USENIX Security ’25 Artifact Appendix: SoK: Machine Learning for Misinformation Detection

Madelyne Xiao
Princeton University

Jonathan Mayer
Princeton University

A Artifact Appendix

A.1 Abstract

We survey literature on automated detection of misinformation across a corpus of 248 well-cited papers in the field. We then examine subsets of papers for data and code availability, design missteps, reproducibility, and generalizability. Our paper corpus includes published work in security, natural language processing, and computational social science. We demonstrate the limitations of current detection methods in a series of three representative replication studies: these replications can be found in sections 5.1, 5.2, and 5.3 of the paper, and address article-, user-, and website-scoped misinformation detection, respectively. The paper artifact contains code and datasets for reproducing the results found in these replication sections

A.2 Description & Requirements

No special setup required beyond access to recent versions of Python (3.10 or later) and IPython (7.16 or later). All Python library requirements are specified in the accompanying `requirements.txt` file in the root directory.

A.2.1 Security, privacy, and ethical concerns

We anticipate no risks to evaluators as a result of code execution.

A.2.2 How to access

For purposes of evaluation, code and data are available at `https://github.com/citp/sok_misinformation` and `https://zenodo.org/records/15613696`.

A.2.3 Hardware dependencies

None.

A.2.4 Software dependencies

Python 3.10 (or later) required to run pyscripts and a Jupyter notebook. All Python library requirements are specified in the accompanying `requirements.txt` file in the root directory.

A.2.5 Benchmarks

Datasets required for each test are described within the relevant directory in the replications repository. These include the FAKES and ISOT datasets used in the article-level replication, the troll and non-troll datasets used in the user-level replication, and the EMNLP datasets used in the website-level replication. All are included in the replications archive.

A.3 Set-up

No special setup required beyond access to recent versions of Python (3.10 or later) and IPython (7.16 or later).

A.3.1 Installation

Evaluators working from the Git repository should additionally download the `articles_data.zip` archive linked in the git repository; package requirements are included in each pyscript where necessary.

A.3.2 Basic Test

Evaluator may run `python CNN_revised.py -test_source nytimes -rand_seed 0` within the `articles` directory to observe evaluation of a small test dataset in advance of completing all 30 test cases. Output will include accuracy and error reporting.

To observe the evaluation of a small test dataset used in the website-level replication tests, evaluator may run `python3 train_artifact.py -tk fact -f articles_body_glove,articles_title_glove -sb small` within the `websites` directory. Output will include accuracy and error reporting.

A.4 Evaluation workflow

We include summaries of high-level results in the following subsection; detailed discussion of these points can be found in section 5 of the manuscript.

A.4.1 Major Claims

(C1): Article-level detection methods are susceptible to text dependencies resulting from authorship, style, and source. We demonstrate this through a series of analyses on datasets whose credible/factual texts are sourced from a reputable news agency (Reuters). We find that altering the semantic content of news articles without altering the style of those articles induces no change in the ultimate classification of those articles, suggesting that there is 1) no semantic understanding of news article contents during classification, and 2) style signals likely override any other semantic signals.

(C2): User-scoped detection tasks are frequently tautological, and proceed from known lists of troll users. These classifiers can only detect users who very closely resemble users who were manually identified as trolls, oftentimes using data not included in available training sets.

(C3): Website-scoped detection tasks purport to leverage multimodal feature sets—including infrastructural, textual, and social media data—but the performance of these methods is often overdetermined by features derived from website text semantics (and, as such, the classifier we evaluate is approximately as accurate as a text-only classifier, and is susceptible to the same dependencies we discuss in C1).

A.4.2 Experiments

Articles. See Table 2 for results of a replication analyses of the ISOT, FA-KES, and NYTimes and Reuters datasets. The original paper, which presents results on the ISOT and FA-KES datasets, can be found here: `https://www.researchgate.net/publication/348379370_Fake_news_detection_A_hybrid_CNN-RNN_based_deep_learning_approach`.

In the `articles` directory of the main replications repository, execute `./run.sh` to run 30 iterations of the model (with prespecified random seeds) on 1) ISOT and FAKES datasets used in the original paper and 2) original and modified datasets we developed for purposes of robustness testing. With multithreading (four cores, each running a separate instance of the model with the same seed and a different dataset), this script runs in about 7.5 hours.

Users. See Figure 4 for the results of a partial dependence analysis on TrollMagnifier data. The original paper can be found here: `https://arxiv.org/pdf/2112.00443`. The Jupyter notebook containing code to train and run the classifier and create these PDP plots is `trollmagnifier_pdp_artifact.ipynb`. Instructions for use are included in the notebook.

Websites. See Table 4 in Appendix D for the results of ablation analyses on an EMNLP dataset for this paper: `https://aclanthology.org/D18-1389.pdf`. To run all tests, execute the bashscript contained in the `websites` directory using `./run.sh`.

A text header appears at the start of each test case (e.g. "fact, full corpus, all features") denoting 1) the classification task (factuality or bias classification), 2) the corpus to be used (these vary by size (full, medium, small), or are stratified by bias (left, center, right), or credibility (low, mixed, high)), and 3) the feature set. Feature sets are printed to the console at runtime. Extraction code for features is included in the features directory. You can additionally inspect feature sets and function calls in the 'run.sh' bashscript.

Accuracy, MAE, and F1 scores are reported for each test. To compare these results to those found in Table 4, read each table row horizontally, matching accuracy scores to those reported on the console. Hold-one-out analyses are denoted by (-) in the feature set labels (e.g., "articles(-)" denotes *all* features excluding those relating to article contents and title). Evaluation of specific feature sets are denoted by (+) (e.g., "articles" denotes only those features derived from article contents and title). With the same hardware specs reported previously, the full script runs in about 70 minutes.

A.5 Version

Based on the LaTeX template for Artifact Evaluation V20231005. Submission, reviewing and badging methodology followed for the evaluation of this artifact can be found at `https://secartifacts.github.io/usenixsec2025/`.