УДК 004.054

О. А. Довгань
Є. А. Паламарчук

ПОШУКОВА СИСТЕМА ДЛЯ ЛОКАЛЬНОЇ БАЗИ ДАНИХ НА ОСНОВІ МЕТОДІВ ГЛИБИННОГО НАВЧАННЯ

Вінницький національний технічний університет

Анотація
В роботі досліджено шляхи реалізації пошукової системи для локальної бази даних на основі методів глибинного навчання з використання алгоритмів, які враховують специфіку даних у ній. Основна мета цієї системи, у порівнянні з іншими методами і системами пошуку, полягає у поліпшенні ефективності та точності пошуку внутрішньої інформації, що міститься в локальній базі даних.
Ключові слова: інтелектуальна пошукова система, локальна база даних, методи пошуку даних.

Abstract
The work explores ways of implementing a search system for a local database based on deep learning methods using algorithms that take into account the specifics of the data in it. The main goal of this system, in comparison with other search methods and systems, is to improve the efficiency and accuracy of the search for internal information contained in the local database.
Key words: intellectual search engine, local data base, methods of data searching.

Вступ

Попит на ефективні пошукові системи зріс експоненціально в епоху цифрових технологій, а локальні пошукові системи стали незамінними інструментами для підприємств і організацій, які керують великими обсягами даних. Серед них найбільш відомі системи пошуку такі як OpenAI Embeddings with plugins [1], Milvus [2], FAISS [3]. Вони використовують векторну репрезентацію слів для ефективного пошуку інформації, класифікації даних і семантичного пошуку. Ці системи пропонують гнучке налаштування алгоритму пошуку, новітні методи безпеки та підтримку кількох мов. Вони також інтегруються з іншими програмами, що спрощує керування даними та процес автоматизації процесів.

Створення пошукової системи для локальних баз даних, за рахунок технологій глибинного навчання – це проблема, яка вирішується в цій роботі. Дана проблема є досить актуально, адже все більше підприємств віддає перевагу локальним базам даних для зберігання власного продукту. Для швидкої взаємодії між розробниками необхідна пошукова система, що буде ефективно та швидко знаходити необхідні матеріали в базі даних.

Постановка задачі

Існує ряд програмних рішень, які можуть розв'язувати дану задачу: OpenAI Embeddings with plugins, Milvus, FAISS.

OpenAI Embeddings with Plugins

Поєднання моделей обробки великих текстових даних OpenAI із плагінами дозволяє розробникам використовувати потужності OpenAI для розширеного розуміння людської мови та семантичного пошуку в локальних базах даних. Завдяки інтеграції продуктів OpenAI з різними векторними базами даних ці плагіни забезпечують гнучкі та точні можливості пошуку, адаптовані до конкретних випадків використання та програм [4]. До їх основних технічних характеристик відносяться:

- вони створені для інтеграції векторних репрезентацій від OpenAI з різними векторними базами даних, що надає можливості семантичного пошуку;
- використовують можливості моделей GPT для розуміння людської мови, що дозволяє розробникам виконувати семантичні пошукові запити в локальних документах;
- векторна репрезентація дозволяє поєднувати декілька векторних баз даних, що забезпечує полегшення семантичого пошуку і надає гнучкість для розробників;

- дозволяють адаптувати до конкретних випадків використання попередньо навчених векторних репрезентації або спеціальні векторні репрезентаціїї на основі даних для вузьконаправлених задач;
- мають гнучкий механізм налаштування й оновлення параметрів, які визначають точність і релевантність пошуку.

Серед недоліків OpenAI Embeddings with Plugins слід відмітити таке:

- Обмежений обсяг функціоналу плагінів: плагіни часто створюються для конкретних випадків використання, що може обмежити їх застосування для різних сценаріїв.
- Залежність від OpenAI: можливості цих плагінів залежать від мовних моделей OpenAI, що робить їх залежними від оновлень і вдосконалень OpenAI.
- Затримка пошуку в базі даних: інтеграція векторних репрезентацій від OpenAI із зовнішніми векторними базами даних може призвести до затримки в процесі пошуку, що вплине на продуктивність.
- Інтелектуальна власність і ліцензування: моделі OpenAI підлягають умовам ліцензування, які можуть обмежувати їх використання в деяких комерційних програмах або вимагати додаткової плати.

Milvus

Пошукова система Milvus відмінно справляється з великомасштабними векторними даними, що дозволяє здійснювати ефективний пошук подібності та класифікувати дані алгоритмом найближчих сусідів. Пропонує кілька методів індексування та сумісність із різними фреймворками машинного навчання, Milvus дозволяє розробникам створювати потужні додатки на основі штучного інтелекту (ШІ) з високопродуктивними можливостями пошуку [5]. До основних технічних характеристик відносять:

- Milvus — це векторна база даних із відкритим вихідним кодом, спеціально розроблена для додатків ШІ;
- він підтримує різні типи векторних репрезентацій та може працювати з кількома фреймворками машинного навчання;
- пропонує розподілену архітектуру, що робить її масштабованою та придатною для обробки великих обсягів даних;
- підтримує кілька методів індексування;
- надає SDK для різних мов програмування, у тому числя для Python, Java і Go, що полегшує його інтеграцію в різні програми та системи;
- підтримує розділення даних і керування версіями даних, що спрощує керування та підтримку великих наборів даних;
- підтримує прийом і обробку даних у режимі реального часу;
- містить вбудовані функції моніторингу та спостереження, що полегшує відстеження продуктивності системи та діагностику проблем.

Недоліки використання Milvus:

- Складність: величезний набір функцій Milvus може ускладнити налаштування, конфігурацію та підтримку для розробників, які не знайомі з векторними базами даних;
- Відсутність вбудованої обробки людської мови: він зосереджується на пошуку схожості векторів і не пропонує вбудованих інструментів для обробки даних, що вимагає від розробника інтегрувати цю базу даних з інструментами NLP;
- Споживання ресурсів: він може споживати значні обчислювальні ресурси, особливо під час роботи з великими наборами даних або складними методами індексування.

FAISS (Facebook AI Similarity Search)

FAISS (Facebook AI Similarity Search) — це високопродуктивна бібліотека, розроблена Facebook AI Research для ефективного пошуку подібності та кластеризації векторів. Бібліотека оптимізована як для обчислень CPU, так і для GPU. FAISS добре підходить для виконання наступних задач, таких як семантичний пошук, пошук зображень і реалізації на базі засобів бібліотеки системи рекомендацій [6]. FAISS має такі параметри:

- призначена для ефективного пошуку схожості та кластеризації векторів;
- розроблена для роботи з великими даними та підтримує прискорення графічного процесора для швидшого пошуку;
- застосовується для індексування та пошуку вбудованих елементів. Її можна інтегрувати з іншими базами даних або системами для зберігання та керування даними;
- оптимізована для швидкодії та може ефективно обробляти мільярди векторів навіть на одному GPU;
- підтримує численні показники розташування (евклідова відстань, пошук подібності за рахунок знаходження косинуса кута) і типи індексів, що дозволяє точно виконувати алгоритм пошуку відповідно до своїх вимог;
- оптимізована до завдань, які вимагають пошуку подібності, наприклад семантичного пошуку, пошуку зображень і систем рекомендацій;
- розроблена та підтримується компанією Facebook AI Research, що гарантує швидке усування багів і стабільні оновлення продукту.

Недоліки:

- це лише потужна бібліотека для індексування та пошуку векторів, а не повноцінна система обробки даних. Це означає, що потрібні додаткові інструменти для зберігання та керування даними;
- розроблена лише для мов програмування Python і C++;
- має дуже широкий набір функцій і параметрів, які вимагають наявності у розробників високого порогу входження;
- Попри те, що FAISS підтримує обчислення на основі центрального процесора, він досягає найкращої продуктивності з графічним процесором, що потенційно збільшує вартість апаратного забезпечення.

Порівняльна характеристика цих систем наведена у таблиці 1.

Таблиця 1.

Особливості	**OpenAI Embeddings with plugins**	**Milvus**	**FAISS**
Початкова мета	Розуміння людської мови	Векторна база даних	Бібліотека ефективного пошуку подібності між векторами та їх кластеризація
Масштабованість	Залежить від розмірів векторної бази даних	Підтримує різні рівні масштабування	Високопродуктивна, прискорена за рахунок GPU
Підтримка NLP	Вбудовані GPT моделі	Підтримує бібліотеки з вбудованими технологіями ШІ	Підтримує бібліотеки з вбудованими технологіями ШІ
Інтеграція з ML фреймворками	Обмежена	Декілька фреймворків	Фреймворки, що підтримують Python і C++
Методи індексування	Залежить від векторної бази даних	Декілька методів	Декілька методів
Зберігання та управління даними	Залежить від векторної бази даних	Вбудована в векторну базу даних	Вимагає сторонніх баз даних
Підтримка мов програмування	Залежить від векторної бази даних	Python. Java, Go	Python, C++
Налаштування	Fine-tuning	Методи індексації	Метрики відстані, типи індексування
Ком’юніті та підтримка	Велике та активне	Активне, не велике	Активне та Facebook AI Research
Ліцензування та обмеження	Існують обмеження	Open source	Open source

Результати дослідження

Існує ряд способів створення пошуку у локальній базі даних, які виконуються за рахунок застосування алгоритмів бінарного пошуку, лінійного пошуку, бінарного дерева пошуку. Ці алгоритми ефективно застосовувати для здійснення пошуку текстових даних, які добре структуровані та впорядковані, розмір яких не надто великий та пошукові операції відносно прості [7]. Але основна операція, яка буде здійсненна між екземплярами – це порівняння. І для порівняння будуть застосовуватись – відповідні методи. Але для цього дані необхідно репрезентувати в вигляді векторів.

Будь-які текстові дані можна репрезентувати за рахунок векторів. Це можна здійснити за рахунок методу Bag of Words, Tf-Idf, векторних репрезентацій, таких як Word2Vec, Sentence2Vec або нейронних мереж, таких як Bidirectional Encoder Representation From Transformer, ELMo, GLoVe, FastText, Skip-Gram та Continuous Bag of Words [8].

У ролі локальної бази даних має виступати векторна база даних. Векторні бази даних — це спеціалізовані бази даних, призначені для ефективного зберігання, керування та створення запитів до векторних репрезентацій великої розмірності. До операцій зберігання, керування відносять: пошук подібності між векторами, кластеризацію та семантичний пошук. Деякі ключові аспекти векторних баз даних включають [9]:

1. Для покращення продуктивності пошуку векторні бази даних створюють індекси збережених векторів. Такі методи індексування, як ієрархічна кластеризація, квантування та розподіл простору, можуть значно прискорити пошук подібності та класифікація за алгоритмом найближчих сусідів.
2. Векторні бази даних створені для ефективного масштабування зі збільшенням обсягу даних.
3. Векторні бази даних часто забезпечують інтеграцію з популярними фреймворками та бібліотеками машинного навчання, такими як TensorFlow, PyTorch і scikit-learn, що забезпечує безперебійну взаємодію між базою даних і моделями машинного навчання.

Для порівняння вхідного запиту, який буде репрезентовано в вигляді вектора, з репрезентацією векторів, можна використати декілька способів, такі як косинус кута між векторами для знаходження подібності між усіма векторами [10] або евклідова відстань, скалярний добуток. У даній роботі була розроблена десктопна програма, яка використовує нейрону мережу BERT (Bidirectional Representation From Encoder) [11] для створення векторної репрезентації та збереження її в векторній базі даних; алгоритм K-Nearest Neighbours, задача якого полягає в класифікації вхідного вектора та створенні кінцевого результату, який буде наданий користувачу [12].

Для розробки даного програмного забезпечення було використано певний ряд технологій. Було обрано мову програмування Python, яка співпрацює з PyCharm [13], що дозволяє ефективно створювати додатки завдяки простоті у вивченні та можливості писати гнучкі програми. Використання Python дозволяє швидко та ефективно обробляти та аналізувати текстові дані завдяки потужним бібліотекам для обробки даних, таким як NumPy та Pandas. PyCharm також є популярним інструментом для розробки програм на Python. Він пропонує низку функцій, які полегшують написання та налагодження програмного коду на Python, а саме: автозавершення коду, підсвічування синтаксису та аналіз коду. PyCharm також інтегрується з низкою систем контролю версій, що полегшує керування змінами коду та співпрацює з іншими розробниками. Як наслідок, Python є оптимальним вибором для аналізу тексту завдяки своїй простоті, гнучкості та потужним бібліотекам.

Висновки

У роботі обгрунтована доцільність використання методів глибинного навчання, таких як BERT, ElMo, Skip-gram, Word2Vec, та алгоритмів машинного навчання, наприклад k-Nearest Neigbours, методів порівняння векторів, таких як евклідова відстань, косинус кута між векторами, для реалізації пошукової системи для локальної бази даних та застосування мови програмування Python та бібліотек NumPy та Pandas, Scikit-learn, Tensorflow, PyTorch та середовище програмування PyCharm.

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. Chat GPT Plugins [Електронний ресурс] – Режим доступу до ресурсу: https://roedigital.com/chat-gpt-plugins/
2. Milvus Introduction [Електронний ресурс] – Режим доступу до ресурсу: https://milvus.io/docs/overview.md
3. Facebook AI Similarity Search [Електронний ресурс] - Режим доступу до ресурсу: https://ai.facebook.com/tools/faiss/
4. OpenAI ChatGPT retrieval plugins [Електронний ресурс] – Режим доступу до ресурсу: https://openai.com/blog/chatgpt-plugins
5. Milvus Search Engine [Електронний ресурс] – Режим доступу до ресурсу: https://milvus.io/docs/overview.md
6. Introduction to Facebook Artificial Intelligence Similarity Search [Електронний ресурс] – Режим доступу до ресурсу: https://www.pinecone.io/learn/faiss-tutorial/
7. Search engine [Електронний ресурс] - Режим доступу до ресурсу: https://medium.com/analytics-vidhya/search-engine-using-machine-learning-and-nlp-c1ec1e28be7a
8. How to create word embeddings [Електронний ресурс] – Режим доступу до ресурсу: https://www.coursera.org/lecture/probabilistic-models-in-nlp/how-to-create-word-embeddings-a6J0B

9. What is a Vector Database [Електронний ресурс] – Режим доступу до ресурсу: https://www.pinecone.io/learn/vector-database/
10. How to represent data as a vector [Електронний ресурс] – Режим доступу до ресурсу: https://towardsdatascience.com/why-data-is-represented-as-a-vector-in-data-science-problems-a195e0b17e99
11. BERT [Електронний ресурс] – Режим доступу до ресурсу: https://huggingface.co/docs/transformers/model_doc/bert
12. K-Nearest Neighbours [Електронний ресурс] – Режим доступу до ресурсу: https://towardsdatascience.com/k-nearest-neighbors-knn-algorithm-23832490e3f4
13. PyCharm guide [Електронний ресурс] – Режим доступу до ресурсу: https://realpython.com/pycharm-guide/

Довгань Олексій Андрійович *– студент групи 1АКІТ-19б, кафедра Автоматизації та інтелектуальних інформаційних технологій, факультет інтелектуальних інформаційних технологій та автоматизації, Вінницький національний технічний університет, м.Вінниця, e-mail: odovhan08@gmail.com*

Паламарчук Євгеній Анатолійович *– професора кафедри, к.т.н., доц. автоматизації та інтелектуальних інформаційних технологій, факультет комп'ютерних систем і автоматики, Вінницький національний технічний університет, м.Вінниця, e-mail: p@vntu.edu.ua*

Dovhan Oleksii Andriovich *– student of 1ACIT-19B group, Department of Automation and Intelligent Information Technologies, Faculty of Intelligent Information Technology and Automation, Vinnytsia National Technical University, Vinnytsia, e-mail: odovhan08@gmail.com*

Palamarchuk Yevhenii Anatoliyovych - *professor of the department of Automation and Intelligent Information Technologies, Faculty of Computer Systems and Automatics Vinnytsia National Technical University, Vinnytsia, e-mail: p@vntu.edu.ua*