DOI 10.31891/2307-5732-2023-321-3-314-319
УДК 004.78:81`32

ФАНТ МИКОЛА
Державний університет «Житомирська політехніка»
ORCID ID: 0000-0002-4994-8009
e-mail: fantkolja@gmail.com

АРХІТЕКТУРА СИСТЕМИ МАШИННОГО НАВЧАННЯ ДЛЯ СТВОРЕННЯ ПАРАЛЕЛЬНИХ ДВОМОВНИХ КОРПУСІВ ТЕКСТІВ

Паралельні двомовні корпуси текстів – одна з основних частин будь-якого інструменту автоматизованого перекладу (CAT), а також важливі для інших завдань, пов'язаних із будь-яким типом перетворення тексту з однієї мови на іншу. У цій статті пропонується унікальна архітектура сервісу вирівнювання тексту, який базується на технологіях машинного навчання. Запропонована архітектура враховує новітні підходи до побудови систем мікросервісів, беручи до уваги легке розгортання і обслуговування таких систем. У статті детально розглядаються вимоги до системи створення паралельних корпусів текстів як вирішальної передумови розробки архітектури. Встановлені вимоги враховують обидві сторони системи: систему як застосунок машинного навчання та систему як CAT-сервіс. Запропонована архітектура дає можливість побудувати універсальну систему з декількома точками входу для кінцевих споживачів, системних адміністраторів і дата-інженерів. Вона також дозволяє різні варіанти використання системи: із власних користувацьких інтерфейсів або за допомогою викликів REST API зі стороннього сервера. Система містить три різні користувацькі інтерфейси, призначені для звичайних користувачів, системних адміністраторів, а також дата-інженерів. Такий гетерогенний підхід UX має вирішальне значення для безпечного, але гнучкого обслуговування системи. Система, побудована на запропонованій архітектурі, може охоплювати різні користувацькі сценарії: використовувати загальну модель для прогнозування власних двомовних текстових корпусів клієнтів, навчати власну модель або просто використовувати сервіс як сховище вирівняних двомовних текстів. Щоб досягти такої універсальності використання, велика увага приділяється підтримці керування версіями моделі, оскільки система повинна керувати різними паралельними версіями моделей прогнозування. Сервіс планується як система мікросервісної архітектури з оркестратором як центральним компонентом. Важливою частиною системи є служба моніторингу, яка буде оцінювати ефективність моделей, а також отримувати відгуки користувачів на основі дій користувачів після прогнозування моделі. У статті пропонується стек технологій, необхідний для легкої та безпечної розробки, розгортання та доставки продукту з нульовим часом простою за допомогою синьо-зеленої моделі розгортання.

Ключові слова: машинне навчання, модель, архітектура, двомовний корпус, інструмент CAT.

FANT MYKOLA
State University «Zhytomyrska Politekhnika»

ARCHITECTURE OF A MACHINE LEARNING SYSTEM FOR TEXT ALIGNMENT

The text alignment service is one of the essential parts of any Computer Aided Translation (CAT) tools and also important for other tasks, related to any kind of text transformation from one language to another. This article proposes a unique architecture of a text alignment service, which is based on machine learning technologies. The suggested architecture considers the newest approaches to constructing micro-services systems considering both easy deployment and maintenance of such systems. The article elaborates on requirements for the text alignment system as a crucial precondition of developing the architecture. The established requirements take into account both sides of the system: the system as a machine learning application and the system as a CAT service. The suggested architecture gives the possibility to build a universal system with several entry points for end customers, system administrators, and data scientists. It also preserves different options of the system usage: e.g. from the own user interfaces or with REST API calls from a third-party server. The system contains three different user interfaces designed for ordinary users, system administrators as well as data-scientists. That heterogenous UX approach is crucial for secure yet flexible system maintenance. The service built on the proposed architecture will be able to cover different user scenarios: using a general model for predicting customers' own bilingual text corpora, training their own model, or just using the service as a storage of aligned bilingual texts. To achieve such usage universality a great emphasis is given to model versioning support since the system should manage different parallel versions of the predicting models. The system is planned as a microservice architecture system with an orchestrator as its central component. An important part of the system is the monitoring service which will estimate the efficiency of trained models as well as get user feedback based on user actions after model predictions. The article suggests the technology stack needed for easy and secure development, deployment, and delivery of the product with zero downtime using the blue-green model of the deployment.

Keywords: machine learning, model, architecture, text alignment, CAT-tool.

Постановка проблеми

Паралельні корпуси текстів і сервіси для вирівнювання текстів (text aligners) є одним з центральних компонентів будь-якої CAT-системи. Водночас такі сервіси, та їх продукти мають велике значення як самостійна сутність для лінгвістики, особливо в перекладознавстві та корпусній лінгвістиці [1, 2]. Студії, присвячені вивченню застосування технологій машинного навчання для розробки інструментів автоматизації перекладу, тобто інструментів CAT, здебільшого орієнтовані на проблему побудови правильної моделі, підбору оптимального алгоритму або аналіз вхідного матеріалу [3–5]. Питання створення цілісної системи машинного навчання, готової до споживання кінцевим користувачем, особливостей її життєвого циклу, підтримки такої системи з подальшим натреновуванням та удосконаленням використаної моделі, залишається часто поза увагою науковців [6].

Аналіз останніх джерел

При аналізі останніх набутків у цій сфері важливо охопити загальні підходи до вибудовування архітектури систем машинного навчання, а вже на їх тлі розглянути, що було запропоновано саме в сфері CAT-інструментів.

В статтях *Луї Дорара* [7, 8] детально зображено узагальнену систему машинного навчання, яку було апробовано в декількох платформах для генерування моделей з необроблених текстових (Lateral – https://www.lateral.io, Google AutoML Natural Language – https://cloud.google.com/natural-language) або візуальних (Clarifai – https://docs.clarifai.com, Google AutoML Vision – https://cloud.google.com/vision) даних. В книзі *Джефа Сміта* подано та обґрунтовано цілісну загальну архітектуру системи машинного навчання. Варто зазначити, що автор вибудовує свою архітектуру базуючись на постулатах маніфесту реактивних систем (https://www.reactivemanifesto.org) і надає системі машинного навчання всіх ознак і переваг *реактивної системи*. Особливе місце в поданій архітектурі займають пайплайни (напр., для навчання моделі, для оцінки моделі тощо), що пояснюється бажанням автора створити справді безперебійну систему з системою "відкатів" у випадку виникнення помилки. *Джеоф Гультен* в своїй книзі розглядає системи машинного навчання як підвид "розумних систем" (intelligent system). Автор підходить до опису системи здебільшого з концептуальної точки зору, аналізує ситуації де такі системи взагалі матимуть значення, загострює свою увагу на проблемах збору даних, оцінки моделі. Також важливим є аналіз взаємодії між користувачем та "розумною системою", зокрема автор класифікує типи такої взаємодії як автоматизуюча, спонукальна, організуюча, анотаційна і гібридна. В нашому випадку система машинного навчання для створення паралельних двомовних корпусів відноситься до *автоматизуючої*, тобто така система матиме на меті замістити рутинні дії користувача.

Іоаніс Тріантафілу, Іасон Демірос, Крістос Малавазос та Стеліос Піперідіс [4] одними з перших спробували запропонувати архітектуру для вирівнювання текстів як елементу CAT-інструментів, проте на той час ще не було інтеграції з технологіями машинного навчання, крім того запропонована архітектура є дуже схематичною і складається всього з чотирьох здебільшого концептуальних елементів: обробки тексту, побудови двомовного лексикону, усунення неоднозначності опорної точки та фреймворку для динамічного програмування. Основна частина статті базується на визначенні найкращого алгоритму для визначення ступеня подібності між двома паралельними послідовностями слів. *Тетяна Вакалюк, Черниш Оксана* та *Левківський Віталій* провели широкий аналіз архітектур сучасних електронних словників для різних мов, з використанням різних технологій [9]. Авторам вдалося здійснити досить детальний розгляд, який охоплював як конкретні технології і сервіси, які входили до складу систем, так і загальний опис архітектури з зображенням відношень та ролі кожної складової, а також можливості користувацьких інтерфейсів зображених електронних словників. Проведений аналіз став важливим підґрунтям для розробки власного алгоритму проектування електронного тлумачного словника. *Едуардо Сендехас, Гретель Барсело, Александер Гельбух* та *Грігорій Сідоров* [10] дослідили можливості включення лінгвістичної інформації до системи вирівнювання текстів на статистичному лексичному рівні. У статті було запропоновано загальну архітектуру вирівнювача текстів, яка експліцитно не включає технології машинного навчання, проте залишає простір для їх застосування. Головну увагу статті все ж приділено спробі поєднання статистичного та лінгвістичного підходів під час побудови двомовного корпусу текстів. Вважаємо можливим використання такого підходу для розбудови системи властивостей (features) для навчання моделі.

Метою статті є розробка власної архітектури системи машинного навчання для створення паралельних двомовних корпусів текстів для її подальшої реалізації на основі аналізу існуючих підходів та зразків архітектур систем машинного навчання для розв'язання задач обробки природної мови в цілому та моделювання CAT-інструментів зокрема, а також на основі аналізу вимог до запланованого застосунку.

Виклад основного матеріалу

Запланований застосунок переслідуватиме дві головні цілі – автоматичне створення та використання двомовного корпусу текстів, а також можливість створення нового (в тому числі на основі кастомізації загального) користувацького корпусу, доступного лише для користувача, чи групи користувачів-авторів цього корпусу. Крім цього застосунок можна буде використовувати просто як сховище готових двомовних корпусів. У всіх випадках створення корпусу буде відбуватися за допомогою технологій машинного навчання, базуючись на загальній або похідній користувацькій моделі. Така система відношень між загальною і користувацькою моделями має бути зручною і продуктивною, з огляду на спосіб генерування та використання пам'яті перекладів у опен-сорс CAT-системи Matecat (https://www.matecat.com). Адже важко не провести паралелі між моделлю, як ядром системи машинного навчання, та пам'яттю перекладів, як ядром будь-якої CAT-системи: в обох випадках ядро акумулює досвід (машинний у машинному навчанні, та перекладацький у CAT-системі) і допомагає генерувати рішення для розв'язання подальших нових задач (обраних типових задач у машинному навчанні або нових сегментів для перекладу тексту-оригіналу в CAT-системі) на основі цього досвіду.

Таким чином, важливими *факторами*, які будуть впливати на систему, є:

- здатність підтримувати та варіювати версійність моделі;
- удосконалення існуючої моделі шляхом отримання даних користувача і шляхом додавання нових даних від одного з внутрішніх сервісів;

- можливість використання окремо за допомогою власного користувацького інтерфейсу, поряд з можливістю інтегрування в інші CAT-інструменти, сервіси і системи, наприклад за допомогою публічного REST або GraphQL API.

Наведені фактори впливають на загальні *вимоги* до застосунку як до однієї з систем машинного навчання.

Вимоги до системи: Загалом системи машинного навчання вимагають спеціальної архітектури, розробленої для підтримки ефективного навчання та високої точності передбачень. Важливо розрізняти поняття *застосунку машинного навчання* (machine learning application) та *системи машинного навчання* (machine learning system). *Застосунок машинного навчання* – це програма, яка базується на алгоритмах, які уможливлюють навчання та вдосконалення машини на базі аналізу даних без експліцитного програмування [11]. *Система машинного навчання* – це набір програмних компонентів, які вирішують певну прикладну проблему, організовані певним чином для їх використання кінцевим споживачем, включають в себе застосунок машинного навчання і передбачають можливості для безперебійної роботи, розгортання, удосконалення, постачання нових версій та покращення моделі застосунку машинного навчання [12].

З огляду на своє призначення і будову архітектура системи машинного навчання диктується з одного боку механізмом функціонування застосунку машинного навчання, а з іншого боку прикладним аспектом створеної системи та способу її постачання кінцевому споживачу.

Процес машинного навчання зазвичай складається з наступних фаз [13]:

1. *Визначення бізнес-задачі*. На цьому етапі вирішується питання доцільності використання застосунку машинного навчання, наявні альтернативи, ресурсо- і трудозатрати.
2. *Отримання даних*. Цей етап фокусується на пошуку і отриманні даних, потрібних для тренування моделі. В залежності від специфіки даних визначається вид машинного навчання (контрольоване, неконтрольоване, підкріплене). Дані є основою будь-якої системи машинного навчання. Якість і кількість даних, які використовуються для навчання системи, мають значний вплив на точність її прогнозів. Тому збір даних і попередня обробка є вирішальними кроками в створенні системи машинного навчання.
3. *Розробка властивостей даних (feature engineering)*. Цей етап передбачає обробку сирих даних з другого етапу у систему даних, представлених спеціальним чином (напр., як стовпці в електронній таблиці) для можливості їх споживання алгоритмами машинного навчання.
4. *Тренування моделі*. На цьому етапі відбувається вибір алгоритму та налаштування його параметрів на базі підготовлених на третьому етапі даних. У контрольованому навчанні модель навчається на позначених даних, тоді як неконтрольоване навчання покладається на непозначених дані. Навчання з підкріпленням передбачає метод проб і помилок із системою винагород. Тренування моделі відбувається в ході почергового споживання даних та оцінки отриманої моделі. Важливим при цьому є також розподілення даних на тренувальну, оціночну та тестувальну частини.
5. *Розгортання моделі*. На цьому етапі відбувається перевірка, наскільки якісно модель здатна справлятися з новими даними, тобто наскільки оптимально вирішується бізнес-задача, встановлення на першому етапі.

Крім вище згаданих ключових фаз процесу машинного навчання, існують також інші фактори, які необхідно враховувати при розробці архітектури системи машинного навчання. До них належать:

- апаратне забезпечення;
- програмне забезпечення;
- постачання.

Вибір *апаратного забезпечення* може значно вплинути на продуктивність системи машинного навчання. Спеціалізоване обладнання, таке як GPU і TPU, зазвичай використовується для прискорення процесів навчання та використання моделі машинного навчання, зокрема TPU було розроблено компанією Google виключно для виконання задач машинного навчання [14].

Стек *програмного забезпечення***,** що використовується для створення системи машинного навчання, має бути оптимізовано для забезпечення продуктивності, масштабованості та простоти використання. Популярні програмні інфраструктури, такі як *TensorFlow, PyTorch і Keras*, забезпечують високорівневий інтерфейс для створення та навчання моделей машинного навчання.

Постачання системи машинного навчання включає її інтеграцію у виробниче середовище. Архітектура розгортання повинна бути розроблена для забезпечення масштабованості, надійності та безпеки.

Пропозиція архітектури системи машинного навчання для вирівнювання текстів. Для легкості функціонування системи бажано організувати її таким чином, щоб була єдина точка входу для взаємодії з цією системою. Проте у випадку з системою машинного навчання потрібно на архітектурному рівні забезпечити можливість не лише кінцевих користувачів системи і адміністраторів, а і дата-інженерів, які повинні мати змогу без додаткової публікації нової версії сервісу, в режимі реального часу вносити зміни у характеристики (features) для підготовки даних і навчання моделі.

З огляду на вищевказане, слід розрізняти *користувацький, адміністраторський* та *дата-інженерний* виміри роботи цієї системи. Під *користувацьким* виміром ми розуміємо можливості та способи взаємодії з системою сервісів або клієнтів кінцевого користувача. Центральною користувацькою точкою входу є сервіс REST API, з яким може комунікувати сервіс користувацького інтерфейсу або зовнішній сервіс, який входить до CAT-системи з інтеграцією в нашу систему. Адміністраторський вимір володітиме власним сервісом з

окремим користувацьким інтерфейсом. Користувацький та адміністраторський користувацькі інтерфейси розділені в окремі docker-контейнери для того щоб мінімізувати ймовірні небезпеки та вразливості системи, внаслідок маніпуляції з фронтендом. В тому числі для цього доступ до адміністраторський фронтенду можливий лише через VPN-під'єднання. Для дата-інженерів виділено окремий контейнер з фронтендом, який через сервіс оркестратора має доступ до сервісу генерування характеристик, сервісу оцінки і побудови моделі. Таким чином дата-інженери матимуть змогу продовжувати розробку додатку машинного навчання в системі безпосередньо під час її функціонування.

Зображена архітектура є мікросервісною. Кожен із сервісів, в тому числі фронтенд-сервіси запущено в окремому докер-контейнері (https://www.docker.com). Запуск цих контейнерів та комунікація між ними здійснюється за допомогою автоматизованої системи постачання, масштабування і менеджменту контейнеризованих систем – Kubernetes (https://kubernetes.io). Версійність кожного сервісу відповідає версійності образів (image) контейнерів. Образи розміщено в приватному сховищі докер-образів Quay (https://quay.io). Завдяки цьому досягається безпека та приватність програмного забезпечення.

Розгортання сервісів та підняття версії відбувається автоматично через вбудовану у git процедуру (pipeline) безперервної інтеграції і безперервного постачання (CI/CD). CI/CD реалізовано за допомогою GitHub Actions та ArgoCD (https://argo-cd.readthedocs.io). Сервіс ArgoCD відповідає за безперебійність роботи контейнерів, а також за їх версійність і параметризацію. Параметризація відбувається шляхом задавання змінних середовища, з якими запускається відповідний контейнер.

Що стосується баз даних для зберігання готових корпусів і для зберігання готових моделей, то до них не висувається особливих вимог. Оптимальною буде будь-яка нереляційна база даних, на кшталт MongoDB. Як альтернативу можна розглянути базу даних в парі з хмарним файловим сховищем, для зберігання корпусів у вигляді XML файлів, а моделей у серіалізованих JSON файлах.

Центральне місце в системі займає сервіс оркестратора. Він поєднує всі інші сервіси, зокрема головну точку входу, REST API, з сервісами для побудови і оцінки моделі, а також сервісом для генерування характеристик (feature store). Процедура створення нової версії моделі починається з обробки вхідних даних через сервіс генерування характеристик. Після цього натреновується нова модель в сервісі побудови моделі, яка потім проходить через етап оцінки в сервісі оцінки моделі. Якщо модель задовольняє вимоги системи, вона стає відкритою для використання. Частиною роботи оркестратора є також процес планувальника (scheduler), який з певною періодичністю запускає процес оновлення версії моделі з використанням нових даних. Моделі зберігаються в базах даних версійно, а також відповідно до користувача, якщо користувач вирішив створити свою власну модель, на базу загальної.

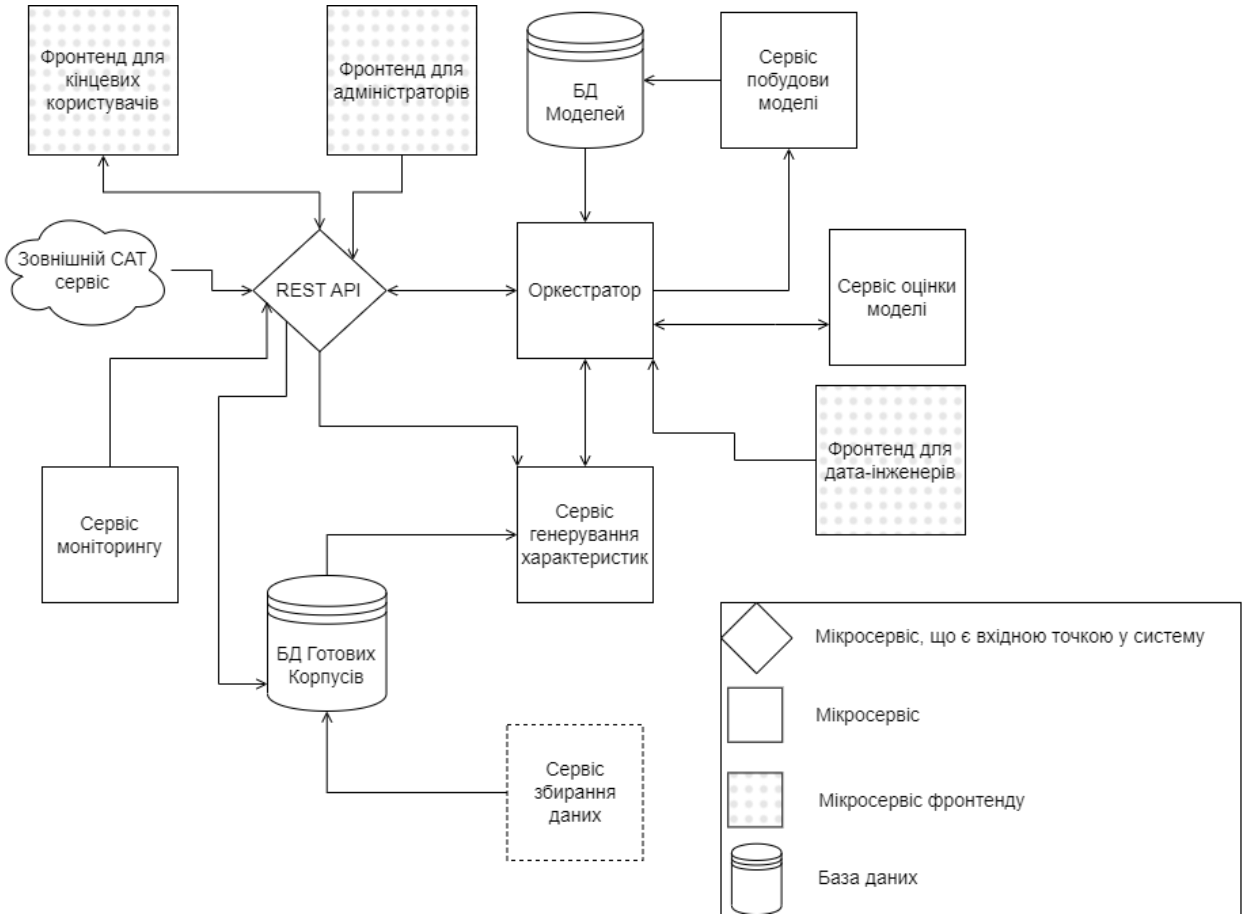

Рис. 1. Архітектура СМШ для створення двомовних корпусів

Важливим елементом системи є можливість інтеграції з будь-якими CAT-сервісами через REST API з двома цілями: використання готової моделі для виконання припущень, а також імпортування готових корпусів на базі подальшого редагування користувачем. Таким чином відбувається постійний приріст готових корпусів, а також приховано ведеться моніторинг за ефективністю роботи тієї чи іншої моделі, порівнюючи первинні результати припущення моделі та фінальну версію від користувача. Крім такого прихованого моніторингу слід також додати можливість безпосереднього зворотного зв'язку від користувача для отримання його суб'єктивної оцінки продуктивності моделі. Крім того, для підбору оптимальної моделі сервіс моніторингу може застосовувати так зване "тестування канарок" (canary testing), тобто випадковим чином обирати групу користувачів, яким надавати доступ до нової версії моделі і порівнювати їх результати з результатами користувачів з поточною версією, таким чином робити висновки про продуктивність нової версії.

Висновки

Розроблена архітектура уможливлює ефективний збір даних і попередню обробку, навчання моделі та прогнозування, а також розгортання у виробничому середовищі. При проектуванні було враховано останні напрацювання та проекти у відповідній сфері. Для забезпечення масштабованості, надійності та безпеки в тому числі враховано питання апаратного забезпечення, програмного забезпечення та постачання сервісу до кінцевого споживача. Архітектура надає можливість отримати доступ до сервісу з різних точок входу: через користувацький інтерфейс для кінцевих користувачів, за допомого REST API для сторонніх CAT-сервісів, а також через окремі фронтенд-сервіси для адміністраторів і дата-інженерів. Таким чином, система може обслуговувати різні потреби користувачів: полегшене створення нових двомовних корпусів текстів на базі загальної моделі або власної створеної в межах цієї ж системи моделі або просто використання наявної бази готових двомовних корпусів.

Література

1. Онищук М. Застосування паралельних корпусів текстів в перекладознавстві. Переяславська мовознавча толока. Переяслав-Хмельницький, 2019. С. 198–199.

2. Рубана Є. С. Конструювання лінгвістичного корпусу: від теорії до практики (на матеріалі німецької фахової мови архітектури та будівництва). 2022. URL: https://archer.chnu.edu.ua/handle/123456789/6005.

3. Марченко О. О., Никоненко А. О, Россада Т. В., Мельников Є.А. Метод машинного навчання для ідентифікації парафрази. Штучний інтелект, 2016, № 3. С. 128-136.

4. Triantafyllou I., Demiros I., Malavazos C., Piperidis S. An alignment architecture for Translation Memory bootstrapping. BCS International Academic Conference. URL: https://www.aclanthology.org/2000.bcs-1.3.pdf.

5. Giguet E., Luquet P. Multilingual Lexical Database Generation from Parallel Texts in 20 European Languages with Endogenous Resources. Proceedings of the COLING/ACL. Main Conference Poster Sessions. 2006. P. 271-278.

6. Лісік Ю. А., Грайворонський М. В. Архітектура аналітичної системи для виявлення шахрайських транзакцій. URL: https://ela.kpi.ua/bitstream/123456789/20805/1/9.Лісік.с.151-154.pdf.

7. Dorard L. Architecture of a real-world Machine Learning system. 2020. URL: https://medium.com/louis-dorard/architecture-of-a-real-world-machine-learning-system-795254bec646.

8. Dorard L. An overview of ML development platforms. 2020. URL: https://medium.com/louis-dorard/an-overview-of-ml-development-platforms-df953060b9a9.

9. Вакалюк Т., Черниш О., Левківський В. Зарубіжний досвід проектування електронних словників. Збірник наукових праць Уманського державного педагогічного університету. Вип. 2. 2021. С. 74-83.

10. Cendejas E., Barcelo G., Gelbukh A., Sidorov G. Incorporating Linguistic Information to Statistical Word-Level Alignment. Progress in Pattern Recognition, Image Analysis, Computer Vision, and Applications. 2009. P. 387-394.

11. Kanade V. What Is Machine Learning? Definition, Types, Applications, and Trends for 2022. URL: https://www.spiceworks.com/tech/artificial-intelligence/articles/what-is-ml.

12. Smith J. Machine Learning Systems: Designs that scale. Manning. 2018. 200 p.

13. Hulten G. Building Intelligent Systems – A Guide to Machine Learning Engineering. Apress Berkeley, CA. 2018. 339 p.

14. Fahim F. CPU vs GPU vs TPU: Understanding the Difference Between Them. 2022. URL: https://serverguy.com/comparison/cpu-vs-gpu-vs-tpu.

References

1. Onyshchuk M. Zastosuvannia paralelnykh korpusiv tekstiv v perekladoznavstvi. Pereiaslavska movoznavcha toloka. Pereiaslav-Khmelnytskyi, 2019. S. 198–199.

2. Rubana Ye. S. Konstruiuvannia linhvistychnoho korpusu: vid teorii do praktyky (na materiali nimetskoi fakhovoi movy arkhitektury ta budivnytstva). 2022. URL: https://archer.chnu.edu.ua/handle/123456789/6005.

3. Marchenko O. O., Nykonenko A. O, Rossada T. V., Melnykov Ye.A. Metod mashynnoho navchannia dlia identyfikatsii parafrazy. Shtuchnyi intelekt, 2016, № 3. S. 128-136.

4. Triantafyllou I., Demiros I., Malavazos C., Piperidis S. An alignment architecture for Translation Memory bootstrapping. BCS International Academic Conference. URL: https://www.aclanthology.org/2000.bcs-1.3.pdf.

5. Giguet E., Luquet P. Multilingual Lexical Database Generation from Parallel Texts in 20 European Languages with Endogenous Resources. Proceedings of the COLING/ACL. Main Conference Poster Sessions. 2006. P. 271-278.

6. Lisik Yu. A., Hraivoronskyi M. V. Arkhitektura analitychnoi systemy dlia vyiavlennia shakhraiskykh tranzaktsii. URL: https://ela.kpi.ua/bitstream/123456789/20805/1/9.Lisik.s.151-154.pdf.

7. Dorard L. Architecture of a real-world Machine Learning system. 2020. URL: https://medium.com/louis-dorard/architecture-of-a-real-world-machine-learning-system-795254bec646.

8. Dorard L. An overview of ML development platforms. 2020. URL: https://medium.com/louis-dorard/an-overview-of-ml-development-platforms-df953060b9a9.

9. Vakaliuk T., Chernysh O., Levkivskyi V. Zarubizhnyi dosvid proektuvannia elektronnykh slovnykiv. Zbirnyk naukovykh prats Umanskoho derzhavnoho pedahohichnoho universytetu. Vyp. 2. 2021. S. 74-83.

10. Cendejas E., Barcelo G., Gelbukh A., Sidorov G. Incorporating Linguistic Information to Statistical Word-Level Alignment. Progress in Pattern Recognition, Image Analysis, Computer Vision, and Applications. 2009. P. 387-394.

11. Kanade V. What Is Machine Learning? Definition, Types, Applications, and Trends for 2022. URL: https://www.spiceworks.com/tech/artificial-intelligence/articles/what-is-ml.

12. Smith J. Machine Learning Systems: Designs that scale. Manning. 2018. 200 p.

13. Hulten G. Building Intelligent Systems – A Guide to Machine Learning Engineering. Apress Berkeley, CA. 2018. 339 p.

14. Fahim F. CPU vs GPU vs TPU: Understanding the Difference Between Them. 2022. URL: https://serverguy.com/comparison/cpu-vs-gpu-vs-tpu.