ISSN (p) 0321-2211, ISSN (e) 2663-3450

УДК 004.021, 004.62, 004.855.5

РОЗРОБКА СИСТЕМИ ГЕНЕРАЦІЇ НА ОСНОВІ АЛГОРИТМУ ШТУЧНОГО ІНТЕЛЕКТУ. ЧАСТИНА 1

Вережинський В. А., Даниленко А. В., Рупіч С. С., Цибульник С. О.
Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського», Київ, Україна
E-mail: serhii.rupich@gmail.com

Розвиток сучасних інструментів і методів штучного інтелекту забезпечує широкий функціонал у різних наукових і прикладних сферах та надає можливість досягати нових результатів. Постає питання щодо застосування елементів штучного інтелекту для знаходження нових поєднань і сполучень шляхом генерації. В якості цільового результату розглядається застосування нейронної мережі для формування нових, унікальних рецептів харчової суміші, з метою визначення ефективності впровадження та використання методів і моделей штучного інтелекту. Для вирішення задачі генерації поєднань складових рецептів, у роботі розглянуто всі етапи технологічного процесу від збору, аналізу, препроцесингу даних до вибору математичного алгоритму та моделювання навченої системи. Кожен з етапів роботи містить у собі повний опис кроків, необхідних для вирішення поставленого завдання. Алгоритм машинного навчання має можливість обробляти тисячі прикладів для знаходження певної закономірності поєднань інгредієнтів. І значна увага приділяється формуванню тренувального датасету.

На етапі збору даних розглядаються особливості та основні проблеми, що виникають при роботі парсера. Важливою частиною є обхід захисних технологій при запитах до інтернет-ресурсів та зчитування DOM-дерева з коду HTML сторінки. Описано налаштування алгоритму автоматизованого збору інформації. Для розробки парсеру в роботі використовуються інструменти для швидкого розгортання проєкту та ефективного керування додатками Docker та Docker Compose. Окремим етапом є побудова датасету та аналіз даних для моделі нейронної мережі, який полягає у проведенні препроцесінгу та декодуванні і композиції у табличний вигляд. При автоматизованому процесі збору даних формується інформація зі значними шумами та зайвими елементами. Значна увага приділяється саме процесу очищення та підготовки корисної інформації, адже саме від чистоти даних та їх повноти в більшості випадків залежить якість математичної моделі та процесу моделювання для знаходження нових закономірностей та поєднань.

Ключові слова: *генерація; аналіз; препроцесинг; математичний алгоритм; нейронна мережа; машинне навчання.*

Вступ

Кількість різноманітних пропозицій на ринку продажу товарів і послуг зростає з експонентною швидкістю кожен день. Виробники намагаються пропонувати унікальну продукцію, яка могла б знайти місце серед споживачів. Помітна частина їх потужностей направлена на підвищення унікальності свого товару. Крім того, товар має відповідати якісним та кількісним потребам користувачів, а також бути конкурентоспроможним. Проте, на сьогодні, навіть великі команди професіоналів не здатні конкурувати з автоматичними алгоритмами у генерації таких унікальних товарів. Вже існують самостійні алгоритми, які можуть абстрагувати специфіку виробництва та створювати нові товари на основі існуючих. Наприклад, такі системи можуть генерувати оптимальні поєднання хімічних елементів у сплавах, перебирати необхідні компоненти для створення антидотів до різноманітних хвороб, знаходити найдоцільніші пропорції у поєднаннях різних інгредієнтів у харчовій промисловості [1-2]. Однією з істотних переваг таких систем є відсутність необхідності ручного перебору можливих удосконалень товару, тобто процес є повністю або майже повністю автоматизованим. Вони створюються з впровадженням сучасних комп'ютерно-інтегрованих технологій, без яких обробка, аналіз і отримання результату досліджень займали б неймовірно багато часу. Для вирішення поставлених задач часто використовуються елементи штучного інтелекту та різні алгоритми. Тому актуальною задачею використання алгоритму генерації продукту є не тільки створення великої кількості різноманітних варіантів, а й збереження уваги споживачів внаслідок знаходження системою нових, нестандартних і цікавих рішень.

Постановка задачі

На сьогодні вирішення завдання генерації привернуло увагу великої кількості науковців і розробників та отримало широку підтримку серед виробників. Зазвичай це не є тривіальним завданням. Складність процесу зберігається протягом всіх етапів реалізації, починаючи від початкових

даних, вибору алгоритму, підходу генерації і до інтерпретації вихідного результату. Для цього створюються так звані генеративні моделі [1], які повністю або частково реалізують поставлене завдання та побудовані на основі штучного інтелекту або з використанням його елементів. Проте, на практиці генеративні моделі здатні вирішувати вузьке коло завдань, оскільки кожне наступне у прогресії збільшує складність всієї системи в цілому.

Генеративні моделі наразі здатні вирішувати завдання генерації аудіо, зображень, створювати унікальні тексти та навіть підтримувати більш-менш конструктивну розмову з людиною [3]. Однак, ці всі системи неідеальні та постійно вдосконалюються. Як зазначалося, особливостями таких систем є впровадження комп'ютерно-інтегрованих технологій. Однією з широко вживаних та вдалою технологією є штучні нейронні мережі, які використовуються у багатьох галузях виробництва компаніями у різних сферах. Нейронні мережі – це дуже потужний інструмент, особливо у завданнях генерації. Їх особливість полягає у нелінійному перетворенні вхідних даних. Завдяки притаманній їм нелінійності можна отримати зовсім неочікувані результати. Багато з них дуже часто є нелогічними чи дивними з точки зору людини, проте частина результатів може мати цінність, особливо при генерації унікального продукту.

Однією з областей, де є можливість легко створювати та випробовувати автоматично згенеровані результати без завдання шкоди для навколишнього середовища та біологічним організмам, є харчова промисловість, коли навіть абсолютно незвичайні поєднання компонентів можливо зробити внаслідок доступності компонентів. Тому в цій роботі пропонується використати підхід автоматичної генерації для створення нових рецептів по заздалегідь підготовленим інгредієнтам.

Зазвичай, процес генерації не є тривіальною задачею. Щоб побудувати подібну модель з використанням інструментів штучного інтелекту, потрібно виконати значну підготовчу роботу. Класичний алгоритм побудови моделей машинного навчання складається з декількох етапів [4].

Перш за все, потрібно обрати ресурси з даними, що будуть в подальшому оброблятись, та розробити алгоритм їх автономного аналізу. Оскільки навчальної інформації потрібно багато, то в ручному режимі її збір займатиме дуже багато часу. Тому, як правило, в такому випадку застосовується парсинг, тобто автоматичний збір даних.

Після отримання необхідної кількості зібраної інформації виконується її підготовка для подальшого аналізу та використання. Цей процес називається препроцесінг даних, коли виконується чистка від «сміття» та пропусків, структуризація, сортування, класифікація тощо, для подальшої якісної та ефективної роботи з масивами даних у математичних моделях, в тому числі й генеративних.

Після препроцесінгу вся інформація представляється та зберігається у табличному вигляді. Однак, кількість записів в таблиці може досягати десятки, сотні тисяч або навіть мільйони. Для людини оперування таким великим об'ємом даних майже неможливо. Тому, для оцінки отриманих взаємозв'язків, закономірностей та аналізу вихідного результату часто використовуються візуальне представлення.

Перші етапи є надзвичайно важливими, адже в більшості випадків саме від якості та структурованості підготовки початкової інформації залежить кінцевий результат. На фінальному етапі підбирається або обирається та налаштовується найефективніша модель для генерації, після чого вже безпосередньо створюється бажаний продукт проєкту.

Нижче кожний з етапів розглядається детальніше.

Особливості отримання даних

Одна з проблем у навчанні алгоритмів – збір даних. Якість виконання будь-якого алгоритму напряму залежить від якості та кількості даних.

Існує багато підходів до отримання даних, використання яких залежить від області дослідження, необхідного типу та виду інформації, доступних часових та інших ресурсів [5]. Частіше за все, цей процес супроводжується значними ресурсними затратами, особливо потребує витрат часу та фінансів. Тому, в нашому випадку, щоб побудувати генеративну модель, як було зазначено вище, буде застосовуватися метод автоматичного збору інформації, або парсинг, що є одним з ефективних підходів до отримання масиву інформації у відносно стислий проміжок часу та не потребує багато додаткових джерел, на кшталт людських ресурсів, фокусної групи і тому подібних статистичних інструментів.

Для багатьох досліджень використовують дані (так звані датасети) із відкритих джерел, але для певних задач їх може просто не бути. Як наслідок, датасети потрібно добувати самостійно. Можна вважати успіхом, якщо є один або кілька інтернет-ресурсів, котрі можуть містити якісні дані для навчання алгоритму. Але навіть якщо такі джерела було знайдено, це ще не означає, що отримання потрібної інформації є тривіальною задачею. Тут постає проблема, що кожен більш-менш унікальний інтернет-ресурс запобігає передаванню усіх своїх накопичених даних третім особам, оскільки його дані – це актив. Запобігання парсингу є одним з пріоритетів інтернет-ресурсів з багатьох причин, і не тільки через унікальність. Власники подібних ресурсів, скоріш за все, хочуть не тільки забезпечити надійність зберігання даних, а й бути впевненим, що реальні користувачі під час спроб парсингу матимуть надійний та стабільний доступ до повного функціоналу сайту, оскільки більшість методів автоматичного збору інформації створю-

ють серйозне навантаження і заважають нормальній роботі ресурсу. Інтернет-ресурси цілеспрямовано намагаються завадити парсингу. Для цього існує багато різноманітних механізмів.

На початковому етапі формування датасету потрібно обрати інтернет-ресурси з даними, що будуть в подальшому оброблятись, та проаналізувати структуру сайтів за допомогою перегляду DOM-дерева. DOM-дерево – це об'єктна модель документа, котру створює браузер у пам'яті комп'ютера базуючись на HTML-коді, що отримав від сервера [6]. Тобто, коли браузер запитує у сервера певну сторінку та отримує від нього її початковий HTML-код, йому необхідно спочатку його розібрати по складовим вузлам. У процесі розгляду та аналізу цього коду на його основі будується DOM-дерево. І знаючи структуру інтернет-ресурсу, далі вже з відповідних місць у коді зчитується потрібна інформація.

Способи попередження автоматичного збору даних часто є трудомісткими (як і сам парсинг) і можуть негативно впливати на реальних користувачів. Тому необхідно спершу зрозуміти природу захисту конкретного ресурсу від збору даних, і які перешкоди виникають при спробі їх отримати. Очевидно, що чим більш вузько спрямована та складна інформація викладена на сайті, яку необхідно зібрати, тим складнішою буде і методика захисту від парсингу. Наприклад, TOR-проксифікація дає можливість змінювати IP-адресу кожні кілька хвилин, хоча навіть це повністю не захищає від блокування час від часу. Досить популярним методом запобігання парсингу є обмеження числа запитів за одиницю часу з однієї IP-адреси. Наприклад, сайт може дозволяти лише кілька пошукових запитів в секунду від одного користувача, що сповільнює роботу парсерів та суттєво знижує ефективність їх використання. Якщо дії на сайті виконуються занадто швидко або швидше, ніж може реальний користувач, то може з'явитися пропозиція застосувати капчу (невеликий комп'ютерний тест для того, щоб визначити, чи не є користувач системи комп'ютером) (рис. 1).

Рис. 1. Приклади капч

Також сайт може надавати платний або обмежений офіційний доступ по API. API – це опис можливостей (набір функцій, структур, класів процедур та/або констант), за допомогою яких різні програми можуть взаємодіяти одна з одною [7].

Якщо на інтернет-ресурсі буде помічена незвична активність, наприклад, багато схожих пошукових запитів з однієї IP-адреси, фіксується перегляд занадто багатьох сторінок одночасно або відправляється велика кількість запитів, то можливе блокування доступу або пропозиція вирішення капчі.

Веб-ресурси можуть не обмежуватись лише перевіркою по IP-адресі. Використовуються й інші методи виявлення підозрілої активності користувачів. Наприклад, швидкість заповнення форм, місце натиску на кнопки, збір даних про розмір, роздільну здатність екрану, використані шрифти, часовий пояс тощо за допомогою мови JavaScript.

Є сайти, де потрібно пройти реєстрацію, для перегляду інформації. Для парсерів це достатньо сильний стримуючий фактор. Таким чином ресурс має змогу відслідковувати дії з конкретного облікового запису і легко визначає той, що використовується для збору даних, і в подальшому блокує його.

Один із способів подолання захисного функціоналу полягає у запитах, що надсилаються з хмарного хостинку або IP-адрес парсингових вебсервісів, наприклад, Amazon Web Services, Google app Engine або VPS (віртуально виділених серверах). Інтернет-ресурси заздалегідь можуть блокувати такі адреси або спеціально для них запитувати додаткові підтвердження того, що користувач є реальною особою, але це також впливає на справжніх користувачів.

Ще один цікавий спосіб захисту даних полягає у повній конвертації текстового контенту на сайті у зображення. Це одразу відсікає можливість застосувати парсери, що працюють за принципом зчитування тексту. Проте, використання цієї технології має низку недоліків для власників подібних ресурсів. По-перше, ці сайти майже повністю унеможливлюють роботу програм для зчитування з екрану (наприклад, для людей з вадами зору). По-друге, пошукові системи пропускають такі вебресурси, навіть при повному збігу тексту, оскільки текст представлений у вигляді зображення. Крім того, знижується продуктивність та швидкість роботи такого ресурсу.

Проблематичний для парсерів підхід захисту контенту – фальшиві та невидимі дані. Цей метод ніяк не заважає звичайним користувачам, оскільки вони просто ніколи не зможуть знайти або натиснути на контент, який не відображається на сайті. Візуально сайт ніяк не видає наявність таких даних. Досить просто зробити багато фальшивих блоків інформації, які парсер шукатиме по спільним ідентифікаторам. Туди розробники можуть підставити посилання, яке проігнорувала би звичайна людина, проте парсер не зміг би відрізнити «підробку». Після автоматичного натискання на посилання може бути видано тимчасове або вічне блокування IP-адреси, що використовувалась для збору даних, або видача звичайної капчі для усіх подальших запитів з цієї адреси, що сильно ускладнить процес парсингу. Окрім того, використання несправжніх даних сильно псує пакет усіх отрима-

них результатів, бо при великих об'ємах автоматично відсортувати лише корисну інформацію стає практично неможливо.

Також парсер може буди заблокований, якщо не передбачити запит ресурсів сайту, наприклад, CSS (каскадні таблиці стилів, що є зовнішнім оформленням сайту) або зображення. Справжній браузер при використанні користувачем практично завжди буде давати запит і завантажувати зображення та CSS, проте для парсеру «цікаві» лише зміст сторінки та їх склад. Це може бути виявлено, і парсер буде заблокований.

Алгоритм збору даних. Препроцесінг даних

Знаючи основні методи захисту інформації, наступним кроком є написання алгоритму, що міг би збирати дані з ресурсів у автономному режимі та приводити їх до уніфікованого вигляду. Оскільки ресурси генерують дані по запиту (є статичними), можна використовувати синхронні запити, які повертають HTML-код сторінки.

Проект парсеру складається з декількох конфігураційних файлів. Для швидкої розгортки додатку та роботи на будь-якій платформі були використані Docker та Docker Compose – інструменти для вирішення задач, пов'язаних з розгортанням проектів. Docker – це програмне забезпечення для автоматизації розгортання та керування додатками в середовищах з підтримкою контейнеризації, тобто контейнерізатор додатків. Контейнеризація – технологія, яка дозволяє розробнику ізолювати певні процеси ядра і змусити їх думати, що тільки вони працюють на зовсім новому комп'ютері [5, 8].

Структура парсеру наведена на рис. 2. Програма містить конфігураційні файли – Docker Compose та Dockerfile, основний файл – app.py та файл requirements.txt, який описує залежності проєкту.

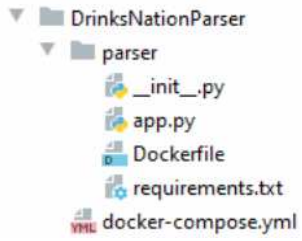

Рис. 2. Структура алгоритму для збору даних

Docker використовується для керування окремими сервісами (контейнерами), з котрих складається додаток. Docker Compose, у свою чергу, використовується для одночасного керування кількома контейнерами, що входять до складу додатку. Цей інструмент дозволяє робити те саме, що і Docker, але з більш складними додатками.

На рис. 3 приведена розроблена функція опиту веб-ресурсів.

```
def get_html(session, url: str,
             timeout: int = 8) → str:
    try:
        with session.get(url,
                         timeout=timeout,
                         proxies=proxies) as resp:
            assert resp.status_code == 200
            return resp.text
    except Exception as ex:
        print(ex)
```

Рис. 3. Функція, що опитує веб-ресурс

На рис. 4 та рис. 5 показано налаштування запуску алгоритму та середовища Docker.

Коли налаштування проєкту завершено, далі було написано необхідний алгоритм для автоматизованого збору інформації.

```
FROM python:3.7

COPY . /parser
WORKDIR /parser
RUN mkdir -p parser/data
ENV PYTHONUNBUFFERED=1

RUN pip install --no-cache-dir -r requirements.txt

CMD ["python", "./app.py"]
```

Рис. 4. Налаштування запуску алгоритму у Dockerfile

Функцію алгоритму наведено на рис. 6. Збір даних з кожного сайту – унікальна процедура, в процесі якої вручну відбувається ознайомлення зі структурою сайту, його архітектурою та особливостями використаних технологій, які формують DOM-дерево ресурсу. Після цього створюється програма, яка за допомогою веб-запитів буде отримувати код веб-сторінок, на яких містяться шукані дані. Наступним кроком потрібно скопіювати шляхи до шуканих елементів у DOM-деревах, де міститься шукана інформація та максимально оптимізувати їх, щоб вони працювали для кожної з груп веб-сторінок. Отриману інформацію потрібно перевірити на валідність за типом даних, регулярними виразами тощо, та повторити дану процедуру необхідну кількість разів. Для зберігання даних використовують бази даних, наприклад SQLite, PostgreSQL, або Redis (залежно від архітектури проєкту).

Після роботи парсеру отримані результати необхідно проаналізувати та виконати їх обробку, тому що в результаті парсингу, зазвичай, формуються засміченні дані, тобто присутня наявність значної кількості зайвої та непотрібної інформації.

Якість вирішення поставленого завдання для генеративної моделі залежить від попередньої підготовки вхідних даних, оскільки значна неінформативна множина прикладів призводить до

суттєвих відхилень та помилок кінцевої генерації. Зазвичай, препроцесінг зводиться до знаходження найбільш вагомих інформативно-ємнісних властивостей (факторів) у базі використовуваних даних, які визначають вагомий вплив на прецизійність закономірностей. В нашому випадку такими факторами та властивостями є закономірності у складі рецептів.

Коли алгоритм парсеру завершив свою роботу, на виході було отримано декілька тисяч json-файлів з рецептами, що представлені на рис. 7. Можна побачити, що файл складається зі значної кількості текстової інформації, що послідовно перераховується. Маркерами корисної інформації є складові інгредієнти, наприклад, «lime juice» або «apple cider». Проте, також присутні багато зайвих елементів, на кшталт роздільних символів, дужок, HTML-коду і т. п. Зробити очищення даних – одна з найбільш складних завдань препроцесінгу та машинного навчання в цілому [9]. Важливо не втратити цінну інформацію, при цьому видалити з масиву даних всю зайву. Варто зазначити, що при автоматизованому процесі неможливо як не втратити жодних корисних даних, так і повністю почистити від «сміттєвої» інформації.

```yaml
version: '3'

services:
  parser:
    build: parser/
    restart: 'no'
    depends_on:
      - tor-proxy
    expose:
      - '9150'
    volumes:
    - C:/Projects/DrinksNationParser:/parser/data
    environment:
      - HTTP_PROXY=socks5://0.0.0.0:9150/
    network_mode: 'host'

  tor-proxy:
    image: peterdavehello/tor-socks-proxy
    container_name: tor-proxy
    expose:
      - '9150'
    environment:
      - TORUSER=root
    network_mode: 'host'
```

Рис. 5. Налаштування середовища Docker Compose

```python
def parse_ingredients(html: str) -> dict:
    ingredients = dict()
    tree = etree.HTML(html)
    soup = BeautifulSoup(html, 'html.parser')
    ingred_table = soup.findAll(lambda tag: tag.name == 'ul' and tag.get('class') == ['small', 'r_ingredients'])
    all_ingr: List = [a1.text for a1 in ingred_table]
    ing_nmeas: List[str] = all_ingr[0].split('\n')[1:-1]
    meas_tree = etree.HTML(str(ingred_table))
    meas: List[str] = meas_tree.xpath('//li/text()')
    for im, m in zip(ing_nmeas, meas):
        ingredients[im.replace(m, '').strip().lower()] = m.strip().lower()
    name: str = tree.xpath('//*[@id="r_name"]/text()')[0].strip()
    try:
        glass = tree.xpath('//*[@id="r_extra"]/div/div/a/span/text()')[0].strip().lower()
    except Exception:
        glass = None
    try:
        instructions = tree.xpath('//*[@id="r_instructions"]/p/text()')
        instructions = [i.replace('\n', '').replace('\t', '').strip().lower() for i in instructions]
    except Exception:
        instructions = None
    return dict(name=name, ingredients=ingredients,
                glass=glass, instructions=instructions)
```

Рис. 6. Функція для збору рецептів з отриманих HTML-файлів

Застосувавши декодування інформації з json-файлів та її композиції у відповідні таблиці коктейлів та інгредієнтів за допомогою бібліотеки pandas, було отримане представлення даних, що наведено на рис. 8.

Для пошуку інгредієнтів для конкретної суміші необхідно звертатися до другої таблиці по графі **drink_id**. Поле **drink_id** було згенеровано, щоб при об'єднанні датасетів з різних веб-ресурсів можна було їх відрізняти, оскільки назви можуть повторюватися. Для об'єднання 4 датасетів, які було зібрано з інтернет-ресурсів, на даному етапі розробки немає можливості зробити це алгоритмічно. Тому було методично проаналізовано, класифіковано та уніфіковано близько 3,3 тисяч інгредієнтів. На виході отримано 411 спільних інгредієнтів. На рис. 9 наведено результат об'єднання датасетів.

Рис. 7. Отримані файли та їх вміст

	name	instructions	link	drink_id
0	Prawn Salad	Pour into an ice-filled highball glass. Garnis...	/drink6102.html	d7974286a00711eb9dda386b1c8624ab
1	Praying Mantis	Shake everything (except cola) with ice; strai...	/drinks1r4822.html	d7974287a00711eb98a7386b1c8624ab
2	Preacher's Hot Daughter	Pour mango rum over ice, fill with orange juic...	/drink1t14s823.html	d7974288a00711eb981b386b1c8624ab
3	Preakness Cocktail	Stir all ingredients (except lemon peel) with ...	/drink86.html	d79769a4a00711eb85ad386b1c8624ab
4	Presbyterian	Pour blended whiskey into a highball glass fil...	/drink5982.html	d79769a5a00711eb9ca2386b1c8624ab

Таблиця коктейлів

	ingredient	amount	drink_id
0	Pimm's® gin	1 1/2 oz	d7974286a00711eb9dda386b1c8624ab
1	Glayva® Scotch liqueur	1/3 oz	d7974286a00711eb9dda386b1c8624ab
2	Mandarine Napoleon® orange liqueur	1/3 oz	d7974286a00711eb9dda386b1c8624ab
3	Scotch whisky	1/3 oz	d7974286a00711eb9dda386b1c8624ab
4	lemonade	4 oz	d7974286a00711eb9dda386b1c8624ab
...	...	...	...

Таблиця інгредієнтів

Рис. 8. Датасети сумішей та інгредієнтів

	name	instructions	link	drink_id	glass
0	Prawn Salad	Pour into an ice-filled highball glass. Garnis...	/drink6102.html	d7974286a00711eb9dda386b1c8624ab	NaN
1	Praying Mantis	Shake everything (except cola) with ice; strai...	/drinks1r4822.html	d7974287a00711eb98a7386b1c8624ab	NaN
2	Preacher's Hot Daughter	Pour mango rum over ice, fill with orange juic...	/drink1t14s823.html	d7974288a00711eb981b386b1c8624ab	NaN
3	Preakness Cocktail	Stir all ingredients (except lemon peel) with ...	/drink86.html	d79769a4a00711eb85ad386b1c8624ab	NaN
4	Presbyterian	Pour blended whiskey into a highball glass fil...	/drink5982.html	d79769a5a00711eb9ca2386b1c8624ab	NaN
...	...	...	...	...	...
21020	Buzz Inducer	add yukon jack to a chilled beer mug and pour ...	NaN	679237fca05311eba0ff386b1c8624ab	beer mug
21021	Buzz Lightyear	build over cracked ice in a collins glass. se...	NaN	679237fda05311ebb99e386b1c8624ab	collins glass
21022	BW's Bong Water	place ice in hurricane glass. add ingredients...	NaN	679237fea05311ebae3e386b1c8624ab	hurricane glass
21023	Bye Bye Bahamas	fill a hurricane glass with crushed ice. add ...	NaN	679237ffa05311ebad46386b1c8624ab	hurricane glass
21024	By The Pool	shake ingredients in a cocktail shaker with ic...	NaN	67923800a05311eb98ea386b1c8624ab	highball glass

21025 rows × 5 columns

	ingredient	amount	drink_id
0	gin	1 1/2 oz	d7974286a00711eb9dda386b1c8624ab
1	liqueur	1/3 oz	d7974286a00711eb9dda386b1c8624ab
2	liqueur, orange	1/3 oz	d7974286a00711eb9dda386b1c8624ab
3	whiskey	1/3 oz	d7974286a00711eb9dda386b1c8624ab
4	lemonade	4 oz	d7974286a00711eb9dda386b1c8624ab
...	...	...	...
82690	oranges	1 slice	679237ffa05311ebad46386b1c8624ab
82691	liqueur, melon	1 oz.	67923800a05311eb98ea386b1c8624ab
82692	schnapps, peach	1 oz.	67923800a05311eb98ea386b1c8624ab
82693	soda, lemon	1 oz.	67923800a05311eb98ea386b1c8624ab
82694	juice, orange	1 oz.	67923800a05311eb98ea386b1c8624ab

82695 rows × 3 columns

Рис. 9. Результат об’єднання дата сетів

Висновки

В першій частині даної роботи розглянуто та наведено перші етапи процесу розробки системи генерації рецептів з використанням елементів штучного інтелекту, а саме налаштування автоматизованого збору даних (парсинг) та попередня обробка, очищення та компонування датасету (препроцесінг). Ці етапи є надзвичайно важливими для подальшої розробки, адже саме від якості навчальної множини значною мірою залежить ефективність всієї системи в цілому. Було розглянуто особливості захисту інтернет-ресурсів, які впливають на роботу парсеру. За допомогою Docker та Docker Compose побудовано алгоритм автоматизованого збору даних та оброблено декілька груп інтернет-ресурсів. За результатами декодованої інформації з отриманих json-файлів, було проаналізовано, класифіковано та уніфіковано більше 3

тисяч інгредієнтів. Дані було очищено від неінформативної, «сміттєвої» інформації та компановано в навчальні датасети, які в подальшому будуть використовуватися для налаштування математичної моделі генеративної системи.

В другій частині цієї роботи буде наведено візуалізацію даних, вибір та налаштування нейронної мережі та моделювання (генерація) результатів розробленим алгоритмом.

Література

[1] D. Foster, A. O. M. C. Safari, *Generative deep learning*. O'Reilly Media, Inc. 2022.
[2] T. Sousa, J. Correia, V. Pereira, M. Rocha, "Generative deep learning for targeted compound design", *J. Chem. Inf. Model*, vol. 61, no. 11, pp. 5343–5361, 2021. DOI: 10.1021/acs.jcim.0c01496
[3] A. Creswell, T. White [et. al.], "Generative Adversarial Networks: An Overview", *IEEE Signal Processing Magazine*, vol. 35, no. 1, pp. 53-65, 2018. [Online]. Available: https://arxiv.org/pdf/1710.07035.pdf
[4] S. Raschka, *Python machine learning*. Packt Publishing ltd. 2015
[5] A. Bhat, Data Collection Methods: Sources & Examples. [Online]. Available: https://www.questionpro.com/blog/data-collection-methods/
[6] S. Gupta, G. Kaiser, D. Neistadt, P. Grimm, "DOM-based Content Extraction of HTML Documents", in *Proceedings of the 12th international conference on World Wide Web*, pp. 207-217, 2003. DOI: 10.1145/775152.775182
[7] What Is An API (Application Programming Interface)? [Online]. Available: https://aws.amazon.com/what-is/api/?nc1=h_ls
[8] Docker overview. [Online]. Available: https://docs.docker.com/get-started/overview/
[9] M. Walker, Python Data Cleaning Cookbook: Modern techniques and Python tools to detect and remove dirty data and extract key insights, 1st Edition. Packt Publishing, 2020.

UDC 004.021, 004.62, 004.855.5

V. Verezhinskyi, A. Danylenko, S. Rupich, S. Tsybulnyk
National Technical University of Ukraine "Igor Sikorsky Kyiv Polytechnic Institute", Kyiv, Ukraine

DESIGN OF GENERATION SYSTEM BASED ON ARTIFICIAL INTELLIGENCE ALGORITHM. PART 1

The development of modern tools and methods of artificial intelligent provides a wide range of functionality for various scientific and applied tasks, and also allows obtaining new results. A question arises about how to apply elements of artificial intelligence to obtain new combinations and connections obtained by generation. As a purpose of the article is considered applying a neural network technology for the formation of new, unique recipes for cocktails with the aim of demonstrating the effectiveness of implementation and verification methods and models of artificial intelligence. To solve generation problems according to ingredient recipes, in the article all stages of the technological process are analyzed from data collection, analysis, preprocessing to the selection of a mathematical algorithm and modeling of the trained system. Each stage of the work contains a separate description of the steps necessary to solve the task. The machine learning algorithm has the ability to process thousands of examples to find a certain pattern of combinations of ingredients. And important attention is paid to the formation of a training date.

At the stage of data collection, individuals and main problems are considered, which are checked during the work of the parser. The important part is the partial use of encrypted technologies when using Internet resources and reading the DOM tree with the code of the HTML page. The paper shows settings for the automated algorithm of information collection. For the parser development, a tool for rapid project development and effective management as Docker and Docker Compose add-ons were used. Completed the stage with the acquisition of the data set and the data for neural network models has been analyzed. The stage consists in preprocessing, decoding and compositions collected data in relevant tables. During the automated data collection process, information with significant noise and redundant elements is processed. Considerable attention is paid to the process of cleaning and preparation of useful information, because the purity of the data and their completeness in the most cases determine the quality of the mathematical model and the modeling process for finding new laws and combinations in generation.

Ключові слова: generation; analysis; preprocessing; mathematical algorithm; neural network, machine learning.

Надійшла до редакції
17 квітня 2023 року

Рецензовано
18 травня 2023 року