УДК 004.89

АВТОМАТИЗАЦІЯ ТА СПРОЩЕННЯ ПОШУКУ ВОЛОНТЕРСЬКОЇ ДОПОМОГИ З ВИКОРИСТАННЯМ СЕМАНТИЧНОГО ПОШУКУ

Власенко Андрій Андрійович, Майорова Анастасія Романівна
к.т.н., доцент каф. ШІАД Годовиченко Микола Анатолійович
Національний університет «Одеська політехніка», УКРАЇНА

АНОТАЦІЯ. Розроблено та протестовано сервіс AidBot для пошуку волонтерської допомоги з використанням семантичного пошуку у вигляді телеграм-боту. Обраховано вартість та точність використання текстових ембедингів у даній задачі.

Вступ. З початком війни, багато людей залишилось без домівок, особистих речей чи документів. Такі люди сильно потребують швидкого пошуку допомоги від волонтерів, а волонтерам треба шукати постраждалих та людей, які можуть надати їм необхідні ресурси. На сьогодні такий пошук є цілком ручною роботою, тому займає багато часу та зусиль, деякі заявки можуть залишитися необробленими чи загубленими, тощо. З використанням сучасних технологій штучного інтелекту у галузі *Natural Language Processing* можна створити сервіс, що автоматизує та спростить обробку та пошук заявок чи пропозицій з використанням текстових ембедингів та векторної бази даних. Така технологія семантичного пошуку задовольнить потреби багатьох постраждалих та волонтерських організацій.

Мета роботи. Метою роботи є зв'язування волонтерських організацій, людей, які потребують допомоги та людей, які можуть цю допомогу надати між собою. Це зв’язування забезпечує волонтерським організаціям зменшення трудовитрат, потребуючим та бажаючим допомогти - зменшення часу на з’єднання. Така система має забезпечити точність не менше 60% *top-5 accuracy* коштуючи не більше $0.5/тисячу запитів.

Основна частина роботи.

Реалізація сервісу для семантичного пошуку складається з декількох частин:

–збір даних;

–обробка даних з формату текста до формату ембедингів;

–створення та заповнення векторної бази даних;

–створення клієнтської частини та реалізація логіки її взаємодії з базою даних.

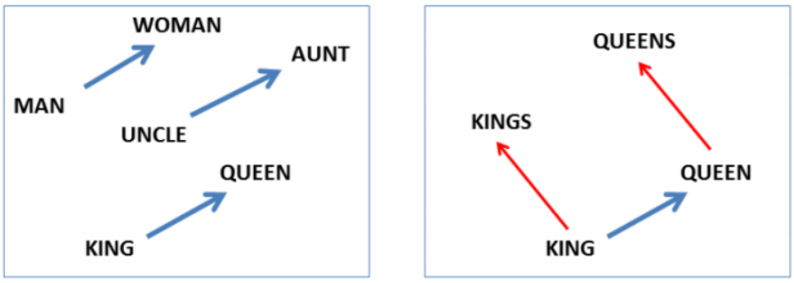
В нашому контексті, *embedding* – зіставлення слову чи тексту певного вектора, відображення його у “просторі змісту”, в такому форматі його можна ефективно хранити у векторній базі даних та виконувати пошук за змістом.

Класичний метод створення ембедингів полягає у побудові матриці сумісності усіх слів у корпусі текстів, та декомпозиції її на три матриці *U, S* та *V,* де *U* та *V* – ортогональні матриці. Але цей метод заснований на гіпотезі «схожі слова часто зустрічаються в одному контексті», тобто поряд друг з другом, та є дуже обмеженим. У 2013 році Томаш Миколов запропонував новий підхід *word2vec* [1], що спирається на гіпотезу локальності – «схожі слова часто зустрічаються у схожих контекстах», тобто не разом один з одним, а взаємозамінно.

Методи засновані на гіпотезі локальності можуть ставити задачу передбачати контекст за словом – *skip-gram*, або слово за контекстом – *continuous bag of words*. Наприклад *word2vec* передбачає саме ймовірність слова за його контекстом:

$$P(w_o|w_c) = \frac{e^{s(w_o,w_c)}}{\Sigma_{w_i \in V} e^{s(w_o,w_c)}}, \quad (1)$$

де w_o – вектор певного слова, w_c – вектор контекста, певним методом обрахований з векторів слів контекста, а s – це функція, що зіставляє двум векторам одне число. Ця формула використовує функцію *softmax*, тобто «м’який максимум». В процесі тренування модель вивчає функцію ймовірності s та вектори саме так, щоб вони передбачали слово з вибірки тренування.



(Mikolov et al., NAACL HLT, 2013)

Рисунок 1 – Натренерована модель Томаша Микола [1]

Після тренування вивченні вектори слів співвідносяться між собою зберігаючи свої семантичні відносини як зображено на рисунку.

Для тренування ембедингів для тексту, а не окремих слів використовуються схожі методи. Наприклад, в статі [2] від *OpenAI,* для отримання вектору з тексту використовується архітектура *Transformer*, в процесі тренування шматки тексту, що розташовані близько один до одного, вважаються близькими за змістом, та додатково запроваджено contrastive *loss* – тексти розташовані далеко один від одного вважаються далекими за змістом. Ця архітектура використовується в застарілому *API* ембедингів від *OpenAI* – ми використовуємо більш нове *API text-embedding-ada*-002 в якості нашого головного інструмента, воно не розкриває деталей реалізації, але скоріш за все використовує схожі методи.

Ембединги, згенеровані моделями штучного інтелекту є багатовимірними, що ускладнює керування ними, тому для них використовуються спеціалізовані векторні бази даних, такі як розширення *PG-Vector для Postgres*, що пропонують можливості традиційної бази даних разом з оптимізованим зберіганням та пошуком ембедингів.

Нижче наведена діаграма (рис. 2) дає нам краще уявлення про роль векторних баз даних.

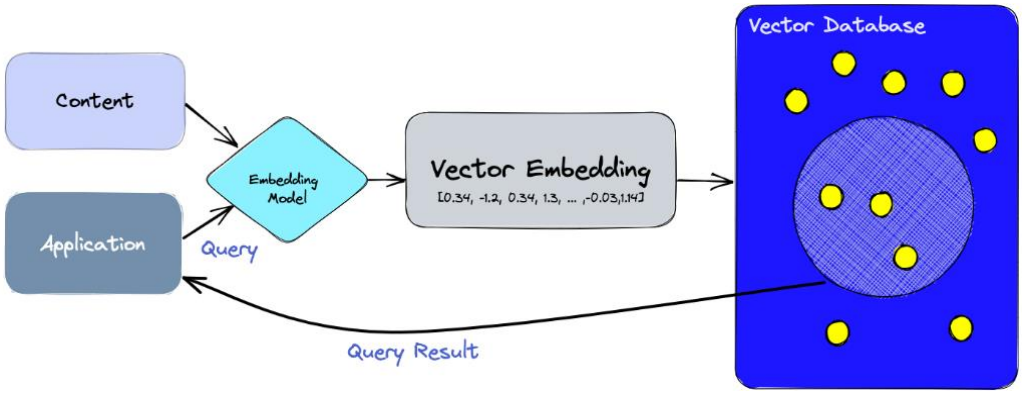


Рисунок 2 – Векторна база даних. Обробка даних [2]

Спочатку створюємо ембединг для контенту та зберігаємо його у векторній базі даних разом з посиланням на контекст. Коли наш додаток отримує запит, він використовує ту ж модель отримання ембедингів щоб отримати контент зі схожими ембедингами з бази даних.

Для пошуку схожих ембедингів використовується формула *cosine similarity*

$$Cosine\ Similarity = 1 - \frac{A \cdot B}{||A||\,||B||} = 1 - \frac{\Sigma_i A_i B_i}{\sqrt{\Sigma_i A_i^2}\sqrt{\Sigma_i B_i^2}}. \quad (2)$$

Вона визначає кут між векторами – головним вважається співвідношення між різними концептами у ембедингу, а не їхня сила. Але обрахування такої функції для усіх ембедингів у великій базі даних є дуже довгим. Для пришвидшення пошуку ми використовуємо індекси для

приблизного пошуку. В роботі використано *ivfflat* вбудовану в *pg-vector*. Ця індексація використовує *KNN* щоб розбити базу даних на дерево кластерів, в якому пошук працює набагато швидше.

Для збору даних, клієнтської частини та взаємодії з базою даних ми використали *Python*.

Було підраховано річні витрати для обробки мільйону запитів на місяць та оцініно вартість зарплатні для працівників щоб проводити таку ж роботу з обробки запитів (табл. 1). Результат показав, що розроблений сервіс дозволяє значно зекономити, в той же час спрощуючи використання, запуск та масштабування. Таким чином, було отримано вартість $0.25/тисячу запитів.

Таблиця 1 – Порівняльні характеристики методів пошуку допомоги

Характеристика	AidBot	Ручна обробка
Вартість	3000$/рік	~22000$/рік
Швидкість	1-2 с	~3600 с
Складність використання	Простий запит до телеграм боту	Листи/повідомлення до реальних людей
Складність запуску	Автоматизований збір з різних джерел	Найм та навчання людей, довгий набір матеріалів

Також порахували *top-5 accuracy* на 100 реальних запитів на допомогу з використанням *ChatGPT* та отримали точність 71±8 %

Напрямки покращення якості результату включають у себе: явне використання геолокації користувачів, порівняння якості для інших типів ембедингів, використання *ChatGPT* для покращення точності взаємодії користувача з ботом.

Висновки. Було розроблено сервіс, що зв'язує волонтерські організацій, людей, які потребують допомоги та людей, які можуть надати допомогу за допомогою семантичного пошуку. Отримано *top-5 accuracy* точність близько 71% при вартості $0.25/тисячу запитів, тому створений продукт може значно зменшити трудовитрати волонтерських організацій та допомогти потребуючим отримати допомогу. Також було запропоновано подальші напрямки розвитку, які планується реалізувати у нподальших розробках.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Tomas Mikolov, Kai Chen, Greg Corrado, Jeffrey Dean, Efficient Estimation of Word Representations in Vector Space (2013). arXiv:1301.3781
2. Arvind Neelakantan et al, Text and Code Embeddings by Contrastive Pre-Training (2022). arXiv: 2201.10005

AUTOMATION AND SIMPLIFICATION OF SEARCH FOR VOLUNTEER HELP USING SEMANTIC SEARCH

Andrii Vlasenko, Anastasiia Maiorova
PhD, Associate Professor of the department of AIDA Mykola Hodovychenko
Odesa Polytechnic National University, UKRAINE

ANNOTATION. AidBot service for finding volunteer help using semantic search in the form of a Telegram bot was developed and tested. The cost and accuracy of using text embeddings in this task were evaluated.