УДК 004.896:004.056.5

АВТОМАТИЗАЦИЯ ПРОЕКТИРОВАНИЯ СИСТЕМ ОБНАРУЖЕНИЯ ФИШИНГОВЫХ URL НА ОСНОВЕ МАШИННОГО ОБУЧЕНИЯ

Свиридчук И.Л., Малькевич К.Д.

Белорусский государственный университет информатики и радиоэлектроники, Минск, Республика Беларусь, isl06work@gmail.com

Аннотация: в работе рассматривается задача автоматизированного проектирования интеллектуальных систем обнаружения фишинговых веб-ресурсов. Предложен подход к созданию классификатора URL-адресов на основе методов машинного обучения, интегрируемый в архитектуру систем информационной безопасности. Проведён сравнительный математический анализ двух моделей: Random Forest и BERT. Исследованы их теоретические свойства, вычислительная сложность и качество классификации. Показано, что выбор архитектуры определяется требованиями к точности и производительности, что соответствует задачам автоматизированного проектирования адаптивных систем защиты.

Ключевые слова: автоматизация проектирования, машинное обучение, обнаружение фишинга, Random Forest, BERT, системы защиты информации.

I. ВВЕДЕНИЕ

Современные системы информационной безопасности требуют автоматизированных решений для оперативного выявления угроз. Проектирование таких систем включает выбор архитектуры, методов обработки данных и алгоритмов принятия решений. В данной работе рассматривается задача автоматизированного проектирования классификатора фишинговых URL как компонента системы защиты, способного принимать решения без участия человека. Обнаружение фишинговых ресурсов по строке URL без обращения к содержимому страницы является актуальной задачей, решение которой может быть интегрировано в браузерные расширения, почтовые фильтры и межсетевые экраны. С формальной точки зрения задача сводится к построению бинарного классификатора [1]:

$$f: \sum^{*} \to \{0,1\},$$

где Σ^{*} – множество всех возможных строк URL, а f – решающее правило, относящее URL к легитимным (0) или фишинговым (1).

II. МОДЕЛЬ RANDOM FOREST

а. Дерево решений

Random Forest – это ансамблевый метод, строящий множество решающих деревьев. Каждое дерево в отдельности является классификатором, а объединение их решений позволяет снизить ошибку за счёт уменьшения дисперсии. Базовым классификатором в Random Forest является бинарное дерево решений. В каждом узле v проверяется условие $x_j<\theta_v$, где j – номер признака, θ_v – пороговое значение, p_c – доля примеров класса c в узле D. Для оценки качества разбиения используется индекс Джини [2]:

$$G(D) = 1 - \sum_{c=0}^{1} p_c^2,$$

При разбиении узла на левую (D_L) и правую (D_R) части вычисляется прирост чистоты:

$$\Delta G = G(D) - \frac{|D_L|}{|D|} G(D_L) - \frac{|D_R|}{|D|} G(D_R).$$

Индекс Джини достигает минимума (G=0), когда все примеры в узле принадлежат одному классу, и максимума (G=0.5), когда классы распределены поровну. Таким образом, дерево стремится к разбиениям с минимальным G(D). Дерево выбирает такое разбиение, которое максимизирует ΔG. Этот процесс продолжается до тех пор, пока не будет достигнута максимальная глубина или пока узлы не станут чистыми.

б. Ансамблирование

Однако одно дерево склонно к переобучению – оно может идеально классифицировать обучающую выборку, но плохо работать на новых данных. Чтобы уменьшить ошибку обобщения, Random Forest строит T деревьев, каждое на своей бутстрап-выборке. Доля примеров, не попавших в выборку [3]:

$$lim_{N\to\infty}(1-\frac{1}{N})^N = e^{-1} \approx 0.368.$$

Дополнительная случайность вводится при выборе признаков в каждом узле ($m \approx \sqrt{d}$, где d — общее число признаков). Это снижает корреляцию между деревьями и улучшает обобщающую способность ансамбля. Снижение корреляции напрямую влияет на ошибку обобщения ансамбля. Ошибка обобщения E ограничена сверху, где ρ – средняя корреляция между деревьями, σ^2 – их средняя сила. Поскольку E пропорциональна ρ, уменьшение корреляции за счёт случайного выбора признаков приводит к снижению ошибки обобщения:

$$E \leq \rho \cdot \frac{1-\sigma^2}{\sigma^2}.$$

Сложность обучения: O(T·d·N log N). Сложность предсказания: O(T·d). Это делает Random Forest пригодным для высоконагруженных систем автоматизированной защиты.

III. МОДЕЛЬ BERT

а. Механизм внимания

BERT (Bidirectional Encoder Representations from Transformers) представляет собой модель глубокого обучения, которая решает ту же задачу минимизации, но принципиально иным способом. Метод основан на архитектуре Transformer [4]. Ключевой элемент – механизм масштабированного скалярного произведения-внимания. Для матриц запросов Q, ключей K и значений V. Механизм внимания позволяет модели учитывать контекст каждого символа URL относительно всех остальных:

$$Attention(Q,K,V) = softmax\left(\frac{QK^T}{\sqrt{d_k}}\right)V,$$

В отличие от рекуррентных сетей, BERT обрабатывает последовательность токенов $t_1,\ldots,t_n$ параллельно, учитывая контекст как слева, так и справа. Это достигается за счёт того, что в механизме внимания BERT отсутствует маскировка будущих позиций. Каждый токен может "общаться" с любым другим токеном в последовательности, независимо от его положения. Это позволяет модели улавливать контекстные зависимости, которые невозможно обнаружить при одностороннем анализе. Классификационное решение принимается на основе скрытого состояния токена [CLS]: $h_{[CLS]} \in R^{768}$.

б. Функция потерь и оптимизация

Обучение BERT – это настройка параметров θ для минимизации ошибки классификации. В качестве меры ошибки используется кросс-энтропия, где θ– параметры модели, а $y_i^{(c)}$ – индикатор того, что i-й пример принадлежит классу:

$$L(\theta) = -\frac{1}{N}\sum_{i=1}^{N}\sum_{c=0}^{1} y_i^{(c)}\, log\, P(y_i = c \mid s_i;\theta).$$

Кросс-энтропия штрафует модель, когда она уверенно ошибается: если модель предсказывает вероятность 99% для неверного класса, логарифм даёт большое отрицательное число, и после смены знака ошибка становится большой. Для минимизации этой функции используется оптимизатор AdamW [5]. Его ключевая идея – адаптивно регулировать скорость обучения для каждого параметра. Он использует две скользящие средние: инерция (среднее градиента), ускоряющая движение в постоянном направлении (m_t); адаптация (среднее квадрата градиента), уменьшающая шаг, если градиент сильно колеблется.

$$\hat{m}_t = \frac{\beta_1 m_{t-1} + (1-\beta_1)\nabla L(\theta_t)}{1-\beta_1^t}, \qquad \hat{\vartheta}_t = \frac{\beta_2 \vartheta_{t-1} + (1-\beta_2)(\nabla L(\theta_t))^2}{1-\beta_2^t}.$$

Если градиент для какого-то параметра сильно колеблется, $\sqrt{\hat{\vartheta}_t}$ становится большим, и шаг уменьшается. Если градиент стабильный, шаг увеличивается; $\epsilon=10^{-8}$ – малая константа, добавленная в знаменатель для предотвращения деления на ноль на ранних шагах обучения; $\eta\lambda\theta_t$ – L2-регуляризация (weight decay), штрафующая большие веса. Маленькая скорость обучения $\eta=2\times10^{-5}$ критически важна: BERT уже предобучен на огромном корпусе текстов, и слишком большие шаги могут разрушить его знания. Адаптивность AdamW позволяет безопасно настраивать модель под задачу классификации URL. Финальное обновление параметров, где $\eta\frac{\hat{m}_t}{\sqrt{\hat{\vartheta}_t}+\epsilon}$ – адаптивный градиентный шаг с учётом инерции и стабильности:

$$\theta_{t+1} = \theta_t - \eta \frac{\hat{m}_t}{\sqrt{\hat{\vartheta}_t} + \epsilon} - \eta\lambda\theta_t.$$

Сложность внимания: $O(n^2 \cdot d)$, где n – длина последовательности (до 128 токенов), d – скрытая размерность (768). Это значительно выше, чем у Random Forest, что отражается в значении.

IV. СРАВНИТЕЛЬНЫЙ АНАЛИЗ

Экспериментальное сравнение моделей проводилось на основе публичного датасета malicious_phish.csv с платформы Kaggle [6], содержащего 651191 URL. Из полученного набора сформирована случайная выборка объёмом N = 30 000 примеров с фиксированным random_state = 42 для воспроизводимости. Для каждой модели данные разделялись на обучающую и тестовую выборки в соотношении 80/20. Random Forest обучался на 11 лексических признаках с гиперпараметрами n_estimators = 100. BERT дообучался на выборке из 5 URL в течение 4 эпох с оптимизатором AdamW ($\eta = 2\times10^{-5}$), что обусловлено ограниченными вычислительными ресурсами и необходимостью быстрой проверки работоспособности модели.

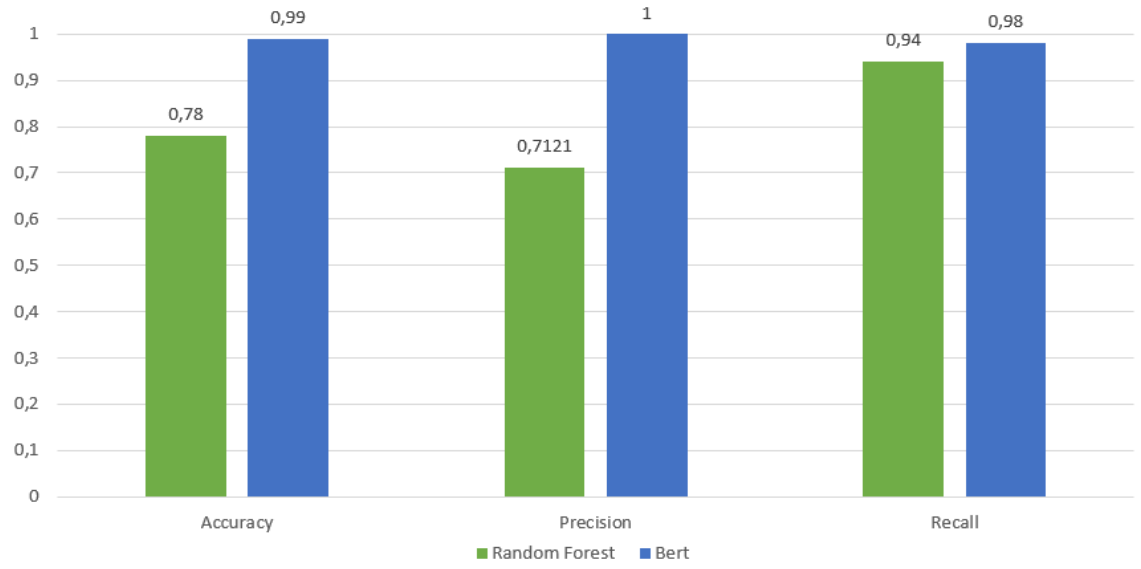


Рисунок 1. Сравнительные характеристики моделей

Для оценки качества классификации использовались следующие метрики. Accuracy – доля правильных ответов среди всех примеров: Accuracy=(TP+TN)/(TP+TN+FP+FN), где TP – верно определённые фишинговые URL, TN – верно определённые безопасные URL, FP – безопасные URL, ошибочно принятые за фишинговые, FN –фишинговые URL, не распознанные моделью. Precision – доля правильно определённых фишинговых URL среди всех URL, которые модель назвала фишинговыми: Precision=TP/(TP+FP). Этот показатель отвечает на вопрос: «Когда модель говорит, что URL фишинговый – можно ли ей верить?». Низкая Precision означает много ложных тревог. Recall – доля правильно обнаруженных фишинговых URL среди всех фишинговых URL в выборке: Recall=TP/(TP+FN). Этот показатель отвечает на вопрос: «Сколько фишинговых URL модель смогла найти?» Низкий Recall означает, что модель пропускает угрозы.

Random Forest достиг точности 78%. Высокий Recall (0.94) означает, что модель обнаруживает большинство фишинговых URL, но низкая Precision (0.71) указывает на значительное число ложных срабатываний – около 29% URL, определённых как фишинговые, на самом деле безопасны. BERT обеспечил точность 99%. Идеальная Precision (1.00) означает отсутствие ложных тревог – ни один безопасный URL не был ошибочно заблокирован. При этом BERT обнаруживает 98% фишинговых URL.

VI. ЗАКЛЮЧЕНИЕ

В работе рассмотрена задача автоматизированного проектирования классификатора фишинговых URL. Проанализированы две архитектуры: Random Forest и BERT. Показано, что Random Forest представляет собой эффективное решение для систем реального времени благодаря низкой вычислительной сложности и интерпретируемости. BERT обеспечивает максимальную точность за счёт глубокого контекстного анализа, но требует значительных ресурсов. Предложенный подход к проектированию позволяет автоматизировать выбор архитектуры на основе заданных критериев.

ЛИТЕРАТУРА

[1] Devlin, J. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding / J. Devlin, M.-W. Chang, K. Lee, K. Toutanova // arXiv preprint arXiv:1810.04805. – 2018.

[2] Hastie, T. The Elements of Statistical Learning / T. Hastie, R. Tibshirani, J. Friedman. – 2nd ed. – New York : Springer, 2009. – 745 p.

[3] Breiman, L. Random Forests / L. Breiman // Machine Learning. – 2001. – Vol. 45, No. 1. – P. 5–32.

[4] Vaswani, A. Attention Is All You Need / A. Vaswani [et al.] // Advances in Neural Information Processing Systems. – 2017. – P. 5998–6008.

[5] Kingma, D. P. Adam: A Method for Stochastic Optimization / D. P. Kingma, J. Ba // ICLR. – 2015.

[6] Malicious URLs dataset [Electronic resource] // Kaggle. – Mode of access: https://www.kaggle.com/datasets/sid321axn/malicious-URLs-dataset. – Date of access: 07.09.2026.

AUTOMATION OF DESIGNING PHISHING URL DETECTION SYSTEMS BASED ON MACHINE LEARNING

I.L. Svirydchuk, K.D. Malkevich

Belarusian State University of Informatics and Radioelectronics, Minsk, Republic of Belarus,
isl06work@gmail.com

Abstract: the paper addresses the problem of automated design of intelligent systems for detecting phishing web resources. An approach to creating a URL classifier based on machine learning methods, integrated into the architecture of information security systems, is proposed. A comparative mathematical analysis of two models – Random Forest and BERT – has been carried out. Their theoretical properties, computational complexity, and classification quality have been studied. It has been shown that the choice of architecture is determined by the requirements for accuracy and performance, which corresponds to the tasks of automated design of adaptive protection systems.

Keywords: design automation, machine learning, phishing detection, Random Forest, BERT, information protection systems.