F_{0.5}$ across noise rates. However, for next-best noise, HARF suffered a dramatic loss in recall when error rates exceeded about 12% (Figure 1d), leading it to have low overall $F_{0.5}$. This may be due to the random forest's use of bagging: if a subset of trees trains on samples with a great deal of non-random noise, those trees could learn to misclassify systematically. Agreed correction with feature neighborhood filtering gave the highest average $F_{0.5}$ for next-best noise.

The upward trend in precision as error rates increase suggests that the same core of false positives are consistently detected. As the number of true positives increases with higher error rates, the core of false positives makes up a smaller fraction of the total number of examples flagged as suspicious.

Table 4 lists F_β scores. As expected, using a neighborhood filter, which reduces the number of suspicious examples shown to a user, is particularly advantageous when precision is valued more than recall ($F_{0.2}$ and $F_{0.1}$), but often extracts a cost when recall and precision are equally important ($F_{1.0}$). Thus, agreed correction with no neighborhood filter is the best system to optimize $F_{1.0}$ when using next-best noise. Nevertheless, the strongest system for $F_{1.0}$ on random noise is still HARF with activation neighborhood filtering, followed closely by consensus with activation neighborhood filtering.

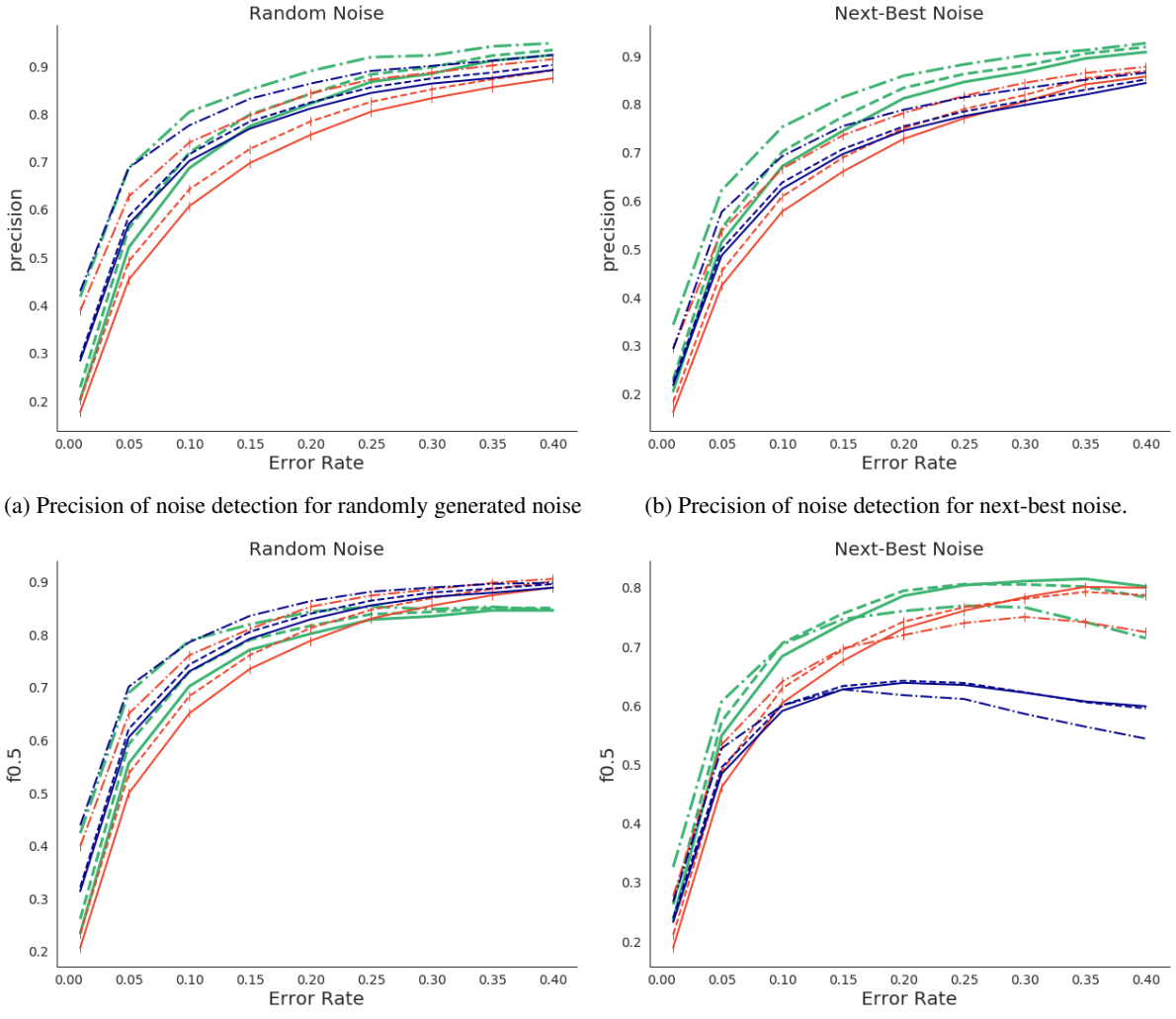
6 The LNIC Tool

The LNIC tool implements the algorithms described above and provides a web interface to review label noise in context. The interface visually summarizes the overall label noise within a dataset and links to groups of suspicious examples in con-

[3] All data is publicly available. Lists of the exact subsets we used for Stack Exchange, Stack Overflow, and Jeopardy are available at https://github.com/cfd-01/LNiC_data.

	Source	Description	Train	Classes
Stack Exchange	Collected by the authors from https://archive.org/details/stackexchange	Question titles from general-interest forums classified by topic	5000	15
Stack Overflow	Subset of (Xu et al., 2015)	Question titles from programming forum classified by topic	10000	20
Jeopardy	Subset of www.j-archive.com	Gameshow question-answer pairs classified by category	5080	21
ATIS	(Price, 1990; Hakkani-Tur et al., 2016)	Questions from air travel domain classified by intent	4952	17
Snips-2017	https://github.com/snipsco/nlu-benchmark/	Requests to a digital assistant classified by intent	13784	7

Table 2: Dataset details.



(a) Precision of noise detection for randomly generated noise

(b) Precision of noise detection for next-best noise.

(c) $F_{0.5}$ of noise detection for randomly generated noise

(d) $F_{0.5}$ of noise detection for next-best noise.

Figure 1: Precision and $F_{0.5}$ at various noise levels, averaged across the five datasets.

text. LNIC's representation of the noise summary (Figure 2) is similar to a confusion matrix. In the *label noise matrix* each cell indicates the number of noisy examples discovered where the context includes the classes specified by the row and the column.[4] Clicking on a cell brings the user to a list of examples flagged as noise. Each of these examples can be expanded to show the context, as

[4]The agreed-correction algorithm guarantees that each context contains exactly two classes. When using larger contexts, the summary can be a list of class tuples.

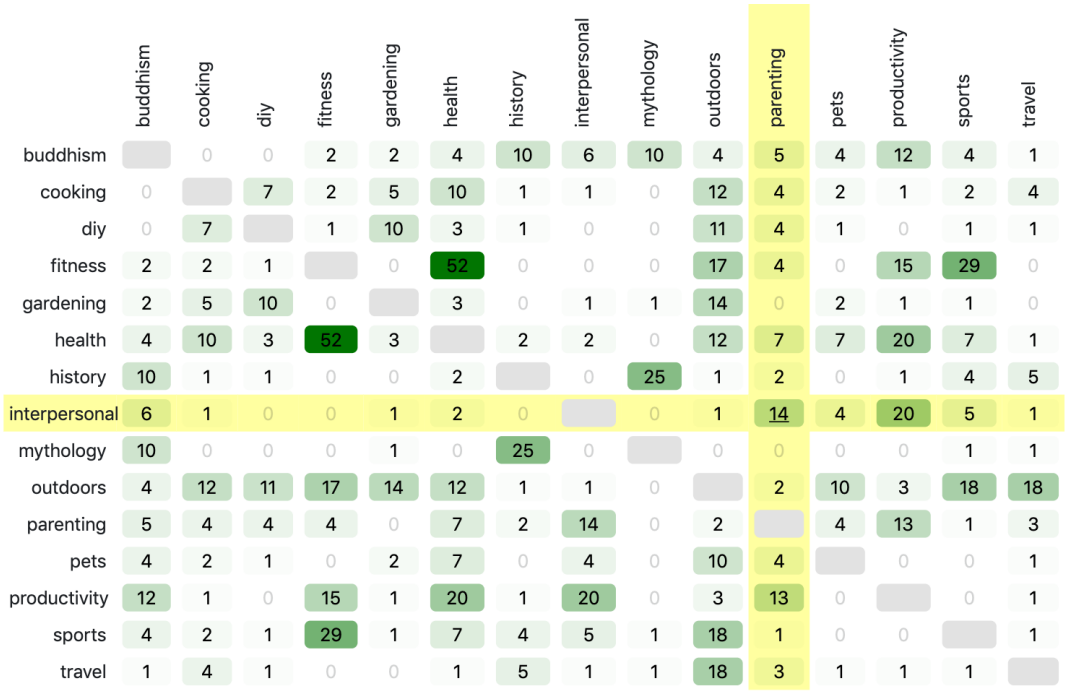


Figure 2: The label noise matrix summarizing noise discovered in approximately 30k examples from the Stack Exchange dataset.

	Random Noise		Next-Best Noise	
	P	$F_{0.5}$	P	$F_{0.5}$
Agreed correction	0.732	0.713	0.717	0.691
+ *Neighborhood filtering using*				
feature	0.753	0.729	0.738	**0.698**
activation	**0.819**	0.773	**0.779**	0.681
Consensus	0.672	0.702	0.647	0.645
+ *Neighborhood filtering using*				
feature	0.698	0.725	0.668	0.654
activation	0.774	0.781	0.713	0.646
HARF	0.734	0.751	0.667	0.559
+ *Neighborhood filtering using*				
feature	0.746	0.761	0.677	0.563
activation	0.801	**0.798**	0.718	0.549

Table 3: Mean precision and $F_{0.5}$ for the five datasets, averaged across all error rates. The top row in each section is a baseline system with no filtering.

illustrated in Figures 3 and 4.

Data from Stack Exchange illustrates how context helps a reviewer understand problems in a dataset. Sometimes, context shows that an example is mislabeled. Without context, it is easy for an annotator to be uncertain of whether a question about the existence of a myth belongs in the *history* class; it is a question about a historical civilization, after all. However, from the context in Figure 4, it is clear that even questions about the history of myths are categorized as *mythology*, and so the example's label should be changed to maintain consistency.

	Random Noise			Next-Best Noise		
	$F_{1.0}$	$F_{0.2}$	$F_{0.1}$	$F_{1.0}$	$F_{0.2}$	$F_{0.1}$
Agreed correction	0.695	0.728	0.731	**0.666**	0.712	0.716
+ *Neighborhood filtering using*						
feature	0.706	0.748	0.752	0.658	0.729	0.736
activation	0.721	**0.809**	**0.817**	0.585	**0.757**	**0.773**
Consensus	0.760	0.678	0.674	0.652	0.646	0.647
+ *Neighborhood filtering using*						
feature	0.776	0.703	0.699	0.647	0.665	0.667
activation	0.800	0.775	0.774	0.581	0.698	0.709
HARF	0.787	0.737	0.734	0.477	0.639	0.659
+ *Neighborhood filtering using*						
feature	0.794	0.748	0.746	0.477	0.647	0.669
activation	**0.803**	0.800	0.800	0.430	0.672	0.705

Table 4: Average F-scores across the datasets valuing precision to different degrees.

Other times, context can reveal more complex issues with the class structure of the data. Figure 5 shows a suspicious example from the *health* class that the noise detection algorithm suggests may belong in the *fitness* class. The context shows that in fact, both classes include questions about the timing of meals with regard to exercise. A human reviewer should make a decision about where the boundary between these two classes should lie and assign these utterances consistently to one class.

7 Conclusion

Although NLP practitioners know that label noise harms performance, and noise detection algorithms have long been available, this technology is not

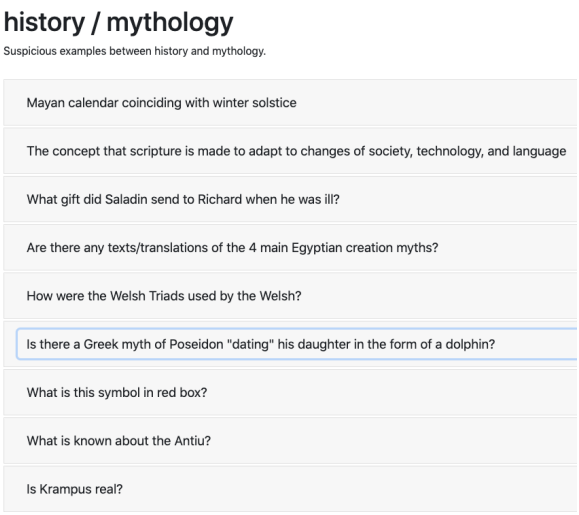


Figure 3: Suspicious examples at the intersection of *history* and *mythology* classes without context.

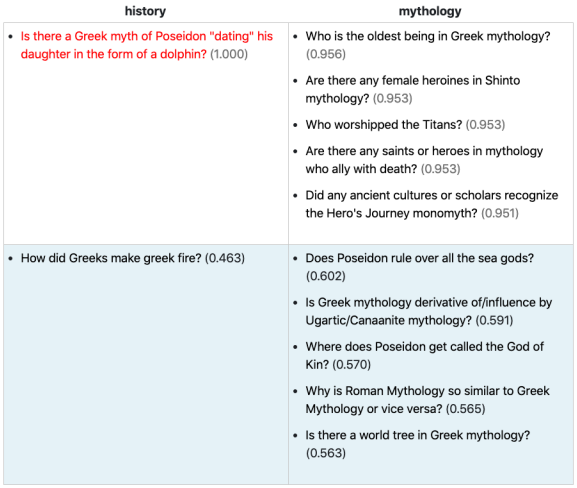


Figure 4: An example from Figure 3, with context. In red is the suspicious example. Examples in the white box are its context from activation space, and those in the blue box are context from raw embedding space. Numbers in parentheses indicate cosine similarity.

being applied in practice, perhaps because human

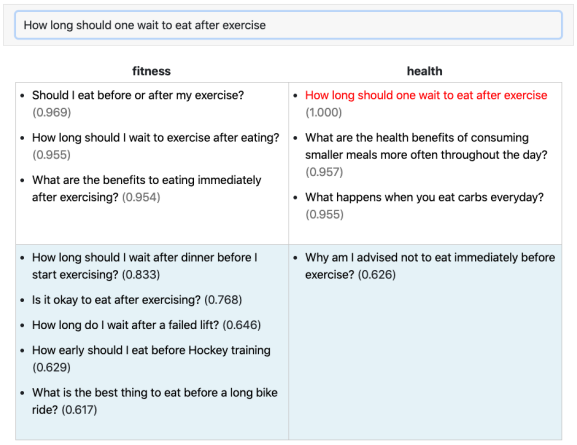


Figure 5: Context shows overlapping class definitions.

review of detected errors is difficult and time consuming. LNIC makes human review of possible label noise easier and more efficient. It reduces the number of false positive examples that the reviewer must look at, providing state-of-the-art precision and $F_{0.5}$ across several short text datasets. And by providing an explanation of why the model flagged an example as suspicious, it makes the output of label noise detectors understandable and actionable.

Acknowledgments

We thank Evelyn Duesterwald as well as the anonymous reviewers for helpful feedback.

References

Carla E. Brodley and Mark A. Friedl. 1999. Identifying mislabeled training data. *Journal of Artificial Intelligence Research*, 11:131–167.

Daniel Cer, Yinfei Yang, Sheng-yi Kong, Nan Hua, Nicole Limtiaco, Rhomni St. John, Noah Constant, Mario Guajardo-Cespedes, Steve Yuan, Chris Tar, Yun-Hsuan Sung, Brian Strope, and Ray Kurzweil. 2018. Universal sentence encoder. *arXiv preprint*, abs/1803.11175.

Francisco Javier Chiyah Garcia, David A. Robb, Xingkun Liu, Atanas Laskov, Pedro Patron, and Helen Hastie. 2018. Explainable Autonomy: A Study of Explanation Styles for Building Clear Mental Models. In *Proceedings of the 11th International Conference on Natural Language Generation*, pages 99–108, Tilburg University, The Netherlands. Association for Computational Linguistics.

Amit Dhurandhar, Pin-Yu Chen, Ronny Luss, Chun-Chen Tu, Paishun Ting, Karthikeyan Shanmugam, and Payel Das. 2018. Explanations based on the missing: Towards contrastive explanations with pertinent negatives. In *Advances in Neural Information Processing Systems 31*, pages 592–603. Curran Associates, Inc.

Rajmadhan Ekambaram, Sergiy Fefilatyev, Matthew Shreve, Kurt Kramer, Lawrence O. Hall, Dmitry B. Goldgof, and Rangachar Kasturi. 2016. Active cleaning of label noise. *Pattern Recognition*, 51:463–480.

Sergiy Fefilatyev, Matthew Shreve, Kurt Kramer, Lawrence Hall, Dmitry Goldgof, Rangachar Kasturi, Kendra Daly, Andrew Remsen, and Horst Bunke. 2012. Label-noise reduction with support vector machines. In *Proceedings of the 21st International Conference on Pattern Recognition (ICPR2012)*, pages 3504–3508. IEEE.

Benoît Frénay and Michel Verleysen. 2014. Classification in the presence of label noise: A survey. *IEEE transactions on neural networks and learning systems*, 25(5):845–869.

Luís P.F. Garcia, André C.P.L.F. de Carvalho, and Ana C. Lorena. 2016. Noise detection in the meta-learning level. *Neurocomputing*, 176:14–25.

Dilek Hakkani-Tur, Gokhan Tur, Asli Celikyilmaz, Yun-Nung Chen, Jianfeng Gao, and Ye-Yi Wang. 2016. Multi-domain joint semantic frame parsing using bi-directional RNN-LSTM. In *Proceedings of Interspeech*.

Tao Lei, Regina Barzilay, and Tommi Jaakkola. 2016. Rationalizing Neural Predictions. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 107–117, Austin, Texas. Association for Computational Linguistics.

N. Matic, I. Guyon, L. Bottou, J. Denker, and V. Vapnik. 1992. Computer aided cleaning of large databases for character recognition. In *Proceedings., 11th IAPR International Conference on Pattern Recognition. Vol. II. Conference B: Pattern Recognition Methodology and Systems*, pages 330–333. IEEE.

André L.B. Miranda, Luís Paulo F. Garcia, André C.P.L.F. Carvalho, and Ana C. Lorena. 2009. Use of classification algorithms in noise detection and elimination. In *International Conference on Hybrid Artificial Intelligence Systems*, pages 417–424. Springer.

James Mullenbach, Sarah Wiegreffe, Jon Duke, Jimeng Sun, and Jacob Eisenstein. 2018. Explainable Prediction of Medical Codes from Clinical Text. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 1101–1111, New Orleans, Louisiana. Association for Computational Linguistics.

Curtis G Northcutt, Lu Jiang, and Isaac L Chuang. 2019. Confident learning: Estimating uncertainty in dataset labels. *arXiv preprint arXiv:1911.00068*.

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. 2011. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830.

P. J. Price. 1990. Evaluation of spoken language systems: The ATIS domain. In *Proceedings of the Workshop on Speech and Natural Language*, HLT '90, pages 91–95, Stroudsburg, PA, USA. Association for Computational Linguistics.

Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. 2016. "Why Should I Trust You?" Explaining the Predictions of Any Classifier Marco. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 1135–1144, San Francisco, California, USA. ACM.

C. J. Van Rijsbergen. 1979. *Information Retrieval*, 2nd edition. Butterworth-Heinemann, Newton, MA, USA.

José Salvador Sánchez, Ricardo Barandela, Ana I. Marqués, Roberto Alejo, and Jorge Badenas. 2003. Analysis of new techniques to obtain quality training sets. *Pattern Recognition Letters*, 24(7):1015–1022.

Burr Settles. 2014. Active learning literature survey. 2010. *Computer Sciences Technical Report*, 1648.

H Sebastian Seung, Manfred Opper, and Haim Sompolinsky. 1992. Query by committee. In *Proceedings of the fifth annual workshop on Computational learning theory*, pages 287–294.

Borut Sluban, Dragan Gamberger, and Nada Lavra. 2010. Advances in class noise detection. In *Proceedings of the 19th European Conference on Artificial Intelligence*, pages 1105–1106. IOS Press.

Michael R. Smith and Tony Martinez. 2011. Improving classification accuracy by identifying and removing instances that should be misclassified. In *The 2011 International Joint Conference on Neural Networks*, pages 2690–2697. IEEE.

Choh Man Teng. 2000. Evaluating noise correction. In *Pacific Rim International Conference on Artificial Intelligence*, pages 188–198. Springer.

Jiaming Xu, Peng Wang, Guanhua Tian, Bo Xu, Jun Zhao, Fangyuan Wang, and Hongwei Hao. 2015. Short text clustering via convolutional neural networks. In *Proceedings of the 1st Workshop on Vector Space Modeling for Natural Language Processing*, pages 62–69, Denver, Colorado. Association for Computational Linguistics.

A Appendix: Model Details

For ease of replication, this appendix specifies the details of the models used in our experiments.

For consensus and agreed-only noise detection, our ensemble consisted of three classifiers from Scikit Learn (Pedregosa et al., 2011): LogisticRegression, RandomForestClassifier, and MLPClassifier. We used default parameters, except that we set MLPClassifier's `max_iter` parameter to 1000 to speed up experiments.

For HARF, we used a RandomForestClassifier model with 500 trees and required 90% agreement. Sluban et al. (2010) reported on models requiring lower levels of agreement, but preliminary testing demonstrated that 90% improved results on our datasets.

For the neighborhood filter, we set $k = 5$.

Our raw vector representation of all utterances was USE (Cer et al., 2018). The activations for activation-based filtering and context generation were generated using an MLPClassifier with `hidden_layer_sizes = [100, 512]`.

B Appendix: Detailed Results

Results were summarized in the body of the paper for conciseness. In this appendix, we present precision and $F_{0.5}$ for each of the five datasets and for each of the error rates.

Underlying Algorithm	Context Filter	error rate	precision	$F_{0.5}$
Agreed correction	(baseline)	0.01	0.203559	0.235175
		0.05	0.516995	0.551899
		0.10	0.678281	0.692148
		0.15	0.758825	0.754214
		0.20	0.815761	0.792989
		0.25	0.855683	0.815505
		0.30	0.874819	0.821995
		0.35	0.902361	0.829439
		0.40	0.914755	0.823054
	activation	0.01	0.379626	0.374314
		0.05	0.653646	0.647463
		0.10	0.777705	0.745331
		0.15	0.831916	0.782190
		0.20	0.873480	0.801011
		0.25	0.899718	0.810401
		0.30	0.910957	0.806536
		0.35	0.925808	0.796353
		0.40	0.936250	0.779562
	feature	0.01	0.228381	0.261363
		0.05	0.551033	0.582140
		0.10	0.709730	0.716853
		0.15	0.785696	0.772165
		0.20	0.836971	0.804862
		0.25	0.871780	0.821373
		0.30	0.887976	0.823381
		0.35	0.912542	0.825164
		0.40	0.925177	0.816177
	nonsuspicious	0.01	0.219619	0.252413
		0.05	0.541191	0.573870
		0.10	0.700703	0.710048
		0.15	0.777494	0.767097
		0.20	0.832716	0.802072
		0.25	0.870279	0.820193
		0.30	0.885262	0.822268
		0.35	0.909316	0.823756
		0.40	0.922156	0.814744
Consensus	(baseline)	0.01	0.169058	0.197610
		0.05	0.438177	0.479689
		0.10	0.591796	0.626953
		0.15	0.678183	0.704604
		0.20	0.741275	0.758822
		0.25	0.787061	0.794762
		0.30	0.818512	0.818545
		0.35	0.847556	0.837087
		0.40	0.865010	0.843633
	activation	0.01	0.340710	0.337163
		0.05	0.581698	0.591143
		0.10	0.702483	0.699962
		0.15	0.765191	0.752932
		0.20	0.811529	0.785388
		0.25	0.843820	0.806149
		0.30	0.864403	0.817224
		0.35	0.882211	0.818990
		0.40	0.895110	0.814209
	feature	0.01	0.191324	0.221609
		0.05	0.472302	0.511817
		0.10	0.624910	0.655624
		0.15	0.707260	0.727099
		0.20	0.766436	0.776426
		0.25	0.806752	0.805997
		0.30	0.834342	0.824318
		0.35	0.862185	0.838768
		0.40	0.879229	0.843028
	nonsuspicious	0.01	0.185901	0.215997
		0.05	0.463941	0.504450
		0.10	0.614485	0.646574
		0.15	0.701349	0.722953
		0.20	0.762975	0.773727
		0.25	0.804869	0.804780
		0.30	0.833155	0.824230
		0.35	0.859147	0.835885
		0.40	0.875436	0.839302

HARF590	(baseline)	0.01	0.250247	0.272751
		0.05	0.527276	0.544201
		0.10	0.662522	0.659799
		0.15	0.732189	0.709038
		0.20	0.777402	0.732818
		0.25	0.808914	0.744331
		0.30	0.830042	0.745807
		0.35	0.846649	0.742334
		0.40	0.867032	0.742929
	activation	0.01	0.360629	0.352368
		0.05	0.631307	0.613590
		0.10	0.733487	0.692197
		0.15	0.792436	0.730627
		0.20	0.825070	0.739711
		0.25	0.851299	0.745366
		0.30	0.865749	0.736683
		0.35	0.879965	0.729277
		0.40	0.893418	0.720200
	feature	0.01	0.256974	0.279874
		0.05	0.542058	0.557569
		0.10	0.676098	0.671038
		0.15	0.744827	0.718450
		0.20	0.788157	0.739972
		0.25	0.819442	0.750494
		0.30	0.839792	0.750249
		0.35	0.857306	0.745631
		0.40	0.876019	0.744641
	nonsuspicious	0.01	0.263552	0.286632
		0.05	0.542341	0.557493
		0.10	0.676441	0.670760
		0.15	0.744850	0.717847
		0.20	0.788948	0.739795
		0.25	0.818234	0.748751
		0.30	0.839663	0.749268
		0.35	0.856773	0.743603
		0.40	0.875930	0.743391

Table 5: ATIS, Next-Best Noise

Underlying Algorithm	Context Filter	error rate	precision	$F_{0.5}$
Agreed correction	(baseline)	0.01	0.203559	0.235175
		0.05	0.516995	0.551899
		0.10	0.678281	0.692148
		0.15	0.758825	0.754214
		0.20	0.815761	0.792989
		0.25	0.855683	0.815505
		0.30	0.874819	0.821995
		0.35	0.902361	0.829439
		0.40	0.914755	0.823054
	activation	0.01	0.379626	0.374314
		0.05	0.653646	0.647463
		0.10	0.777705	0.745331
		0.15	0.831916	0.782190
		0.20	0.873480	0.801011
		0.25	0.899718	0.810401
		0.30	0.910957	0.806536
		0.35	0.925808	0.796353
		0.40	0.936250	0.779562
	feature	0.01	0.228381	0.261363
		0.05	0.551033	0.582140
		0.10	0.709730	0.716853
		0.15	0.785696	0.772165
		0.20	0.836971	0.804862
		0.25	0.871780	0.821373
		0.30	0.887976	0.823381
		0.35	0.912542	0.825164
		0.40	0.925177	0.816177
	nonsuspicious	0.01	0.219619	0.252413
		0.05	0.541191	0.573870
		0.10	0.700703	0.710048
		0.15	0.777494	0.767097
		0.20	0.832716	0.802072
		0.25	0.870279	0.820193
		0.30	0.885262	0.822268
		0.35	0.909316	0.823756
		0.40	0.922156	0.814744

Consensus	(baseline)	0.01	0.169058	0.197610
		0.05	0.438177	0.479689
		0.10	0.591796	0.626953
		0.15	0.678183	0.704604
		0.20	0.741275	0.758822
		0.25	0.787061	0.794762
		0.30	0.818512	0.818545
		0.35	0.847556	0.837087
		0.40	0.865010	0.843633
	activation	0.01	0.340710	0.337163
		0.05	0.581698	0.591143
		0.10	0.702483	0.699962
		0.15	0.765191	0.752932
		0.20	0.811529	0.785388
		0.25	0.843820	0.806149
		0.30	0.864403	0.817224
		0.35	0.882211	0.818990
		0.40	0.895110	0.814209
	feature	0.01	0.191324	0.221609
		0.05	0.472302	0.511817
		0.10	0.624910	0.655624
		0.15	0.707260	0.727099
		0.20	0.766436	0.776426
		0.25	0.806752	0.805997
		0.30	0.834342	0.824318
		0.35	0.862185	0.838768
		0.40	0.879229	0.843028
	nonsuspicious	0.01	0.185901	0.215997
		0.05	0.463941	0.504450
		0.10	0.614485	0.646574
		0.15	0.701349	0.722953
		0.20	0.762975	0.773727
		0.25	0.804869	0.804780
		0.30	0.833155	0.824230
		0.35	0.859147	0.835885
		0.40	0.875436	0.839302
HARF590	(baseline)	0.01	0.250247	0.272751
		0.05	0.527276	0.544201
		0.10	0.662522	0.659799
		0.15	0.732189	0.709038
		0.20	0.777402	0.732818
		0.25	0.808914	0.744331
		0.30	0.830042	0.745807
		0.35	0.846649	0.742334
		0.40	0.867032	0.742929
	activation	0.01	0.360629	0.352368
		0.05	0.631307	0.613590
		0.10	0.733487	0.692197
		0.15	0.792436	0.730627
		0.20	0.825070	0.739711
		0.25	0.851299	0.745366
		0.30	0.865749	0.736683
		0.35	0.879965	0.729277
		0.40	0.893418	0.720200
	feature	0.01	0.256974	0.279874
		0.05	0.542058	0.557569
		0.10	0.676098	0.671038
		0.15	0.744827	0.718450
		0.20	0.788157	0.739972
		0.25	0.819442	0.750494
		0.30	0.839792	0.750249
		0.35	0.857306	0.745631
		0.40	0.876019	0.744641
	nonsuspicious	0.01	0.263552	0.286632
		0.05	0.542341	0.557493
		0.10	0.676441	0.670760
		0.15	0.744850	0.717847
		0.20	0.788948	0.739795
		0.25	0.818234	0.748751
		0.30	0.839663	0.749268
		0.35	0.856773	0.743603
		0.40	0.875930	0.743391

Table 6: ATIS, Random Noise

Underlying Algorithm	Context Filter	error rate	precision	$F_{0.5}$
Agreed correction	(baseline)	0.01	0.203559	0.235175
		0.05	0.516995	0.551899
		0.10	0.678281	0.692148
		0.15	0.758825	0.754214
		0.20	0.815761	0.792989
		0.25	0.855683	0.815505
		0.30	0.874819	0.821995
		0.35	0.902361	0.829439
		0.40	0.914755	0.823054
	activation	0.01	0.379626	0.374314
		0.05	0.653646	0.647463
		0.10	0.777705	0.745331
		0.15	0.831916	0.782190
		0.20	0.873480	0.801011
		0.25	0.899718	0.810401
		0.30	0.910957	0.806536
		0.35	0.925808	0.796353
		0.40	0.936250	0.779562
	feature	0.01	0.228381	0.261363
		0.05	0.551033	0.582140
		0.10	0.709730	0.716853
		0.15	0.785696	0.772165
		0.20	0.836971	0.804862
		0.25	0.871780	0.821373
		0.30	0.887976	0.823381
		0.35	0.912542	0.825164
		0.40	0.925177	0.816177
	nonsuspicious	0.01	0.219619	0.252413
		0.05	0.541191	0.573870
		0.10	0.700703	0.710048
		0.15	0.777494	0.767097
		0.20	0.832716	0.802072
		0.25	0.870279	0.820193
		0.30	0.885262	0.822268
		0.35	0.909316	0.823756
		0.40	0.922156	0.814744
Consensus	(baseline)	0.01	0.169058	0.197610
		0.05	0.438177	0.479689
		0.10	0.591796	0.626953
		0.15	0.678183	0.704604
		0.20	0.741275	0.758822
		0.25	0.787061	0.794762
		0.30	0.818512	0.818545
		0.35	0.847556	0.837087
		0.40	0.865010	0.843633
	activation	0.01	0.340710	0.337163
		0.05	0.581698	0.591143
		0.10	0.702483	0.699962
		0.15	0.765191	0.752932
		0.20	0.811529	0.785388
		0.25	0.843820	0.806149
		0.30	0.864403	0.817224
		0.35	0.882211	0.818990
		0.40	0.895110	0.814209
	feature	0.01	0.191324	0.221609
		0.05	0.472302	0.511817
		0.10	0.624910	0.655624
		0.15	0.707260	0.727099
		0.20	0.766436	0.776426
		0.25	0.806752	0.805997
		0.30	0.834342	0.824318
		0.35	0.862185	0.838768
		0.40	0.879229	0.843028
	nonsuspicious	0.01	0.185901	0.215997
		0.05	0.463941	0.504450
		0.10	0.614485	0.646574
		0.15	0.701349	0.722953
		0.20	0.762975	0.773727
		0.25	0.804869	0.804780
		0.30	0.833155	0.824230
		0.35	0.859147	0.835885
		0.40	0.875436	0.839302

HARF590	(baseline)	0.01	0.250247	0.272751
		0.05	0.527276	0.544201
		0.10	0.662522	0.659799
		0.15	0.732189	0.709038
		0.20	0.777402	0.732818
		0.25	0.808914	0.744331
		0.30	0.830042	0.745807
		0.35	0.846649	0.742334
		0.40	0.867032	0.742929
	activation	0.01	0.360629	0.352368
		0.05	0.631307	0.613590
		0.10	0.733487	0.692197
		0.15	0.792436	0.730627
		0.20	0.825070	0.739711
		0.25	0.851299	0.745366
		0.30	0.865749	0.736683
		0.35	0.879965	0.729277
		0.40	0.893418	0.720200
	feature	0.01	0.256974	0.279874
		0.05	0.542058	0.557569
		0.10	0.676098	0.671038
		0.15	0.744827	0.718450
		0.20	0.788157	0.739972
		0.25	0.819442	0.750494
		0.30	0.839792	0.750249
		0.35	0.857306	0.745631
		0.40	0.876019	0.744641
	nonsuspicious	0.01	0.263552	0.286632
		0.05	0.542341	0.557493
		0.10	0.676441	0.670760
		0.15	0.744850	0.717847
		0.20	0.788948	0.739795
		0.25	0.818234	0.748751
		0.30	0.839663	0.749268
		0.35	0.856773	0.743603
		0.40	0.875930	0.743391

Table 7: Jeopardy, Next-Best Noise

Underlying Algorithm	Context Filter	error rate	precision	$F_{0.5}$
Agreed correction	(baseline)	0.01	0.203559	0.235175
		0.05	0.516995	0.551899
		0.10	0.678281	0.692148
		0.15	0.758825	0.754214
		0.20	0.815761	0.792989
		0.25	0.855683	0.815505
		0.30	0.874819	0.821995
		0.35	0.902361	0.829439
		0.40	0.914755	0.823054
	activation	0.01	0.379626	0.374314
		0.05	0.653646	0.647463
		0.10	0.777705	0.745331
		0.15	0.831916	0.782190
		0.20	0.873480	0.801011
		0.25	0.899718	0.810401
		0.30	0.910957	0.806536
		0.35	0.925808	0.796353
		0.40	0.936250	0.779562
	feature	0.01	0.228381	0.261363
		0.05	0.551033	0.582140
		0.10	0.709730	0.716853
		0.15	0.785696	0.772165
		0.20	0.836971	0.804862
		0.25	0.871780	0.821373
		0.30	0.887976	0.823381
		0.35	0.912542	0.825164
		0.40	0.925177	0.816177
	nonsuspicious	0.01	0.219619	0.252413
		0.05	0.541191	0.573870
		0.10	0.700703	0.710048
		0.15	0.777494	0.767097
		0.20	0.832716	0.802072
		0.25	0.870279	0.820193
		0.30	0.885262	0.822268
		0.35	0.909316	0.823756
		0.40	0.922156	0.814744

Consensus	(baseline)	0.01	0.169058	0.197610
		0.05	0.438177	0.479689
		0.10	0.591796	0.626953
		0.15	0.678183	0.704604
		0.20	0.741275	0.758822
		0.25	0.787061	0.794762
		0.30	0.818512	0.818545
		0.35	0.847556	0.837087
		0.40	0.865010	0.843633
	activation	0.01	0.340710	0.337163
		0.05	0.581698	0.591143
		0.10	0.702483	0.699962
		0.15	0.765191	0.752932
		0.20	0.811529	0.785388
		0.25	0.843820	0.806149
		0.30	0.864403	0.817224
		0.35	0.882211	0.818990
		0.40	0.895110	0.814209
	feature	0.01	0.191324	0.221609
		0.05	0.472302	0.511817
		0.10	0.624910	0.655624
		0.15	0.707260	0.727099
		0.20	0.766436	0.776426
		0.25	0.806752	0.805997
		0.30	0.834342	0.824318
		0.35	0.862185	0.838768
		0.40	0.879229	0.843028
	nonsuspicious	0.01	0.185901	0.215997
		0.05	0.463941	0.504450
		0.10	0.614485	0.646574
		0.15	0.701349	0.722953
		0.20	0.762975	0.773727
		0.25	0.804869	0.804780
		0.30	0.833155	0.824230
		0.35	0.859147	0.835885
		0.40	0.875436	0.839302
HARF590	(baseline)	0.01	0.250247	0.272751
		0.05	0.527276	0.544201
		0.10	0.662522	0.659799
		0.15	0.732189	0.709038
		0.20	0.777402	0.732818
		0.25	0.808914	0.744331
		0.30	0.830042	0.745807
		0.35	0.846649	0.742334
		0.40	0.867032	0.742929
	activation	0.01	0.360629	0.352368
		0.05	0.631307	0.613590
		0.10	0.733487	0.692197
		0.15	0.792436	0.730627
		0.20	0.825070	0.739711
		0.25	0.851299	0.745366
		0.30	0.865749	0.736683
		0.35	0.879965	0.729277
		0.40	0.893418	0.720200
	feature	0.01	0.256974	0.279874
		0.05	0.542058	0.557569
		0.10	0.676098	0.671038
		0.15	0.744827	0.718450
		0.20	0.788157	0.739972
		0.25	0.819442	0.750494
		0.30	0.839792	0.750249
		0.35	0.857306	0.745631
		0.40	0.876019	0.744641
	nonsuspicious	0.01	0.263552	0.286632
		0.05	0.542341	0.557493
		0.10	0.676441	0.670760
		0.15	0.744850	0.717847
		0.20	0.788948	0.739795
		0.25	0.818234	0.748751
		0.30	0.839663	0.749268
		0.35	0.856773	0.743603
		0.40	0.875930	0.743391

Table 8: Jeopardy, Random Noise

Underlying Algorithm	Context Filter	error rate	precision	$F_{0.5}$
Agreed correction	(baseline)	0.01	0.203559	0.235175
		0.05	0.516995	0.551899
		0.10	0.678281	0.692148
		0.15	0.758825	0.754214
		0.20	0.815761	0.792989
		0.25	0.855683	0.815505
		0.30	0.874819	0.821995
		0.35	0.902361	0.829439
		0.40	0.914755	0.823054
	activation	0.01	0.379626	0.374314
		0.05	0.653646	0.647463
		0.10	0.777705	0.745331
		0.15	0.831916	0.782190
		0.20	0.873480	0.801011
		0.25	0.899718	0.810401
		0.30	0.910957	0.806536
		0.35	0.925808	0.796353
		0.40	0.936250	0.779562
	feature	0.01	0.228381	0.261363
		0.05	0.551033	0.582140
		0.10	0.709730	0.716853
		0.15	0.785696	0.772165
		0.20	0.836971	0.804862
		0.25	0.871780	0.821373
		0.30	0.887976	0.823381
		0.35	0.912542	0.825164
		0.40	0.925177	0.816177
	nonsuspicious	0.01	0.219619	0.252413
		0.05	0.541191	0.573870
		0.10	0.700703	0.710048
		0.15	0.777494	0.767097
		0.20	0.832716	0.802072
		0.25	0.870279	0.820193
		0.30	0.885262	0.822268
		0.35	0.909316	0.823756
		0.40	0.922156	0.814744
Consensus	(baseline)	0.01	0.169058	0.197610
		0.05	0.438177	0.479689
		0.10	0.591796	0.626953
		0.15	0.678183	0.704604
		0.20	0.741275	0.758822
		0.25	0.787061	0.794762
		0.30	0.818512	0.818545
		0.35	0.847556	0.837087
		0.40	0.865010	0.843633
	activation	0.01	0.340710	0.337163
		0.05	0.581698	0.591143
		0.10	0.702483	0.699962
		0.15	0.765191	0.752932
		0.20	0.811529	0.785388
		0.25	0.843820	0.806149
		0.30	0.864403	0.817224
		0.35	0.882211	0.818990
		0.40	0.895110	0.814209
	feature	0.01	0.191324	0.221609
		0.05	0.472302	0.511817
		0.10	0.624910	0.655624
		0.15	0.707260	0.727099
		0.20	0.766436	0.776426
		0.25	0.806752	0.805997
		0.30	0.834342	0.824318
		0.35	0.862185	0.838768
		0.40	0.879229	0.843028
	nonsuspicious	0.01	0.185901	0.215997
		0.05	0.463941	0.504450
		0.10	0.614485	0.646574
		0.15	0.701349	0.722953
		0.20	0.762975	0.773727
		0.25	0.804869	0.804780
		0.30	0.833155	0.824230
		0.35	0.859147	0.835885
		0.40	0.875436	0.839302

HARF590	(baseline)	0.01	0.250247	0.272751
		0.05	0.527276	0.544201
		0.10	0.662522	0.659799
		0.15	0.732189	0.709038
		0.20	0.777402	0.732818
		0.25	0.808914	0.744331
		0.30	0.830042	0.745807
		0.35	0.846649	0.742334
		0.40	0.867032	0.742929
	activation	0.01	0.360629	0.352368
		0.05	0.631307	0.613590
		0.10	0.733487	0.692197
		0.15	0.792436	0.730627
		0.20	0.825070	0.739711
		0.25	0.851299	0.745366
		0.30	0.865749	0.736683
		0.35	0.879965	0.729277
		0.40	0.893418	0.720200
	feature	0.01	0.256974	0.279874
		0.05	0.542058	0.557569
		0.10	0.676098	0.671038
		0.15	0.744827	0.718450
		0.20	0.788157	0.739972
		0.25	0.819442	0.750494
		0.30	0.839792	0.750249
		0.35	0.857306	0.745631
		0.40	0.876019	0.744641
	nonsuspicious	0.01	0.263552	0.286632
		0.05	0.542341	0.557493
		0.10	0.676441	0.670760
		0.15	0.744850	0.717847
		0.20	0.788948	0.739795
		0.25	0.818234	0.748751
		0.30	0.839663	0.749268
		0.35	0.856773	0.743603
		0.40	0.875930	0.743391

Table 9: SNIPS, Next-Best Noise

Underlying Algorithm	Context Filter	error rate	precision	$F_{0.5}$
Agreed correction	(baseline)	0.01	0.203559	0.235175
		0.05	0.516995	0.551899
		0.10	0.678281	0.692148
		0.15	0.758825	0.754214
		0.20	0.815761	0.792989
		0.25	0.855683	0.815505
		0.30	0.874819	0.821995
		0.35	0.902361	0.829439
		0.40	0.914755	0.823054
	activation	0.01	0.379626	0.374314
		0.05	0.653646	0.647463
		0.10	0.777705	0.745331
		0.15	0.831916	0.782190
		0.20	0.873480	0.801011
		0.25	0.899718	0.810401
		0.30	0.910957	0.806536
		0.35	0.925808	0.796353
		0.40	0.936250	0.779562
	feature	0.01	0.228381	0.261363
		0.05	0.551033	0.582140
		0.10	0.709730	0.716853
		0.15	0.785696	0.772165
		0.20	0.836971	0.804862
		0.25	0.871780	0.821373
		0.30	0.887976	0.823381
		0.35	0.912542	0.825164
		0.40	0.925177	0.816177
	nonsuspicious	0.01	0.219619	0.252413
		0.05	0.541191	0.573870
		0.10	0.700703	0.710048
		0.15	0.777494	0.767097
		0.20	0.832716	0.802072
		0.25	0.870279	0.820193
		0.30	0.885262	0.822268
		0.35	0.909316	0.823756
		0.40	0.922156	0.814744

Consensus	(baseline)	0.01	0.169058	0.197610
		0.05	0.438177	0.479689
		0.10	0.591796	0.626953
		0.15	0.678183	0.704604
		0.20	0.741275	0.758822
		0.25	0.787061	0.794762
		0.30	0.818512	0.818545
		0.35	0.847556	0.837087
		0.40	0.865010	0.843633
	activation	0.01	0.340710	0.337163
		0.05	0.581698	0.591143
		0.10	0.702483	0.699962
		0.15	0.765191	0.752932
		0.20	0.811529	0.785388
		0.25	0.843820	0.806149
		0.30	0.864403	0.817224
		0.35	0.882211	0.818990
		0.40	0.895110	0.814209
	feature	0.01	0.191324	0.221609
		0.05	0.472302	0.511817
		0.10	0.624910	0.655624
		0.15	0.707260	0.727099
		0.20	0.766436	0.776426
		0.25	0.806752	0.805997
		0.30	0.834342	0.824318
		0.35	0.862185	0.838768
		0.40	0.879229	0.843028
	nonsuspicious	0.01	0.185901	0.215997
		0.05	0.463941	0.504450
		0.10	0.614485	0.646574
		0.15	0.701349	0.722953
		0.20	0.762975	0.773727
		0.25	0.804869	0.804780
		0.30	0.833155	0.824230
		0.35	0.859147	0.835885
		0.40	0.875436	0.839302
HARF590	(baseline)	0.01	0.250247	0.272751
		0.05	0.527276	0.544201
		0.10	0.662522	0.659799
		0.15	0.732189	0.709038
		0.20	0.777402	0.732818
		0.25	0.808914	0.744331
		0.30	0.830042	0.745807
		0.35	0.846649	0.742334
		0.40	0.867032	0.742929
	activation	0.01	0.360629	0.352368
		0.05	0.631307	0.613590
		0.10	0.733487	0.692197
		0.15	0.792436	0.730627
		0.20	0.825070	0.739711
		0.25	0.851299	0.745366
		0.30	0.865749	0.736683
		0.35	0.879965	0.729277
		0.40	0.893418	0.720200
	feature	0.01	0.256974	0.279874
		0.05	0.542058	0.557569
		0.10	0.676098	0.671038
		0.15	0.744827	0.718450
		0.20	0.788157	0.739972
		0.25	0.819442	0.750494
		0.30	0.839792	0.750249
		0.35	0.857306	0.745631
		0.40	0.876019	0.744641
	nonsuspicious	0.01	0.263552	0.286632
		0.05	0.542341	0.557493
		0.10	0.676441	0.670760
		0.15	0.744850	0.717847
		0.20	0.788948	0.739795
		0.25	0.818234	0.748751
		0.30	0.839663	0.749268
		0.35	0.856773	0.743603
		0.40	0.875930	0.743391

Table 10: SNIPS, Random Noise

Underlying Algorithm	Context Filter	error rate	precision	$F_{0.5}$
Agreed correction	(baseline)	0.01	0.203559	0.235175
		0.05	0.516995	0.551899
		0.10	0.678281	0.692148
		0.15	0.758825	0.754214
		0.20	0.815761	0.792989
		0.25	0.855683	0.815505
		0.30	0.874819	0.821995
		0.35	0.902361	0.829439
		0.40	0.914755	0.823054
	activation	0.01	0.379626	0.374314
		0.05	0.653646	0.647463
		0.10	0.777705	0.745331
		0.15	0.831916	0.782190
		0.20	0.873480	0.801011
		0.25	0.899718	0.810401
		0.30	0.910957	0.806536
		0.35	0.925808	0.796353
		0.40	0.936250	0.779562
	feature	0.01	0.228381	0.261363
		0.05	0.551033	0.582140
		0.10	0.709730	0.716853
		0.15	0.785696	0.772165
		0.20	0.836971	0.804862
		0.25	0.871780	0.821373
		0.30	0.887976	0.823381
		0.35	0.912542	0.825164
		0.40	0.925177	0.816177
	nonsuspicious	0.01	0.219619	0.252413
		0.05	0.541191	0.573870
		0.10	0.700703	0.710048
		0.15	0.777494	0.767097
		0.20	0.832716	0.802072
		0.25	0.870279	0.820193
		0.30	0.885262	0.822268
		0.35	0.909316	0.823756
		0.40	0.922156	0.814744
Consensus	(baseline)	0.01	0.169058	0.197610
		0.05	0.438177	0.479689
		0.10	0.591796	0.626953
		0.15	0.678183	0.704604
		0.20	0.741275	0.758822
		0.25	0.787061	0.794762
		0.30	0.818512	0.818545
		0.35	0.847556	0.837087
		0.40	0.865010	0.843633
	activation	0.01	0.340710	0.337163
		0.05	0.581698	0.591143
		0.10	0.702483	0.699962
		0.15	0.765191	0.752932
		0.20	0.811529	0.785388
		0.25	0.843820	0.806149
		0.30	0.864403	0.817224
		0.35	0.882211	0.818990
		0.40	0.895110	0.814209
	feature	0.01	0.191324	0.221609
		0.05	0.472302	0.511817
		0.10	0.624910	0.655624
		0.15	0.707260	0.727099
		0.20	0.766436	0.776426
		0.25	0.806752	0.805997
		0.30	0.834342	0.824318
		0.35	0.862185	0.838768
		0.40	0.879229	0.843028
	nonsuspicious	0.01	0.185901	0.215997
		0.05	0.463941	0.504450
		0.10	0.614485	0.646574
		0.15	0.701349	0.722953
		0.20	0.762975	0.773727
		0.25	0.804869	0.804780
		0.30	0.833155	0.824230
		0.35	0.859147	0.835885
		0.40	0.875436	0.839302

HARF590	(baseline)	0.01	0.250247	0.272751
		0.05	0.527276	0.544201
		0.10	0.662522	0.659799
		0.15	0.732189	0.709038
		0.20	0.777402	0.732818
		0.25	0.808914	0.744331
		0.30	0.830042	0.745807
		0.35	0.846649	0.742334
		0.40	0.867032	0.742929
	activation	0.01	0.360629	0.352368
		0.05	0.631307	0.613590
		0.10	0.733487	0.692197
		0.15	0.792436	0.730627
		0.20	0.825070	0.739711
		0.25	0.851299	0.745366
		0.30	0.865749	0.736683
		0.35	0.879965	0.729277
		0.40	0.893418	0.720200
	feature	0.01	0.256974	0.279874
		0.05	0.542058	0.557569
		0.10	0.676098	0.671038
		0.15	0.744827	0.718450
		0.20	0.788157	0.739972
		0.25	0.819442	0.750494
		0.30	0.839792	0.750249
		0.35	0.857306	0.745631
		0.40	0.876019	0.744641
	nonsuspicious	0.01	0.263552	0.286632
		0.05	0.542341	0.557493
		0.10	0.676441	0.670760
		0.15	0.744850	0.717847
		0.20	0.788948	0.739795
		0.25	0.818234	0.748751
		0.30	0.839663	0.749268
		0.35	0.856773	0.743603
		0.40	0.875930	0.743391

Table 11: Stack Exchange, Next-Best Noise

Underlying Algorithm	Context Filter	error rate	precision	$F_{0.5}$
Agreed correction	(baseline)	0.01	0.203559	0.235175
		0.05	0.516995	0.551899
		0.10	0.678281	0.692148
		0.15	0.758825	0.754214
		0.20	0.815761	0.792989
		0.25	0.855683	0.815505
		0.30	0.874819	0.821995
		0.35	0.902361	0.829439
		0.40	0.914755	0.823054
	activation	0.01	0.379626	0.374314
		0.05	0.653646	0.647463
		0.10	0.777705	0.745331
		0.15	0.831916	0.782190
		0.20	0.873480	0.801011
		0.25	0.899718	0.810401
		0.30	0.910957	0.806536
		0.35	0.925808	0.796353
		0.40	0.936250	0.779562
	feature	0.01	0.228381	0.261363
		0.05	0.551033	0.582140
		0.10	0.709730	0.716853
		0.15	0.785696	0.772165
		0.20	0.836971	0.804862
		0.25	0.871780	0.821373
		0.30	0.887976	0.823381
		0.35	0.912542	0.825164
		0.40	0.925177	0.816177
	nonsuspicious	0.01	0.219619	0.252413
		0.05	0.541191	0.573870
		0.10	0.700703	0.710048
		0.15	0.777494	0.767097
		0.20	0.832716	0.802072
		0.25	0.870279	0.820193
		0.30	0.885262	0.822268
		0.35	0.909316	0.823756
		0.40	0.922156	0.814744

Consensus	(baseline)	0.01	0.169058	0.197610
		0.05	0.438177	0.479689
		0.10	0.591796	0.626953
		0.15	0.678183	0.704604
		0.20	0.741275	0.758822
		0.25	0.787061	0.794762
		0.30	0.818512	0.818545
		0.35	0.847556	0.837087
		0.40	0.865010	0.843633
	activation	0.01	0.340710	0.337163
		0.05	0.581698	0.591143
		0.10	0.702483	0.699962
		0.15	0.765191	0.752932
		0.20	0.811529	0.785388
		0.25	0.843820	0.806149
		0.30	0.864403	0.817224
		0.35	0.882211	0.818990
		0.40	0.895110	0.814209
	feature	0.01	0.191324	0.221609
		0.05	0.472302	0.511817
		0.10	0.624910	0.655624
		0.15	0.707260	0.727099
		0.20	0.766436	0.776426
		0.25	0.806752	0.805997
		0.30	0.834342	0.824318
		0.35	0.862185	0.838768
		0.40	0.879229	0.843028
	nonsuspicious	0.01	0.185901	0.215997
		0.05	0.463941	0.504450
		0.10	0.614485	0.646574
		0.15	0.701349	0.722953
		0.20	0.762975	0.773727
		0.25	0.804869	0.804780
		0.30	0.833155	0.824230
		0.35	0.859147	0.835885
		0.40	0.875436	0.839302
HARF590	(baseline)	0.01	0.250247	0.272751
		0.05	0.527276	0.544201
		0.10	0.662522	0.659799
		0.15	0.732189	0.709038
		0.20	0.777402	0.732818
		0.25	0.808914	0.744331
		0.30	0.830042	0.745807
		0.35	0.846649	0.742334
		0.40	0.867032	0.742929
	activation	0.01	0.360629	0.352368
		0.05	0.631307	0.613590
		0.10	0.733487	0.692197
		0.15	0.792436	0.730627
		0.20	0.825070	0.739711
		0.25	0.851299	0.745366
		0.30	0.865749	0.736683
		0.35	0.879965	0.729277
		0.40	0.893418	0.720200
	feature	0.01	0.256974	0.279874
		0.05	0.542058	0.557569
		0.10	0.676098	0.671038
		0.15	0.744827	0.718450
		0.20	0.788157	0.739972
		0.25	0.819442	0.750494
		0.30	0.839792	0.750249
		0.35	0.857306	0.745631
		0.40	0.876019	0.744641
	nonsuspicious	0.01	0.263552	0.286632
		0.05	0.542341	0.557493
		0.10	0.676441	0.670760
		0.15	0.744850	0.717847
		0.20	0.788948	0.739795
		0.25	0.818234	0.748751
		0.30	0.839663	0.749268
		0.35	0.856773	0.743603
		0.40	0.875930	0.743391

Table 12: Stack Exchange, Random Noise

Underlying Algorithm	Context Filter	error rate	precision	$F_{0.5}$
Agreed correction	(baseline)	0.01	0.203559	0.235175
		0.05	0.516995	0.551899
		0.10	0.678281	0.692148
		0.15	0.758825	0.754214
		0.20	0.815761	0.792989
		0.25	0.855683	0.815505
		0.30	0.874819	0.821995
		0.35	0.902361	0.829439
		0.40	0.914755	0.823054
	activation	0.01	0.379626	0.374314
		0.05	0.653646	0.647463
		0.10	0.777705	0.745331
		0.15	0.831916	0.782190
		0.20	0.873480	0.801011
		0.25	0.899718	0.810401
		0.30	0.910957	0.806536
		0.35	0.925808	0.796353
		0.40	0.936250	0.779562
	feature	0.01	0.228381	0.261363
		0.05	0.551033	0.582140
		0.10	0.709730	0.716853
		0.15	0.785696	0.772165
		0.20	0.836971	0.804862
		0.25	0.871780	0.821373
		0.30	0.887976	0.823381
		0.35	0.912542	0.825164
		0.40	0.925177	0.816177
	nonsuspicious	0.01	0.219619	0.252413
		0.05	0.541191	0.573870
		0.10	0.700703	0.710048
		0.15	0.777494	0.767097
		0.20	0.832716	0.802072
		0.25	0.870279	0.820193
		0.30	0.885262	0.822268
		0.35	0.909316	0.823756
		0.40	0.922156	0.814744
Consensus	(baseline)	0.01	0.169058	0.197610
		0.05	0.438177	0.479689
		0.10	0.591796	0.626953
		0.15	0.678183	0.704604
		0.20	0.741275	0.758822
		0.25	0.787061	0.794762
		0.30	0.818512	0.818545
		0.35	0.847556	0.837087
		0.40	0.865010	0.843633
	activation	0.01	0.340710	0.337163
		0.05	0.581698	0.591143
		0.10	0.702483	0.699962
		0.15	0.765191	0.752932
		0.20	0.811529	0.785388
		0.25	0.843820	0.806149
		0.30	0.864403	0.817224
		0.35	0.882211	0.818990
		0.40	0.895110	0.814209
	feature	0.01	0.191324	0.221609
		0.05	0.472302	0.511817
		0.10	0.624910	0.655624
		0.15	0.707260	0.727099
		0.20	0.766436	0.776426
		0.25	0.806752	0.805997
		0.30	0.834342	0.824318
		0.35	0.862185	0.838768
		0.40	0.879229	0.843028
	nonsuspicious	0.01	0.185901	0.215997
		0.05	0.463941	0.504450
		0.10	0.614485	0.646574
		0.15	0.701349	0.722953
		0.20	0.762975	0.773727
		0.25	0.804869	0.804780
		0.30	0.833155	0.824230
		0.35	0.859147	0.835885
		0.40	0.875436	0.839302

HARF590	(baseline)	0.01	0.250247	0.272751
		0.05	0.527276	0.544201
		0.10	0.662522	0.659799
		0.15	0.732189	0.709038
		0.20	0.777402	0.732818
		0.25	0.808914	0.744331
		0.30	0.830042	0.745807
		0.35	0.846649	0.742334
		0.40	0.867032	0.742929
	activation	0.01	0.360629	0.352368
		0.05	0.631307	0.613590
		0.10	0.733487	0.692197
		0.15	0.792436	0.730627
		0.20	0.825070	0.739711
		0.25	0.851299	0.745366
		0.30	0.865749	0.736683
		0.35	0.879965	0.729277
		0.40	0.893418	0.720200
	feature	0.01	0.256974	0.279874
		0.05	0.542058	0.557569
		0.10	0.676098	0.671038
		0.15	0.744827	0.718450
		0.20	0.788157	0.739972
		0.25	0.819442	0.750494
		0.30	0.839792	0.750249
		0.35	0.857306	0.745631
		0.40	0.876019	0.744641
	nonsuspicious	0.01	0.263552	0.286632
		0.05	0.542341	0.557493
		0.10	0.676441	0.670760
		0.15	0.744850	0.717847
		0.20	0.788948	0.739795
		0.25	0.818234	0.748751
		0.30	0.839663	0.749268
		0.35	0.856773	0.743603
		0.40	0.875930	0.743391

Table 13: Stack Overflow, Next-Best Noise

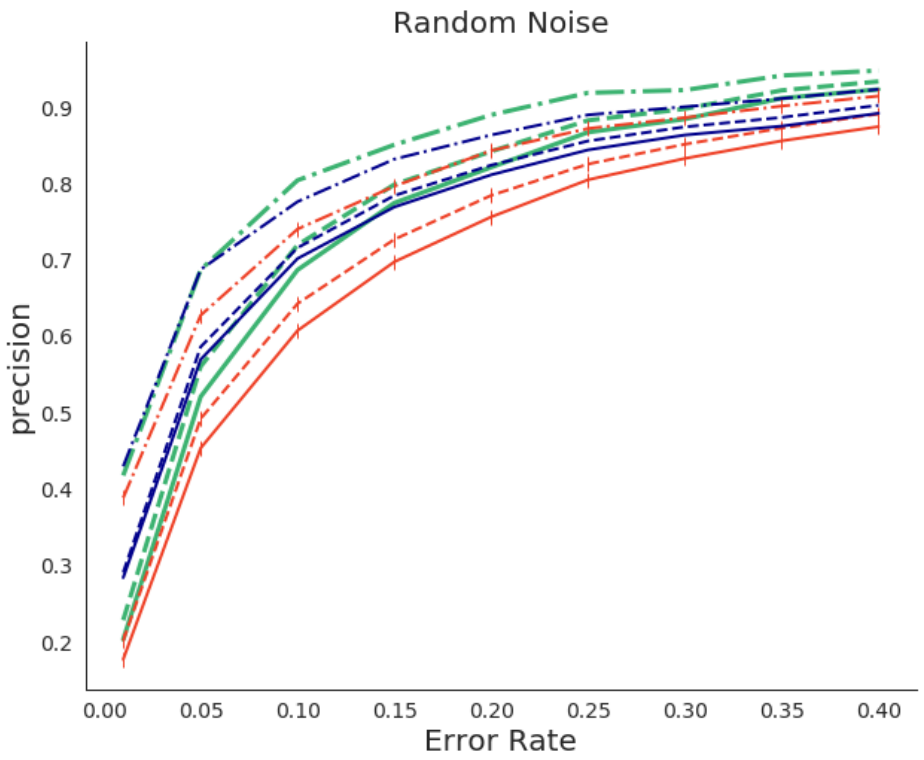
Underlying Algorithm	Context Filter	error rate	precision	$F_{0.5}$
Agreed correction	(baseline)	0.01	0.203559	0.235175
		0.05	0.516995	0.551899
		0.10	0.678281	0.692148
		0.15	0.758825	0.754214
		0.20	0.815761	0.792989
		0.25	0.855683	0.815505
		0.30	0.874819	0.821995
		0.35	0.902361	0.829439
		0.40	0.914755	0.823054
	activation	0.01	0.379626	0.374314
		0.05	0.653646	0.647463
		0.10	0.777705	0.745331
		0.15	0.831916	0.782190
		0.20	0.873480	0.801011
		0.25	0.899718	0.810401
		0.30	0.910957	0.806536
		0.35	0.925808	0.796353
		0.40	0.936250	0.779562
	feature	0.01	0.228381	0.261363
		0.05	0.551033	0.582140
		0.10	0.709730	0.716853
		0.15	0.785696	0.772165
		0.20	0.836971	0.804862
		0.25	0.871780	0.821373
		0.30	0.887976	0.823381
		0.35	0.912542	0.825164
		0.40	0.925177	0.816177
	nonsuspicious	0.01	0.219619	0.252413
		0.05	0.541191	0.573870
		0.10	0.700703	0.710048
		0.15	0.777494	0.767097
		0.20	0.832716	0.802072
		0.25	0.870279	0.820193
		0.30	0.885262	0.822268
		0.35	0.909316	0.823756
		0.40	0.922156	0.814744

Consensus	(baseline)	0.01	0.169058	0.197610
		0.05	0.438177	0.479689
		0.10	0.591796	0.626953
		0.15	0.678183	0.704604
		0.20	0.741275	0.758822
		0.25	0.787061	0.794762
		0.30	0.818512	0.818545
		0.35	0.847556	0.837087
		0.40	0.865010	0.843633
	activation	0.01	0.340710	0.337163
		0.05	0.581698	0.591143
		0.10	0.702483	0.699962
		0.15	0.765191	0.752932
		0.20	0.811529	0.785388
		0.25	0.843820	0.806149
		0.30	0.864403	0.817224
		0.35	0.882211	0.818990
		0.40	0.895110	0.814209
	feature	0.01	0.191324	0.221609
		0.05	0.472302	0.511817
		0.10	0.624910	0.655624
		0.15	0.707260	0.727099
		0.20	0.766436	0.776426
		0.25	0.806752	0.805997
		0.30	0.834342	0.824318
		0.35	0.862185	0.838768
		0.40	0.879229	0.843028
	nonsuspicious	0.01	0.185901	0.215997
		0.05	0.463941	0.504450
		0.10	0.614485	0.646574
		0.15	0.701349	0.722953
		0.20	0.762975	0.773727
		0.25	0.804869	0.804780
		0.30	0.833155	0.824230
		0.35	0.859147	0.835885
		0.40	0.875436	0.839302
HARF590	(baseline)	0.01	0.250247	0.272751
		0.05	0.527276	0.544201
		0.10	0.662522	0.659799
		0.15	0.732189	0.709038
		0.20	0.777402	0.732818
		0.25	0.808914	0.744331
		0.30	0.830042	0.745807
		0.35	0.846649	0.742334
		0.40	0.867032	0.742929
	activation	0.01	0.360629	0.352368
		0.05	0.631307	0.613590
		0.10	0.733487	0.692197
		0.15	0.792436	0.730627
		0.20	0.825070	0.739711
		0.25	0.851299	0.745366
		0.30	0.865749	0.736683
		0.35	0.879965	0.729277
		0.40	0.893418	0.720200
	feature	0.01	0.256974	0.279874
		0.05	0.542058	0.557569
		0.10	0.676098	0.671038
		0.15	0.744827	0.718450
		0.20	0.788157	0.739972
		0.25	0.819442	0.750494
		0.30	0.839792	0.750249
		0.35	0.857306	0.745631
		0.40	0.876019	0.744641
	nonsuspicious	0.01	0.263552	0.286632
		0.05	0.542341	0.557493
		0.10	0.676441	0.670760
		0.15	0.744850	0.717847
		0.20	0.788948	0.739795
		0.25	0.818234	0.748751
		0.30	0.839663	0.749268
		0.35	0.856773	0.743603
		0.40	0.875930	0.743391

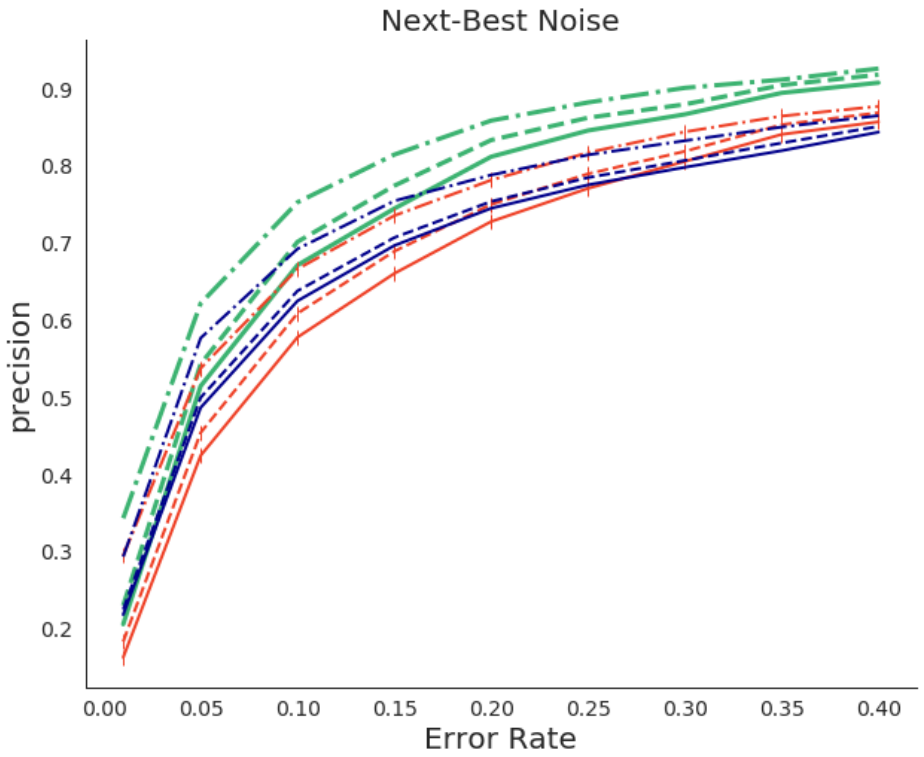
Table 14: Stack Overflow, Random Noise

C Appendix: Enlarged Figures

This appendix contains the same images as the body of the paper, enlarged to improve accessibility.

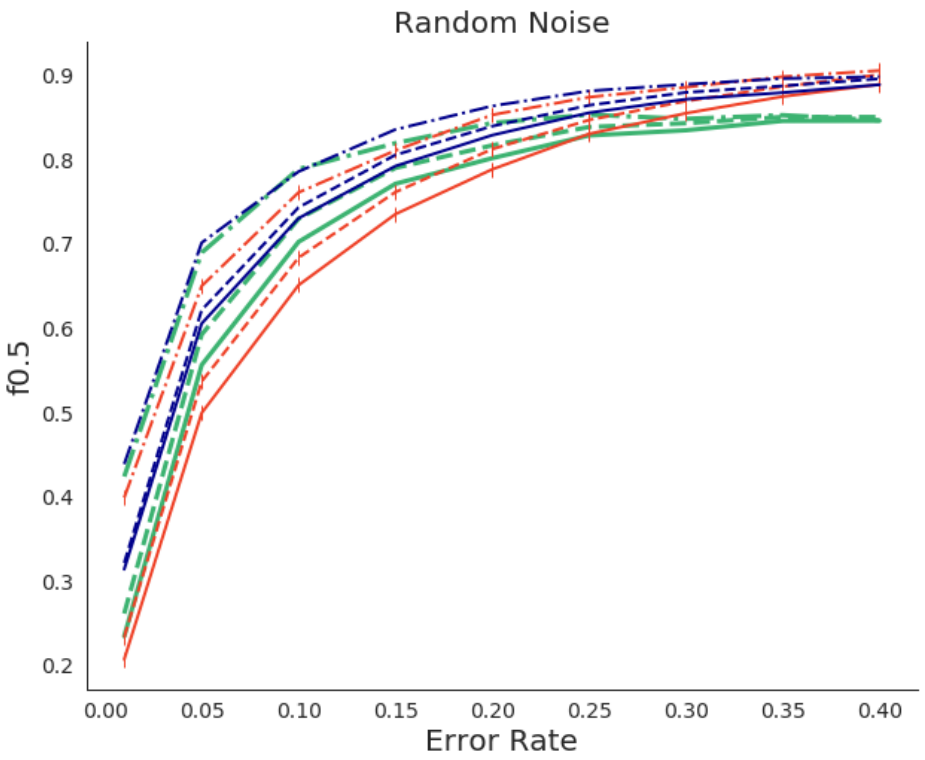


(a) Precision of noise detection for randomly generated noise

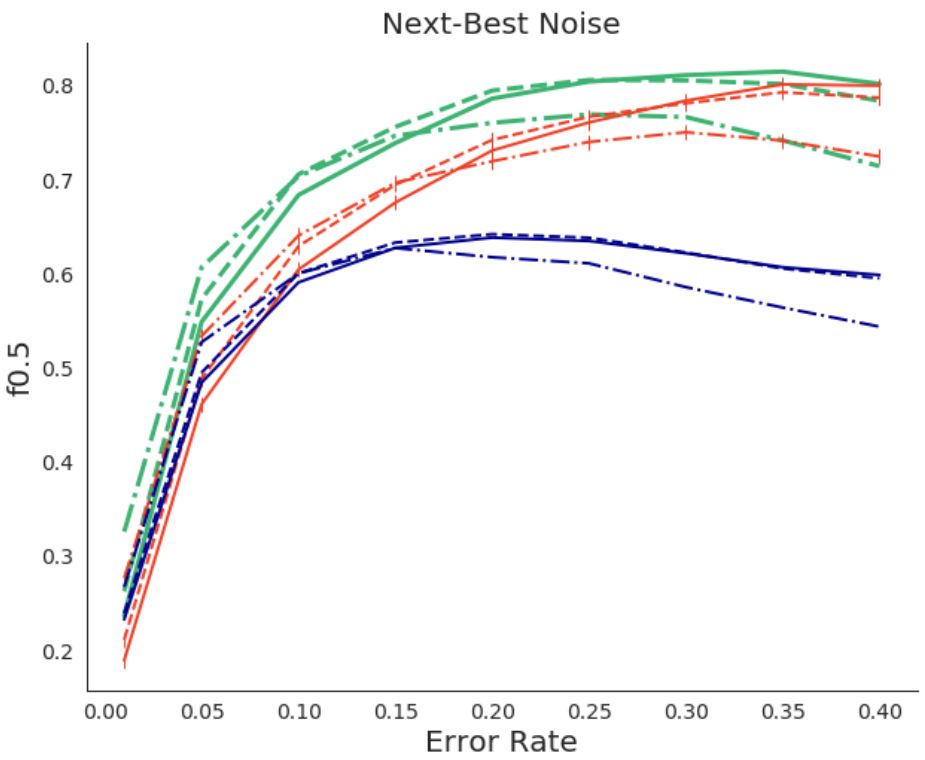


(b) Precision of noise detection for next-best noise.

Figure 6: Precision at various noise levels, averaged across the five datasets.

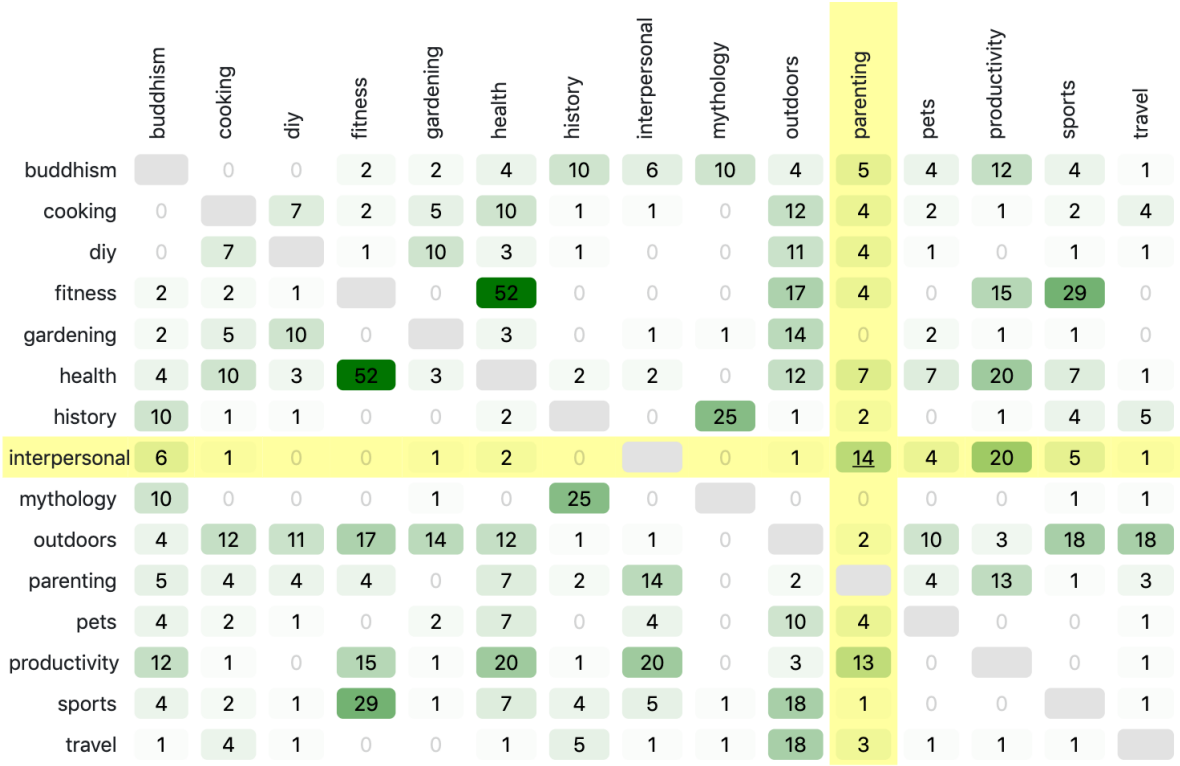


(c) $F_{0.5}$ of noise detection for randomly generated noise



(d) $F_{0.5}$ of noise detection for next-best noise.

Figure 6: $F_{0.5}$ at various noise levels, averaged across the five datasets.



	buddhism	cooking	diy	fitness	gardening	health	history	interpersonal	mythology	outdoors	parenting	pets	productivity	sports	travel
buddhism		0	0	2	2	4	10	6	10	4	5	4	12	4	1
cooking	0		7	2	5	10	1	1	0	12	4	2	1	2	4
diy	0	7		1	10	3	1	0	0	11	4	1	0	1	1
fitness	2	2	1		0	52	0	0	0	17	4	0	15	29	0
gardening	2	5	10	0		3	0	1	1	14	0	2	1	1	0
health	4	10	3	52	3		2	2	0	12	7	7	20	7	1
history	10	1	1	0	0	2		0	25	1	2	0	1	4	5
interpersonal	6	1	0	0	1	2	0		0	1	14	4	20	5	1
mythology	10	0	0	0	1	0	25	0		0	0	0	0	1	1
outdoors	4	12	11	17	14	12	1	1	0		2	10	3	18	18
parenting	5	4	4	4	0	7	2	14	0	2		4	13	1	3
pets	4	2	1	0	2	7	0	4	0	10	4		0	0	1
productivity	12	1	0	15	1	20	1	20	0	3	13	0		0	1
sports	4	2	1	29	1	7	4	5	1	18	1	0	0		1
travel	1	4	1	0	0	1	5	1	1	18	3	1	1	1	

Figure 7: A summary of noise discovered in approximately 30k examples from Stack Exchange.

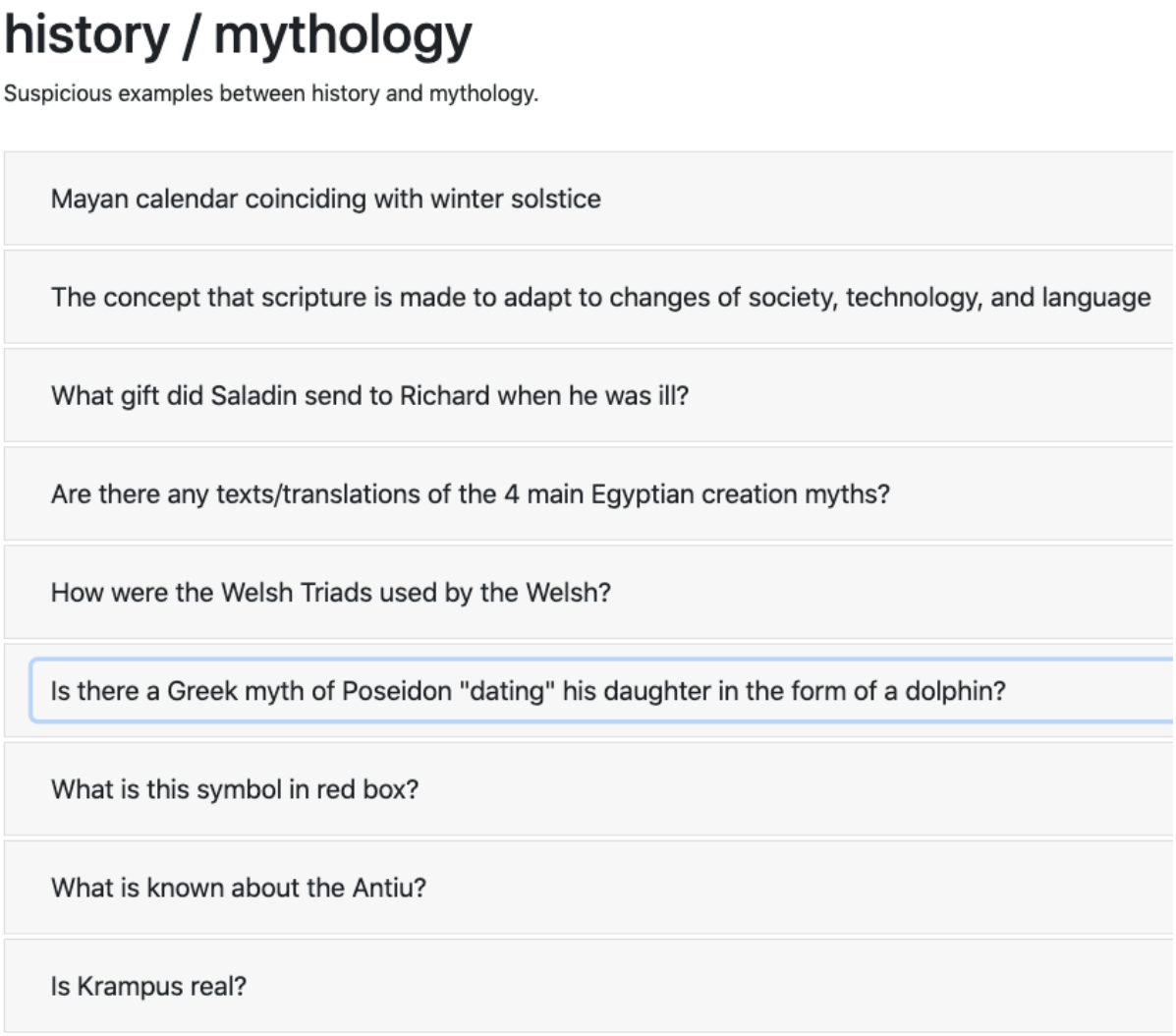


Figure 8: Suspicious examples at the intersection of *history* and *mythology* classes without context.

history	mythology
• Is there a Greek myth of Poseidon "dating" his daughter in the form of a dolphin? (1.000)	• Who is the oldest being in Greek mythology? (0.956) • Are there any female heroines in Shinto mythology? (0.953) • Who worshipped the Titans? (0.953) • Are there any saints or heroes in mythology who ally with death? (0.953) • Did any ancient cultures or scholars recognize the Hero's Journey monomyth? (0.951)
• How did Greeks make greek fire? (0.463)	• Does Poseidon rule over all the sea gods? (0.602) • Is Greek mythology derivative of/influence by Ugartic/Canaanite mythology? (0.591) • Where does Poseidon get called the God of Kin? (0.570) • Why is Roman Mythology so similar to Greek Mythology or vice versa? (0.565) • Is there a world tree in Greek mythology? (0.563)

Figure 9: An example from Figure 3, with context. In red is the suspicious example. Examples in the white box are its context from activation space, and those in the blue box are context from raw embedding space. Numbers in parentheses indicate cosine similarity.

How long should one wait to eat after exercise

fitness	health
• Should I eat before or after my exercise? (0.969) • How long should I wait to exercise after eating? (0.955) • What are the benefits to eating immediately after exercising? (0.954)	• How long should one wait to eat after exercise (1.000) • What are the health benefits of consuming smaller meals more often throughout the day? (0.957) • What happens when you eat carbs everyday? (0.955)
• How long should I wait after dinner before I start exercising? (0.833) • Is it okay to eat after exercising? (0.768) • How long do I wait after a failed lift? (0.646) • How early should I eat before Hockey training (0.629) • What is the best thing to eat before a long bike ride? (0.617)	• Why am I advised not to eat immediately before exercise? (0.626)

Figure 10: Context shows overlapping class definitions.