УДК 004.021
DOI https://doi.org/10.32838/2663-5941/2020.2-1/16

Киричек Г.Г.
Національний університет «Запорізька політехніка»

Тимошенко В.С.
Національний університет «Запорізька політехніка»

СИСТЕМА АНАЛІЗУ ВИКОРИСТАННЯ ІНТЕРНЕТ-РЕСУРСІВ

У роботі проведено дослідження методів аналізу використання інтернет-ресурсів; обрано програмне забезпечення для реалізації системи; спроектовано систему аналізу використання інтернет-ресурсів на основі нейронних мереж за допомогою сервісу creately.com, який дозволяє створювати діаграми різних варіантів; реалізовано програмні компоненти системи: модуль первинної обробки даних, модуль прогнозування та модуль керування; проведено експериментальне дослідження працездатності модулів системи.

Метою роботи є дослідження інтенсивності надходження пакетів і залежності навантаження від часу та прогнозування подальшого навантаження в системі аналізу використання інтернет-ресурсів. Об'єкт дослідження – процес реалізації системи аналізу використання інтернет-ресурсів. Предмет – моделі, методи, програмні та інструментальні засоби аналізу мережевих даних. Методи, які використовувалися: NumPy (для роботи з нелінійною алгеброю); Pandas (для аналізу даних і маніпулювання ними); Mathplot (для відображення даних у графічному вигляді) та TensorFlow – фреймворк, написаний на Python для створення нейронних мереж.

У роботі виконано моделювання загальних модулів системи. Описано послідовність дій первинної обробки даних перед потраплянням їх до модуля прогнозування та виконано моделювання роботи штучної нейронної мережі при керуванні пропускною здатністю мережі. Для перевірки Dos атак автори статті використовували алгоритм з етапами: виявлення перевантаження каналу зв'язку; створення шаблонів трафіку залежно від часу отримання першого піку; виявлення послідовності отриманих піків трафіку; прийняття рішень за наявності або відсутності атаки.

Ключові слова: *балансувальник навантаження, обчислення, штучна нейронна мережа, атака, дані, фреймворк.*

Постановка проблеми. Постачальникам послуг відомо, що користувачів цікавить швидкість отримання даних. Між користувачем і провайдером відносини будуються на компромісній основі, виходячи з їх можливостей. При цьому хмарні технології і технології віртуалізації дозволяють додаткам і мережам абстрагуватися від фізичної інфраструктури і програмно надавати мережу як послугу. Натепер для аналізу інтернет-трафіку та навантаження мережі існує багато сервісів. Але всі вони мають низку недоліків: неможливість прогнозування навантаження системи; отримання даних із системи для подальшого аналізу сторонніми сервісами; виконання сплітестування і моделювання потенційної кількості користувачів; використання для конфігурування ресурсів залежно від навантаження.

Аналіз останніх досліджень і публікацій. У роботі розглянуті аналоги систем використання інтернет-ресурсів. Google Analytics – умовно безкоштовний сервіс для створення статистики відвідувань веб-сторінок; Яндекс метрика – сервіс, призначений для оцінки відвідувань вебсайтів і аналізу поведінки користувачів; Urchin on Demand – програма аналізу веб-статистики та глибока перевірка пакетів (Deep Packet Inspection, DPI). Існуючі аналоги мають низку недоліків: умовна безкоштовність, відсутність можливості експорту даних, зниження продуктивності мережі та вплив на статистику [1–3].

Оскільки обсяг використання інтернет-ресурсів є стохастичним процесом, для його аналізу доцільно використовувати нейронні мережі. Авторами розглянуті нейронні мережі прямого поширення (feed forward neural networks, FF або FFNN) і перцептрони, які передають інформацію від входу до виходу та використовуються в комбінації з іншими; нейронна мережа Хопфілда (Hopfield network, HN) – повнозв'язна нейронна мережа із симетричною матрицею зв'язків; багатошарові персептрони (MLP) – мережі прямого поширення та згорткові нейронні мережі (convolutional neural networks, CNN) і глибинні згорткові нейронні мережі (deep convolutional neural networks,

DCNN), які відрізняються від інших видів мереж [4–6]. Багатошарові персептрони вивчаються за допомогою алгоритму зворотного поширення помилки (back-propagation algorithm) і успішно застосовуються для вирішення багатьох складних завдань класифікації, розпізнавання [4].

Постановка завдання. Метою роботи є дослідження інтенсивності надходження пакетів і залежності навантаження від часу та прогнозування подальшого навантаження в системі аналізу використання інтернет-ресурсів. Об'єкт дослідження – процес реалізації системи аналізу використання інтернет-ресурсів. Предмет – моделі, методи, програмні та інструментальні засоби аналізу мережевих даних.

Для вирішення мети роботи необхідно виконати такі завдання: проаналізувати програмні методи реалізації подібних систем і методи аналізу трафіку; провести моделювання системи аналізу використання інтернет-ресурсів, реалізувати її програмні модулі; провести експериментальне дослідження функціонування модулів системи.

До системи висувається низка жорстких вимог, однією з яких є функціонування в режимі реального часу. В результаті необхідно мінімізувати тимчасові витрати, пов'язані з вивченням нейронної мережі, тому автори вибирають архітектуру, яка характеризується мінімальним часом навчання та розміром навчальної вибірки. Для коректного навчання нейронної мережі досить, щоб розмір навчальної вибірки L відповідав співвідношенню:

$$L = O\left(\frac{W}{\varepsilon}\right), \qquad (1)$$

де W – загальна кількість параметрів, які налаштовують (вагових коефіцієнтів і порогових значень); ε – точність помилки класифікації; O – порядок величини. Для помилки в 5% кількість прикладів навчання має в 5 разів перевершувати кількість вільних параметрів мережі W. Розглянувши архітектури та властивості мереж, для модуля первинної обробки даних автори використовують багатошаровий персептрон, а для модуля прогнозування – мережу Хопфілда.

Балансування завантаження сервера – технологія, яка розподіляє сайти з високим трафіком між декількома серверами. Сервери можуть знаходитись у приміщенні власних центрів обробки даних компанії або в хмарі. В роботі обрано метод балансування Sticky Sessions. Він зберігає список IP-адрес за серверами, що підвищує швидкодію балансувальника.

Потік як випадковий процес характеризується статистичними властивостями. Класичним для трафіку в мережах є пуассонівський потік:

$$P(k) = \frac{(\alpha t)^k}{k!} * e^{-\alpha t}, \qquad (2)$$

де k – кількість повідомлень; t – проміжок часу; α – інтенсивність потоку. Основною властивістю пуассонівского потоку є його адитивність [7]. Динамічні процеси, які відбуваються в сучасних мережах, належать до стохастичних. Тому потрібна модель, яка є випадковим процесом, керованим іншим випадковим процесом. Система є балансувальником навантаження та виконує функцію розподілу запитів між пулом серверів додатків. При створенні балансувальника навантаження піддається його публічна IP-адреса, яка і є адресою сервера додатків, а балансувальник навантаження розподіляє вхідні запити по пулу реальних серверів (Рис. 1).

Рис. 1. Балансувальник навантаження

Обробка даних в системі відбувається в кілька етапів: виявлення фактів – виділення з мережевого трафіку пакетів, які володіють значущими ознаками; підрахунок фактів – обчислення обсягу пакетів для кожної ознаки; агрегація фактів – визначення кількості пакетів для кожної ознаки за період часу; фільтрація даних – застосування статистичних методів для ослаблення впливу випадкових варіацій у тимчасових рядах; перевірка критеріїв – побудова списку критеріїв мережевих аномалій; отримання результату – визначення типів аномалій, виходячи з отриманої комбінації атак [8].

Виклад основного матеріалу дослідження. Для управління мережею автори застосовували

методи маршрутизації, управління трафіком і контролю завантаженості мережі, основані на даних, які надаються засобами прогнозування трафіку на основі попередніх значень. Для прогнозування було використано штучні нейронні мережі та паралельні динамічні системи з топологією спрямованого графа.

Автори знайшли найкоротший маршрут для групи вузлів мережі. Їх позначили як вузли A, B, C, а відстані між ними d_{AB}, d_{AC}, d_{BC}. Рішенням є впорядкована множина з n вузлів. Послідовність, в якій перебираються вузли, представлено матрицею розміру $n * n$, рядки якої відповідають вузлам, а стовпці – номерам вузлів у послідовності. Для вузлів A, B, C, D, E послідовність їх обходу задана матрицею, наведеною в Таблиці 1, вузол C підключається першим, вузол A – другим. Довжина маршруту дорівнює $d_{CA} + d_{AE} + ... + d_{DC}$. В кожному стовпці і в рядку цієї матриці є тільки одна одиниця, оскільки в кожен момент часу підключається тільки один вузол і тільки один раз.

Матрицю автори розглядають як стан нейронної мережі з $N = n^2$ нейронів. З $n!/2n$ маршрутів було обраємо один із найменшою довжиною. Стан кожного нейрона описується двома індексами, які відповідають вузлу і порядковому номеру підключення. $Y_{xj} = 1$ – вузол x є j-м по порядку вузлом маршруту.

Таблиця 1

Матриця для групи вузлів

Назва / №	1	2	3	4	5
A	0	1	0	0	0
B	0	0	0	1	0
C	1	0	0	0	0
D	0	0	0	0	1
E	0	0	1	0	0

Автори наводять функцію обчислення для мережі, призначеної для вирішення завдання маршрутизації, в якої стан із найменшою енергією відповідає найкоротшим маршрутам. Ця функція виглядає так [7]:

$$E = -\frac{1}{2}\sum_i\sum_j w_{ij}Y_iI_j - \sum_j I_iY_j + \sum_j T_jY_j, \qquad (3)$$

де E – обчислювальна енергія мережі; w_{ij} з – вага від виходу нейрона i до входу нейрона j; Y_j – вихід нейрона j; I_j – вхід нейрона j; T_j – поріг спрацювання нейрона j. Зміна енергії викликана зміною стану j-го нейрона:

$$\acute{}\mathrm{E} = \left(\sum_{i\neq j}\left(\mathrm{w_{ij}Y_i}\right) + \mathrm{I_j} - \mathrm{T_j}\right)\acute{}\mathrm{Y_j}, \qquad (4)$$

де $\acute{}\mathrm{Y_j}$ – зміна вхідного j-го нейрона; Y_j – вихід нейрона j; I_j – зовнішній вхід нейрона j; T_j – поріг спрацювання нейрона j.

Кожному стану системи відповідає конкретна величина обчислювальної енергії. При цьому функція енергії повинна підтримувати стійкість стану в формі матриці та ті рішення, які відповідають коротким маршрутам. Цим вимогам відповідає функція енергії виду [9]:

$$E = -\frac{A}{2}\sum_x\sum_i\sum_{j\neq i}Y_{xi}Y_{xj} + \frac{B}{2}\sum_i\sum_x\sum_{k\neq x}Y_{xi}Y_{kj} + \frac{C}{2}\left(\sum_x\sum_i Y_{xi} - n\right)^2 + \\ + \frac{D}{2}\sum_x\sum_{k\neq x}\sum_i d_{xk}Y_{xi}\left(Y_{k,j+1} + Y_{k,j-1}\right), \qquad (5)$$

При цьому $Y_{xi} = 0,1$. A, B, C, D – додатні множники. Перший член дорівнює 0, якщо кожен рядок x містить не більше одної одиниці. Другий член дорівнює нулю, якщо кожен стовпець містить не більше однієї одиниці. Третій член дорівнює нулю, якщо в матриці n одиниць. Без урахування четвертого члена функція енергії має мінімуми (E = 0) у станах, представлених матрицею з однією одиницею в кожному стовпці і кожному рядку. Всі інші мають більш високу енергію. Короткі маршрути підтримує четвертий член. В ньому індекси i беруться по mod n, щоб показати, що i-й вузол в маршруті з $(n-1)$ є першим, тобто $Y_{k,j+n} = Y_{k,j}$. Четвертий член дорівнює довжині маршруту. Вибір маршрутів максимізує ступінь вузла в мережі, дозволяє спланувати роботу так, щоб час її виконання був мінімальним.

Система розподіляє запити, які надходять не від всіх серверів, а серед необхідних за інформацією прогнозування. Для розподілу запитів автори використовували метод Sticky Sessions. Перед їх надходженням до модуля прогнозування виконується обробка даних і перевірка на підозрілість.

Сигнатурний аналіз як процес виявлення мережевих атак включає широкий клас правил, спрямованих на порівняння значень певних полів пакетів із рядом сигнатур, заданих користувачем або закладених статично в систему. Для пошуку шаблонних підстрок було використано алгоритм Бойера-Мура. Вхідні дані: підстроки keywords, вихідні дані: таблиці good_styffs і bad_char. Порівняння вхідного рядка string з keywords здійснюється справа наліво. Величина зсуву задається таблицями good_stuffs і bad_char. Для підвищення швидкості було використано багатопотокове

розпаралелювання операцій пошуку на рівні безлічі шаблонів підстрок.

Дані, які успішно пройшли перевірку на виявлення атак [10], зберігаються до бази даних системи, звідки надсилаються до модулю прогнозування. Було реалізовано і частину системи, відповідальної за прогнозування навантаження на сервіс. Набір даних автори розділили на дві частини: для тестування та для навчання.

```
start_training = 0
end_training = int(np.floor(0.5*n)
start_checking = end_training
end_checking = n
training_data = data[np. arange(start_trainingend_training,
check_data = data [np.arange(start_checking, end_checking)
```

Архітектури нейронних мереж використовують масштабування вхідних даних. Найчастіше використовуються активації випрямленою лінійною одиницею, тому було вирішено масштабувати вхідні дані і цілі, використовуючи для цієї мети метод MinMaxScaler в Python [11].

```
from sklearn.preprocessing import MinMaxScaler
sc = MinMaxScaler()
sc.fit(training_data)
training_data = sc.transform(training_data)
check_data = sc.transform(check_data)
training _X = training_data [1:]
training _Y = training_data [: 0]
check _X = check_data [:, 1:]
check _Y = check_data [:, 0]
```

Підрахунок статистики проводився на тренувальних даних, а потім отриманий результат застосовували до тестових даних.

Для реалізації нейронної мережі було використано TensorFlow. Після імпорту бібліотеки TensorFlow за допомогою tf.placeholder () визначалися плейсхолдери. Модель складається з чотирьох прихованих рівнів. Перший містить 1024 нейрони, що вдвічі перевищує обсяг вхідних даних. Наступні приховані рівні завжди вдвічі менші – вони об'єднують 512, 256 і 128 нейронів. Зниження кількості нейронів на кожному рівні стискає інформацію, яку мережа опрацювала на попередніх рівнях:

```
n_count = 500
neurons_count = [1024, 512, 256, 128]
target_count = 1
№ 1 hidden layer
layer_weight_1 = tensor_flow.Variable (weight_initializer ([n_stocks, n_neurons_1])
layer_shift_1 = tensor_flow.Variable (bias_initializer ([neurons_count [0]]).
№ 4 hidden layer
layer_weight_4 = tensor_flow.Variable (weight_initializer ([neurons_count [2], neurons_count[3])
layer_shift_4 = tensor_flow.Variable (bias_initializer ([neurons_count[3])
№ out layer
out_weight = tensor_flow.Variable (weight_initializer ([neurons_count [3], target_count])
out_shift = tf.Variable(bias_initializer([n_target])
```

Приховані рівні мережі трансформуються функціями активації. Ці функції – важливі елементи мережевої інфраструктури [12; 13].

Створено нейронну мережу 1024-512-256-128-1 за допомогою бібліотеки TensorFlow та перевірено її функціонування на дата-сеті, взятому у відкритому доступі при прогнозуванні кількості підключень (Рис. 2).

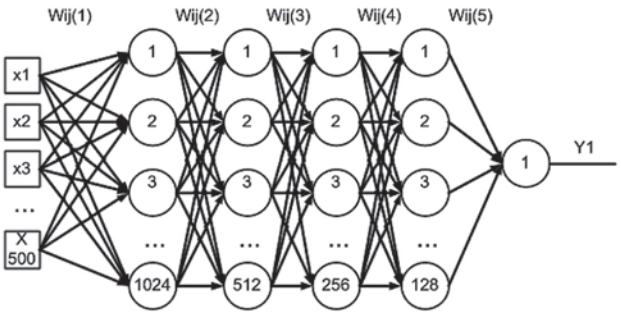

Рис. 2. Архітектура мережі

Інформація прогнозування надходить до керуючого модулю. Він поділяє отримані ряди на рівномірні проміжки часу та залежно від прогнозованої кількості запитів виконує форматування списку серверів. Автори наводять список при всіх активних серверах:

```
upstream somesite
{server general.someserv.com;
server reserve1.someserv.com;
server reserve2.someserv.com;
server reserve3.someserv.com;}
server {location / {proxy_pass http://somesite;}
```

Далі наведено конфігурацію при не активності деяких серверів:

```
upstream somesite
{server general.someserv.com;
server reserve1.someserv.com;
server reserve2.someserv.com fail_timeout = 1800s;
server reserve3.someserv.com backup;}
```

В разі не використання серверу більше ніж 12 годин, його помічали як бекап-сервер. Сервери, які скоро знадобляться, помічали як fail_timeout зі значенням у секундах (скільки часу він вважається вимкнутим) (Рис. 3).

Рис. 3. Загальний алгоритм роботи системи

Потім було виконано перевірку на наявність атаки, а із загальної кількості запитів вираховувалися підозрілі. Після обробки інформація надходила до керуючого модулю. Налаштування на сервері включає низку етапів. В роботі виконано конфігурування серверів на створення резервних записів та одержання доступу до головного сервера. Автори наводять процес генерації запитів:

import requests
def getSite (url):
response = requests.get(url)
for i in range(100):
getSite('http://ec2-18-217-201-16.us-east-2.
compute.amazonaws.com/')

При надходженні понад 10 запитів з однієї адреси за 1 секунду виконувалося блокування доступу з цієї адреси на 5 хвилин.

Висновки. В результаті роботи реалізовано систему аналізу використання інтернет-ресурсів на основі нейронних мереж. Система має преваги: статистка та дані належать власнику ресурсу, бо розташовані фізично на головному сервері додатку; дані можна отримати без зайвих зусиль для аналізу в будь-якому сторонньому аналізаторі; система за логікою відділена від основного додатку і зберігає статистику до файлу, тому відсутня ймовірність впливу на статистику під час її перегляду; є можливість проведення спліт-тестів; на відміну від звичайного балансиру навантаження розподіляє користувачів серед необхідної кількості серверів; система зменшує витрати та гарантує максимальну якість обслуговування.

Список літератури:

1. Можливості навчання й підтримки для служби Analytics. URL: https://support.google.com/analytics/answer/4553001?hl=uk (дата звернення: 10.03.20).

2. Эволюция структур данных в Яндекс.Метрике. URL: https://habr.com/ru/company/yandex/blog/273305/ (дата звернення: 10.03.20).

3. Киричек Г.Г., Киричек О.О. Модель оцінки плагіату програмного коду на основі системи контролю версій // Восточно-европейский журнал передовых технологий, 2012. № 2(2). С. 25–28.

4. Rosenblatt F. The perceptron: a probabilistic model forinformation storage and organization in the brain. URL: https://www.ling.upenn.edu/courses/ cogs501/Rosenblatt1958.pdf (дата звернення: 10.03.20).

5. Hopfield J.J. Neural networks and physical systems with emergent collective computational abilities. URL: https://bi.snu.ac.kr/Courses/g-ai09-2/hopfield82.pdf (дата звернення: 10.03.20).

6. Zeiler M., Krishnan D., Taylor G., Fergus R. Deconvolutional Networks. URL: https://www.matthewzeiler.com/mattzeiler/deconvolutionalnetworks.pdf (дата звернення: 10.03.20).

7. Barbu V. Semi-Markov Chains and Hidden Semi-Markov Models toward Applications. URL: Режим доступу: https://books.google.com.ua/books?hl =en&lr=&id=U9pQ_LyIeLcC&oi=fnd&pg=PR7&dq=Markov+hidden+chains&ots=LYVh_aJRNi&sig=_Rpw2tKTnSHML6iv9L7KDvukP_Y&redir_esc=y#v=onepage&q=Markov%20hidden%20chains&f=false (дата звернення: 10.03.20).

8. Kirichek G., Harkusha V., Timenko A., Kulykovska N. System for detecting network anomalies using a hybrid of an uncontrolled and controlled neural network. In: CEUR Workshop Proceedings 2546, 2019. P. 138–148.

9. LeCun Y. et al. Gradient-based learning applied to document recognition. Proceedings of the IEEE, 1998. № 86(11). P. 2278–2324.

10. Tao Y., Yu S. DDoS attack detection at local area networks using information theoretical metrics. 12th IEEE International Conference on Trust, Security and Privacy in Computing and Communications. IEEE, 2013. С. 233–240.

11. Kirichek G., Tymoshenko V., Rudkovskyi O., Hrushko S. Decentralized System for Run Services. In: CEUR Workshop Proceedings 2353, 2019. P. 860–872.

12. Vincent P. et al. Extracting and composing robust features with denoising autoencoders. In: 25th international conference on Machine learning, 2008. P. 1096–1103.
13. Kingma D.P., Welling M. Auto-encoding variational bayes. arXiv preprint arXiv:1312.6114, 2013.

Kirichek G.G., Timoshenko V.C. SYSTEM OF ANALYSIS OF INTERNET RESOURCES USE

In the work carried methods research of analysis of Internet resources use; selected software for system implementation; designed a system of analysis of Internet resources use based on neural networks using the service creately.com, which allows you to create diagrams of different options; the software components of the system are implemented: primary data processing module, forecasting module and control module; experimental study of system modules performance was performed.

In this work propose to research the intensity of packet arrivals and the dependence of load on time and predict further load in the system of analysis of Internet resources use. Object of research – the process of implementing a system of analysis of Internet resources use. Subject – models, methods, software and tools for analyzing network data. Methods used: NumPy (for working with nonlinear algebra); Pandas (for analyzing and manipulating data); Mathplot (for graphical display of data) and TensorFlow are Python-based frameworks for creating neural networks.

In work was perform the modeling of system common modules. The sequence of actions of the primary data processing before entering them in the prediction module is described and simulation of the work of artificial neural network is performed while managing the network bandwid. To test Dos attacks, we use an algorithm with the steps of: detecting the congestion of the communication channel; creating traffic patterns based on the time of the first peak; detecting the sequence of traffic peaks received; making decisions about the presence or absence of an attack.

Key words: *load balancer, computation, artificial neural network, attack, data, framework.*