ARTIFACT EVALUATED usenix ASSOCIATION AVAILABLE

ARTIFACT EVALUATED usenix ASSOCIATION FUNCTIONAL

USENIX Security '24 Artifact Appendix: CellularLint: A Systematic Approach to Identify Inconsistent Behavior in Cellular Network Specifications

Mirza Masfiqur Rahman*, Imtiaz Karim*, Elisa Bertino
Purdue University

A Artifact Appendix

A.1 Abstract

We introduce CellularLint–a semi-automatic framework for inconsistency detection within the standards of 4G and 5G, capitalizing on a suite of natural language processing techniques. Our proposed method uses a revamped few-shot learning mechanism on domain-adapted large language models. Pre-trained on a vast corpus of cellular network protocols, this method enables CellularLint to simultaneously detect inconsistencies at various levels of semantics and practical use cases. In doing so, CellularLint significantly advances the automated analysis of protocol specifications in a scalable fashion. In our investigation, we focused on the Non-Access Stratum (NAS) and the security specifications of 4G and 5G networks, ultimately uncovering 157 inconsistencies with 82.67% accuracy. After verification of these inconsistencies on 3 open-source implementations and 17 commercial devices, we confirm that they indeed have a substantial impact on design decisions, potentially leading to concerns related to privacy, integrity, availability, and interoperability.

We are seeking the **artifact available badge** and **the artifact functional badge** for this artifact. To make the review process convenient, we provided ipynb notebooks to run the experiments.

A.2 Description & Requirements

A.2.1 Security, privacy, and ethical concerns

The artifact does not pose any security concern.

A.2.2 How to access

The code for the artifact can be accessed at: https://github.com/CellularLint/cellularlint-codes/releases/tag/v1.0.0
The pretrained models can be accessed at: https://zenodo.org/records/12199206

*Equal contribution. The student author's name is given first.

The processed SNLI train dataset can be accessed at: https://zenodo.org/records/12249320

A.2.3 Hardware dependencies

Our artifact was successfully run with the following hardware-

- CPU: AMD Ryzen Threadripper PRO 5965WX (24 core, 3.8 GHz)
- GPU: Nvidia RTX 3090 (24GB)
- Memory: 64GB

Additionally, we recommend 50 GB of available disk space. However, it may work with less.

A.2.4 Software dependencies

The artifact can be successfully run on Ubuntu 20.04 LTS and using python3. Please follow the README in our GitHub repository to setup the environment and solve software dependencies.

A.3 Set-up

To setup the environment, we provided a `requirements.txt` (with exact versions we used) in our repository. We recommend creating a virtual environment and installing the packages. All the required libraries can be installed with-

```
pip install -r requirements.txt
```

The requirements were generated using `pip freeze` from a multi-user server and modified manually to consider only the required packages. If a package is missing or runs into a problem, please feel free to contact us.

A.3.1 Installation

Once the GitHub repository (https://github.com/CellularLint/cellularlint-codes/) is cloned or the artifact is downloaded from the latest release (https://github.com/CellularLint/

cellularlint-codes/releases/tag/v1.0.0) please download the pretrained models and store them under "./Pretrained Models/" directory. Detailed instructions can be found in the GitHub README file.

A.4 Evaluation workflow

A.4.1 Major Claims

Following are the major claims we make for the artifact available and artifact functional badge-

(C1): CellularLint generates two similarity matrices from context-preserving segments (Paper Figure 3). This is proved by part of experiment 1.

(C2): CellularLint compares embedding techniques and finds TF-IDF to be most effective (Paper Figure 4). This is proved by part of experiment 1.

(C3): CellularLint can be successfully run to train models and generate evaluation metrics (to produce Table 1) in the paper. This is proved by experiment 2.

A.4.2 Experiments

Please run the following experiments to verify the claims:

(E1): Tokenizers & Embeddings. [30 human-minutes+1 compute-hour+1GB disk]: From the main directory, run-
`python3 tokenizer_and_sim_matrix.py 4G` and
`python3 tokenizer_and_sim_matrix.py 5G`
for 4G and 5G datasets, respectively. Each of these generates one PDF and one PNG formatted image file (Thus, in total 4 files are generated). The generated files are `4G_embedding_times.png`, `heatmap_4G.pdf`, `5G_embedding_times.png`, `heatmap_5G.pdf`. The PDF files can be compared to Figure 3 of the paper, and PNG files can be compared to Figure 4 of the paper.

(E2): Model Performance.
Small dataset: [1 human-hour+2 compute-hours+50GB disk]
Large dataset: [1 human-hour+24 computer-hours+50GB disk]

- Run the notebooks sequentially in the following order-
  `train_bert.ipynb`,
  `train_roberta.ipynb`,
  `train_xlnet.ipynb`,
  `phase_train_bert.ipynb`,
  `phase_train_roberta.ipynb`,
  `phase_train_xlnet.ipynb`
- From the `eval/` directory, run-
  `python3 -W ignore eval.py`
  It will generate the metrics like Table 1 (one phase) of the paper.

Note. Each notebook can be run without any modification to the cells. Since there are many cells, we recommend using **Kernel** → **Run all cells** (such as found in Jupyter interface). For faster and more convenient run, we provided small datasets and set them in the notebooks. This should take minimal time. However, the notebooks can also be configured to use the larger datasets. To do so, please comment out the 6th cell from each of the first 3 notebooks (`train_bert.ipynb`, `train_roberta.ipynb`, `train_xlnet.ipynb`) before running them sequentially.

A.5 Notes

Please note that while running notebooks for experiment 2, it might be necessary to stop other notebook kernels before proceeding with the next ipynb notebook, particularly for models like XLNet.

A.6 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/usenixsec2024/.