DOI: 10.59647/978-617-520-936-3/1

Автоматизація перекладу коду за допомогою штучного інтелекту

Олексій Мацієвський[1] асистент (ORCID: 0009-0008-2341-8166)**, Ігор Ачкасов**[1] д.т.н проф. (ORCID: 0000-0002-7049-0530)

[1] *Київський національний університет будівництва і архітектури, 03037, м. Київ, проспект Повітряних Сил, 31, Україна*

АНОТАЦІЯ

У даній роботі представлено вирішення проблеми автоматизації перекладу програмного коду між різними мовами програмування за допомогою штучного інтелекту (ШІ). Така технологія дозволяє ефективно адаптувати код з однієї мови програмування на іншу, зберігаючи семантику та логіку. Методологія базується на використанні нейронних мереж і абстрактного синтаксичного дерева (AST). У статті описані основні етапи конвертації, проблеми та перспективи використання ШІ у програмному забезпеченні.

Ключові слова: штучний інтелект, машинне навчання, програмний код, конвертація коду, нейронні мережі..

1. ВСТУП

У сучасному світі багато програмних рішень розробляються на різних мовах програмування, що створює складнощі під час інтеграції та підтримки багатомовних проектів. Переклад коду між мовами програмування часто виконується вручну, що є трудомістким процесом і може призводити до помилок. Штучний інтелект (ШІ) та машинне навчання відкривають нові можливості для автоматизації цього процесу [1].

Застосування методів ШІ, зокрема нейронних мереж, дозволяє автоматично перекладати код з однієї мови на іншу, зберігаючи функціональність та семантику оригінального коду. Такий підхід може використовуватися для рефакторингу, оновлення програмного забезпечення або навчання [2].

2. МЕТА ДОСЛІДЖЕННЯ

Метою даного дослідження є розробка та впровадження нейронної мережі, здатної автоматизувати процес перекладу коду між мовами програмування [3]. У роботі зосереджено увагу на перекладі коду з мови Python на JavaScript з використанням трансформаторної архітектури для обробки текстових даних.

МЕТОДИ ДОСЛІДЖЕННЯ

Для вирішення поставленого завдання було використано декілька етапів:

- **Аналіз коду вихідної мови**. Штучний інтелект аналізує синтаксис та семантику вихідного коду за допомогою інструментів, які генерують абстрактне синтаксичне дерево (AST) [4].
- **Проміжне представлення коду**. Код на Python перетворюється у внутрішнє представлення, яке є агностичним до конкретної мови [5].
- **Генерація коду цільовою мовою**. Система генерує код на JavaScript, використовуючи правила трансформації, основані на нейронній мережі [6].

Основою для навчання моделі є дані, що містять приклади пар коду на Python та JavaScript. Для обробки використовувалася трансформаторна архітектура, яка добре зарекомендувала себе у вирішенні задач природної мовної обробки.

3. АРХІТЕКТУРА ТРАНСФОРМАТОРА ДЛЯ КОНВЕРТАЦІЇ КОДУ

Нейронна мережа, побудована на архітектурі трансформаторів, включає в себе два основні компоненти: енкодер та декодер. Енкодер аналізує вихідний код на Python та перетворює його у проміжне представлення. Декодер, у свою чергу, генерує код на JavaScript на основі проміжного представлення.

На рисунку 1, зображено функцію на Python, що додає два числа:

```python
def add_numbers(a, b):
    return a + b
```

Рисунок 1. Функція на python яка додає два числа

Ця функція отримує два аргументи a і b, додає їх і повертає результат.

Перед перетворенням потрібно зрозуміти логіку роботи функції:

- Параметри функції: a, b - числа, що додаються.
- Тіло функції: Виконується операція додавання a + b.
- Повернення результату: Функція повертає суму чисел a і b.

Наступним кроком є створення проміжного представлення для моделювання процесу. На рис. 2 показано просте проміжне представлення цієї функції у форматі JSON.

```json
{
  "function_name": "add",
  "parameters": ["a", "b"],
  "operation": "addition",
  "return_type": "number"
}
```

Рисунок 2. Функція проміжного представлення

Процес перекладу цієї функції передбачає створення проміжного представлення у форматі JSON, після чого генерується еквівалентний код на JavaScript, який зображено на рисунку 3.

```
function addNumbers(a, b) {
    return a + b;
}
```

Рисунок 3. Результат перетворення в JavaScript

JavaScript використовує інший синтаксис для оголошення функцій, але логіка операцій залишається тією ж самою.

Пояснення перетворення:

- При оголошенні функції в Python використовується ключове слово def, а в JavaScript - ключове слово function.
- Ім'я функції залишається тим самим, але в JavaScript прийнято використовувати camelCase для імен змінних і функцій.
- Параметри передаються у функцію без змін.
- Тіло функції та значення, що повертається: Логіка додавання не змінюється. JavaScript використовує крапку з комою для завершення виразів.

Цей код демонструє, як можна змоделювати процес конвертації коду з Python на JavaScript з використанням проміжного представлення та генерації шаблонного коду, а також метод ілюструє основний принцип конвертації між мовами програмування, хоча в реальних додатках процес може бути набагато складнішим і вимагати більш досконалих алгоритмів синтаксичного аналізу та генерації коду.

Такий підхід забезпечує високу точність та збереження логіки коду, що було підтверджено на експериментальних даних.

4. РЕЗУЛЬТАТИ

Розроблена модель показала високу ефективність у перекладі простих функцій і класів між мовами програмування. Процес конвертації є швидким, а отриманий код зберігає початкову функціональність. Проміжне представлення забезпечує гнучкість при роботі з різними мовами програмування.

Експериментальні результати демонструють здатність системи правильно обробляти більшість стандартних конструкцій мови Python та генерувати коректний код на JavaScript, як показано на прикладі класів та функцій.

5. ВИСНОВКИ

Використання штучного інтелекту для перекладу коду відкриває нові перспективи для розробників програмного забезпечення, скорочуючи час і витрати на ручний переклад коду між мовами. Однак цей процес все ще стикається з такими проблемами, як забезпечення точності перекладу та підтримання оптимальної продуктивності згенерованого коду. Подальші дослідження та розробка технологій у цій галузі можуть значно покращити якість та ефективність процесу конвертації коду, зробивши його більш доступним та зручним для широкого кола розробників.

Список джерел

[1] Use of Artificial Intelligence Systems for Determining the Career Guidance of Future University Student / S. Dolhopolov et al. *2022 International Conference on Smart Information Systems and Technologies (SIST)*, Nur-Sultan, Kazakhstan, 28–30 April 2022. 2022. URL: https://doi.org/10.1109/sist54437.2022.9945752 .

[2] Zhu Z., Sato Y. Deep Investigation of Intermediate Representations in Self-Supervised Learning Models for Speech Emotion Recognition. 2023 IEEE International Conference on Acoustics, Speech, and Signal Processing Workshops (ICASSPW), Rhodes Island, Greece, 4–10 June 2023. 2023. URL: https://doi.org/10.1109/icasspw59220.2023.10193018 .

[3] Wu X., Zheng Z., Weng J. Developmental Network-2: The Autonomous Generation of Optimal Internal-Representation Hierarchy. *IEEE Transactions on Neural Networks and Learning Systems*. 2021. P. 1–14. URL: https://doi.org/10.1109/tnnls.2021.3083759 .

[4] Visual Construction and Source Code Transformation Model Technology Based on Interface Dynamic Arrangement Technology / L. Yongqing et al. 2023 IEEE 6th International Conference on Automation, Electronics and Electrical Engineering (AUTEEE), Shenyang, China, 15–17 December 2023. 2023. URL: https://doi.org/10.1109/auteee60196.2023.10407672 .

[5] Maturana F., Rashmi K. V. Locally Repairable Convertible Codes: Erasure Codes for Efficient Repair and Conversion. 2023 IEEE International Symposium on Information Theory (ISIT), Taipei, Taiwan, 25–30 June 2023. 2023. URL: https://doi.org/10.1109/isit54713.2023.10206604 .

[6] Learning From Architectural Redundancy: Enhanced Deep Supervision in Deep Multipath Encoder-Decoder Networks / Y. Luo et al. IEEE Transactions on Neural Networks and Learning Systems. 2021. P. 1–14. URL: https://doi.org/10.1109/tnnls.2021.3056384 .