Kochuieva, Z. A., & Babkova, N. V. (2024). Using neural networks in development of intelligent chatbots. *Actual Issues of Modern Science. European Scientific e-Journal, 28*, 96-105. Ostrava: Tuculart Edition, European Institute for Innovation Development. (In Ukrainian)

DOI: 10.47451/inn2024-02-01

The paper is published in Crossref, ICI Copernicus, BASE, Zenodo, OpenAIRE, LORY, Academic Resource Index ResearchBib, J-Gate, ISI International Scientific Indexing, ADL, JournalsPedia, Scilit, EBSCO, Mendeley, and WebArchive databases.

Zoia A. Kochuieva, Candidate of Technical Sciences (Ph.D.), Associate Professor, Department of Intelligent Computer Systems, National Technical University "Kharkiv Polytechnic Institute". Kharkiv, Ukraine. ORCID 0000-0002-4300-3370, Scopus 57223796440

Nadiia V. Babkova, Candidate of Technical Sciences (Ph.D.), Associate Professor, Department of Intelligent Computer Systems, National Technical University "Kharkiv Polytechnic Institute". Kharkiv, Ukraine. ORCID 0000-0002-2200-7794, Scopus 57204916728

Using neural networks in development of intelligent chatbots

Abstract: Recently, there has been a strong increase in the volume of publicly available information. A huge amount of both structured and unstructured data is available to us via the Internet. As a result, there is a problem of finding and obtaining the information we need. This task of searching and processing information is further enhanced by the fact that information on the Internet is constantly changing and updating, it has a high level of dynamics. At every moment of time, new materials and new facts appear. The volume of information arrays is constantly increasing, as a result of which constant accounting of information is required, which is often impossible. The presented article deals with one of the current problems – the development of a prototype of an intelligent chatbot based on the application of artificial neural network algorithms. The need to consider the topic is due to the fact that such a direction as the use of chatbots for various areas of a person's life is gaining popularity. The types of question-and-answer systems and intelligent chatbots are considered within the scope of the work. The use of a recurrent neural network for the implementation of an intelligent chatbot is substantiated. A recurrent neural network learning algorithm has been developed. Selected material for training recurrent neural network. A neural network was built to implement a chatbot.

Keywords: recurrent neural network, chatbot, question-and-answer system, root mean square error, machine learning method, intelligent data analysis.

Зоя Анатоліївна Кочуєва, кандидат технічних наук, доцент, кафедра інтелектуальних комп'ютерних систем, Національний технічний університет «Харківський політехнічний інститут». Харків, Україна. ORCID:0000-0002-2200-7794, Scopus 57223796440

Надія Вікторівна Бабкова, кандидат технічних наук, доцент, кафедра інтелектуальних комп'ютерних систем, Національний технічний університет «Харківський політехнічний інститут». Харків, Україна. ORCID:0000-0002-2200-7794, Scopus 57204916728

Використання нейронних мереж в розробці інтелектуальних чат-ботів

Анотація: В последнее время наблюдается сильный рост объема общедоступной информации. Огромное количество как структурированных, так и неструктурированных данных доступно нам с помощью сети Интернет. Вследствие этого существует проблема поиска и получения

необходимой нам информации. Эта задача поиска и обработки информации что больше усиливается тем, что информация в интернете находится в постоянном изменении и обновлении, она имеет высокий уровень динамики. В каждый момент времени появляются новые материалы, новые факты. Объем информационных массивов постоянно увеличивается, вследствие этого требуется постоянный учет информации, что зачастую невозможно. В представленій статті розглядається одна з актуальних проблем – розробка прототипу інтелектуального чат-боту заснованого на застосуванні алгоритмів штучних нейронних мереж. Необхідність розгляду теми зумовлена тим, що зараз набуває популярності такий напрям як використання чат-ботів для різних напрямів життя та людини. В межах роботи розглянуто типи питально-відповідних систем та інтелектуальних чат-ботів. Обґрунтовано використання рекурентної нейронної мережі для реалізації інтелектуального чат-боту. Розроблено алгоритм навчання рекурентної нейронної мережі. Підібрано матеріал для навчання рекурентної нейронної мережі. Побудовано нейронну мережу для реалізації чат-боту.

Ключові слова: рекурентна нейрона мережа, чат-бот, питально-відповідна система, середньоквадратична помилка, метод машинного навчання, інтелектуальний аналіз даних.

Вступ

Останнім часом спостерігається сильне зростання об’єму загальнодоступної інформації. Величезна кількість як структурованих, так і неструктурованих даних доступна нам за допомогою мережі Інтернет. Унаслідок цього існує проблема пошуку і отримання необхідної нам інформації. Це завдання пошуку і обробки інформації ще більше посилюється тим, що інформація в Інтернеті знаходиться в постійній зміні і оновленні, вона має високий рівень динаміки. У кожен момент часу з’являються нові матеріали, нові факти. Об’єм інформаційних масивів постійно збільшується, внаслідок цього потрібен постійний облік інформації, що часто є неможливим.

Нині для користувача мережою Інтернет доступні системи інформаційного пошуку. Ці системи вимагають запиту, сформульованого з ключових слів, які відповідають текстовій інформації, потрібній для користувача. Необхідно відмітити, що часто ці системи не враховують порядок слів, їх форми і зв'язки між самими словами, тобто розглядають запит, просто як невеликий набір слів. Тоді як людині більш властиво формулювати запити у формі питання на природній мові. В результаті роботи пошукових систем видається велика кількість посилань і текстових фрагментів в порядку релевантності, тобто подальший пошук інформації повинен вести сам користувач, що може утруднити її сприйняття і збільшити час пошуку. Коли ми хочемо щось упізнати, ми запитуємо – ставимо питання, що, загалом, і природно в процесі пізнання. Більшість систем по пошуку інформації, не мають можливості відповідати на наші питання. Для пошуку і отримання людині треба сформувати запит з ключових слів і задати його пошуковій машині.

Мета даної роботи – огляд проблем та методів розробки питально-відповідних систем взагалі та інтелектуальних чат-ботів зокрема. У результаті буде отримано алгоритм та матеріал для розробки інтелектуального чат-боту.

Основна частина

Аналіз проблемної галузі

Підходи до реалізації і відповідно принципи побудови питально-відповідних систем можна розділити на наступні декілька груп (*Allam et al, 2012*):

- питально-відповідні системи, що базуються на веб-пошуку (англ. "web-based question answering system");
- питально-відповідні системи з власною розміченою колекцією документів;
- питально-відповідні системи з базою даних, що містить питання і відповіді;
- питально-відповідні системи експертного типу.

Огляд існуючих розробок питально-відповідних систем

Більшість існуючих на даний час реалізацій питально-відповідних систем орієнтовані на одну з найпоширеніших мов світу – англійську.

На сьогоднішній день однією з найрозвиненіших і відомих питально- відповідь них систем є питально-відповідна система, створена групою розробників фірми IBM (керівник групи David Ferrucci) на суперкомп'ютері IBM Watson (*Magnini et al, 2001*). У 2011 році Watson взяв участь у телепередачі «Jeopardy!», обігравши двох кращих гравців «Jeopardy!». Виграш комп'ютера склав 1 млн. доларів, в той час, як його суперники-люди отримали по 200 і 300 тисяч доларів відповідно. Під час гри система мала доступ до інформації (у тому числі до повного тексту Вікіпедії) обсягом у 4 терабайта.

На першому етапі роботи Watson відбувається аналіз питального речення: виділяється фокус питання, питання класифікується відповідно до внутрішньої класифікації Watson (*Ceglarek, 2014*). Після цього відбувається декомпозиція питання: при необхідності питання розбивається на кілька простіших. Потім системою генеруються гіпотези – фрази з корпусу текстів, які з певною ймовірністю можуть містити відповідь на поставлене користувачем питання. Цей корпус складається з безлічі всіляких структурованих і неструктурованих знань, таких як: підручники, новини, наукові статті та, в тому числі, текст Вікіпедії, DBpedia і ін. Після того, як Watson згенерував безліч гіпотез, частина відсівається за допомогою «м'якого фільтра», який залишає тільки 100 гіпотез, найбільш релевантних питанню. На наступному етапі відбувається оцінка кожної гіпотези, яка залишилася, на релевантність питання. Для цього у системі використовуються так звані «докази»: в базах знань шукаються речення, які підтвердять гіпотезу (*Mikolov et al, 2013*). Гіпотеза вбудовується у структуру питання і отримане речення шукається у базах. Кожна гіпотеза отримує набір оцінок, які показують наскільки конкретна гіпотеза відповідає тому чи іншому доказу.

Кожній такій оцінці за допомогою статистичної моделі ставиться у відповідність певний коефіцієнт, який показує наскільки важливий для відповіді на питання даний доказ (згодом цей коефіцієнт буде використовуватися для підрахунку впевненості системи у фінальній відповіді). На наступному етапі за допомогою машинного навчання (машинне навчання організовано на корпусі питань з відомими відповідями) відбувається вибір єдиної відповіді, яка надається користувачу з величиною, що позначає ступінь впевненості машини у правильності відповіді (*Patel et al, 2019*).

Широку популярність також отримала питально-відповідна система Lasso, яка була розроблена у лабораторії комп'ютерної лінгвістики Південного Методичного університету, штат Даллас, США (*Baby et al, 2017*). Архітектура системи LASSO складається з трьох основних модулів:

- модуль обробки питання;
- модуль індексації абзаців;
- модуль обробки відповіді.

Модуль обробки питання визначає:

- тип поставленого питання ("what-who", "what-when", "how-long", "how rich", тощо);
- тип очікуваної відповіді ("DATE", "LOCATION", "PERSON" й т.п.);
- фокус питання (фокус питання визначається як основна інформація, запитувана питальним реченням) (*Manning et al, 2009*).

Модуль обробки питання також визначає ключові слова запиту, які повинні бути передані модулю індексації даних (*Рисунок 1*)

Для індексації документів в Lasso використовується пошукова система Zprise IR System. Також модуль індексації абзаців проводить оцінку якості знайдених абзаців. У разі визнання якості задовільним, виробляється їх упорядкування відповідно до ступеня правдоподібності змісту відповіді, в іншому випадку відбувається додавання або видалення деяких ключових слів пошуку, після чого пошук за оновленим списком ключових слів відновлюється і система повторно оцінює якість знайдених абзаців (*Lee et al, 2020*). Модуль перевірки якості абзаців дозволяє розумно зменшити кількість тексту для надсилання модулю відповіді на питання.

Обґрунтування вибору мови програмування

Для реалізації проекту було використано мову програмування Python. Python проста у використанні, та водночас повноцінна мова програмування, що надає набагато більше засобів для структурування і підтримки великих програм, ніж shell (*Su et al, 2017*). З іншого боку, вона краще за C/C++ обробляє помилки, і, будучи мовою дуже високого рівня, має вбудовані типи даних високого рівня, такі як гнучкі масиви і словники, ефективна реалізація яких на C/C++ потребує значних витрат часу(*Cromieres et al, 2016*).

Результати дослідження

В межах дослідження в якості результатів було розроблено алгоритм та програмну реалізацію прототипу чат-боту.

В загальному випадку алгоритм навчання рекурентної нейронної мережі складається з наступних кроків:

1) в початковий момент часу всі контекстні нейрони встановлюються в нульовий стан, тобто значення їх виходу дорівнює 0;
2) вхідний вектор навчання подається до нейронної мережі і здійснюється етап прямого розповсюдження інформації;
3) здійснюється модифікація вагових коефіцієнтів та порогів всіх рівнів згідно з алгоритмом зворотного розповсюдження помилки;

4) обирається наступний момент часу $(t+1)$ і здійснюється перехід до п. 2. Процес навчання закінчується, коли сумарна середньоквадратична помилка стане меншою заданого значення.

Для навчання рекурентної нейронної мережі використано алгоритм зворотного розповсюдження помилки (*Scott et al, 2001*). Середньоквадратична помилка навчання для одного вектора навчання:

$$E(t) = \frac{1}{2}\big(y(t) - d(t)\big)^2.$$

Для налаштування параметрів навчання рекурентної нейронної мережі:

$$\Delta V_i(t+1) = -\alpha\big(y(t) - d(t)\big) \times y_j(t),$$
$$\Delta T(t+1) = \alpha\big(y(t) - d(t)\big),$$
$$\Delta W_{ij}(t+1) = -\alpha\big(y(t) - d(t)\big) \times V_j y_j'(t) \times x_i,$$
$$\Delta W_i(t+1) = -\alpha\big(y(t) - d(t)\big) \times V_j y_j'(t) \times y_k(t-1),$$
$$\Delta W_{kj}(t+1) = -\alpha\big(y(t) - d(t)\big) \times V_j y_j'(t) \times y_k(t-1),$$
$$\Delta T_j(t+1) = \alpha\big(y(t) - d(t)\big) \times V_i \times y_i'(t).$$

В якості функції активації нейронів схованого рівня може застосовуватись функція гіперболічного тангенсу або сигмоїдна функція (*Mulik et al, 2021*). При використанні сигмоїдної функції:

$$y_j'(t) = y_j(t) \times \Big(1 - y_j(t)\Big).$$

При використанні функції гіперболічного тангенсу:

$$y_j'(t) = 1 - y_j^2(t).$$

Для прискорення процесу навчання доцільно застосовувати адаптивний крок навчання. Для вихідного рівня нейронної мережі (яка містить один нейрон з лінійною функцією активації):

$$\alpha_1(t) = \frac{1}{1 + \sum_j^m y_j^2(t)}.$$

Для схованого рівня нейронів вираз для адаптивного кроку навчання при використанні сигмоїдної функції активації:

$$\alpha_2(t) = \frac{4}{R^2 \times \sum_j^m V_j^2 \times y_j(t) \times \Big(1 - y_j(t)\Big)}$$
$$R = 1 + \sum_i^m x_i^2(t) + y^2(t-1) + \sum_k^m y_k^2(t-1).$$

При використанні логарифмічної функції активації адаптивний крок навчання для обчислення зміни вагових коефіцієнтів схованого рівня (*Gnewuch et al, 2022*):

$$\alpha_2(t) = \frac{1}{R^2 \times \sum_j^m V_j^2 \times \Big(1 - y_j^2(t)\Big)}.$$

В межах розробки програмного забезпечення прототипу інтелектуального чат-боту» було створено 4 класи: chat, chatbotlstmtrain, chatbotPreprocessing, chatFrame. Кожний з

цих класів містить методи, що дозволяють побудувати нейронну мережу, обробити інформацію та вивести її на екран.

Клас chatbotPreprocessing відповідає за первинну обробку а також за розподілення корпусу на питання та відповіді (*Рисунок 1*).

Далі відбувається токенізація отриманих питань та відповідей. Після цього програма знаходить вектор для кожного слова (*Рисунок 2*).

Після аналізу повідомлень було вирішено обрати максимальну кількість слів 15. У зв'язку з цим, якщо кількість слів у реченні менша за максимальну то додається вектори з набору одиниць для отримання максимальної кількості слів. Якщо кількість слів більша за максимально допустиму то відбувається відкидання усіх наступних токенів.

Клас chatbotlstmtrain відповідає за навчання нейронної мережі. Для вирішення завдання розробки прототипу чат-боту було вирішено розробити нейронну мережу яка складається з 4 прошарків. Кожен прошарок складається з 300 нейронів. Активаційна функція на прошарках сигмоїдальна (*Рисунок 3*).

Для подальшого використання моделі було збережено декілька моделей з різною кількістю тренувань (*Рисунок 4*).

Клас chat призначений для основної функції – підтримання бесіди. Користувачу пропонується ввести повідомлення. Після цього програма обробляє його та отримує повідомлення у вигляді вектору. Кількість слів у введеному повідомленні не повинна перевищувати 15 слів. Потім кожному слову знаходиться найбільш підходяще слово із моделі та виводиться відповідь на екран(*Рисунок 5*).

Клас ChatFrame служить як графічний інтерфейс для приємнішого користування програмою. Метод __init__ використовується для завантаження натренованої моделі word2vec та векторного представлення слів. Метод initUI служить для отримання основного вікна програми. Вікно було розроблено за допомогою бібліотеки PyQT написаної на мові Python (*Рисунок 6*)

За допомогою методу send_message відбувається відправка повідомлення для обробки та для виводу її на екран(*Рисунок 7*).

Розроблений програмний застосунок використовує натренований word2vec корпус ap_news, який складається з 1,5 мільйона векторів розмірності 300 (*Рисунок 8*).

Для навчання нейронної мережі було використано декілька чатів за допомогою яких прототип чат-боту відповідає користувачеві (*Рисунок 9*).

В роботі представлено проєктування та основні програмної реалізації прототипу чат-боту. Проаналізувавши вищезгаданий матеріал було створено діаграму класів (*Рисунок 10*).

Наступним важливим етапом є розробка діаграми варіантів використання (*Adamopoulou et al, 2020*). Основна мета створення будь-якої програмної системи це створення програмного продукту, який допомагає користувачу виконувати свої повсякденні завдання. Для створення таких програм насамперед визначаються вимоги, яким повинна задовольняти система. Проте якщо дати користувачам написати ці вимоги на папері, то часто можна одержати список функцій, по якому важко судити чи буде майбутня система виконувати своє призначення і чи зможе вона полегшити користувачу виконання його роботи взагалі. Діаграма варіантів використання наведена на рисунку (*Рисунок 11*).

Дискусія

У своїй роботі ми провели аналіз існуючих методів та підходів розробки питально-відповідних систем, розглянули їх типи та типи інтелектуальних чат- ботів. Унікальні можливості Інтернет такі, як швидкість, оперативність, доступність комунікації між користувачами – дозволяють використовувати мережу як засіб спілкування і створювати інтерактивні форми спілкування: чати, форуми, телеконференції, електронну пошту та інші. На зміну реальним співрозмовникам приходять програми штучного інтелекту, такі як чати, консультанти, помічники, розважальні програми та інші. Але, на відміну від розмови людей, програма не володіє гнучким розумовою інтелектом. На жаль, сучасні віртуальні співрозмовники лише частково вирішують питання імітації розмови людини. Словниковий запас більшості віртуальних співрозмовників обмежений, крім цього, у них відсутня емоційне забарвлення, тембр голосу тощо. Тому більшість віртуальних співрозмовників запрограмовані на ведення нескладної бесіди.

Висновки

Поставлена мета дослідження була досягнута вирішенням таких завдань:

- оглянуто існуючі методи та підходи питально-відповідних систем;
- розглянуто типи питально-відповідних систем та інтелектуальних чат-ботів;
- обґрунтовано використання рекурентної нейронної мережі для реалізації інтелектуального чат-боту;
- розроблено алгоритм навчання рекурентної нейронної мережі;
- підібрано матеріал для навчання рекурентної нейронної мережі.

Список джерел інформації:

Adamopoulou, E., & Moussiades, L. (2020). An overview of chatbot technology. *Artificial Intelligence Applications and Innovations: 16th IFIP WG 12.5 International Conference*, *16*, 373-383.

Allam, A. M. N. & Haggag, M. H. (2012). The Question Answering Systems: A Survey. *International Journal of Research and Reviews in Information Sciences*, *2*(3), 10-21.

Baby, C. J., Khan, F. A., & Swathi, J. N. (2017) Home automation using IoT and a chatbot using natural language processing. *Proceedings of the 2017 Innovations in Power and Advanced Computing Technologies (i-PACT)*, 1-6.

Ceglarek, D. (2014). Semantic Compression for Text Document Processing. *Transactions on Computational Collective Intelligence*, *14*, 20-48.

Cromieres, F. (2016) Kyoto-NMT: A neural machine translation implementation in chainer. *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: System Demonstrations*, 307-311.

Gnewuch, U., Morana, S., Adam, M., & Maedche, A. (2022). Opposing effects of response time in human-chatbot interaction: The moderating role of prior experience. *Business & Information Systems Engineering*, *64*(6), 773-791.

Lee, M.-C., Chiang, S.-Y., Yeh, S.-C., & Wen, T.-F. (2020). Study on emotion recognition and companion chatbot using deep neural network. *Multimedia Tools and Applications*, *79*, 19629-

19657.

Magnini, B., Negri, M., Prevete, R., & Tanev, H. (2001). Multilingual question answering: The DIOGENE system. *10th Text Retrieval Conference*, 433-450. Italy: Centro per la Ricerca Scientifica e Tecnologica Via Sommarive.

Manning, C., Raghavan, P., & Schütze, H. (2009). *An Introduction to Information Retrieval*. Cambridge: Cambridge University Press.

Mikolov, T., Sutskever, I., Chen, K., & Corrado, G. (2013). Distributed representations of words and phrases and their compositionality. *Computation and Language*. Cornell University.

Mulik, S., & Bhosale, V. (2021). Application of NLP: Design of chatbot for new research scholars. *Turkish Online Journal of Qualitative Inquiry*, *12*(8), 2817-2823.

Patel, F., Thakore, R., Nandwani, I., & Bharti, S. K. (2019). Combating Depression in Students Using an Intelligent ChatBot: A Cognitive Behavioral Therapy. *In Proceedings of the 2019 IEEE 16th India Council International Conference (INDICON)*, *IEEE*, 1-4.

Question-Answer Dataset. (2023). https://kaggle.com/rtatman/questionanswerdataset

Scott, S., & Gaizauskas, R. (2001). QA-LaSIE: A natural language question answering system. *Proceedings of the 14th Biennial Conference of the Canadian Society on Computational Studies of Intelligence*, 172-182.

Su, M.-H., Wu, C.-H., Huang, K.-Y., Hong, Q.-B., & Wang, H.-M. (2017). A chatbot using LSTM-based multi-layer embedding for elderly care. *In Proceedings of the 2017 International Conference on Orange Technologies (ICOT)*, 70-74.

Додатки

```
model = gensim.models.Word2Vec.load('train/word2vec.bin');
file=open('train/conversation.json');
data = json.load(file, strict=False)
cor=data["conversations"];

x=[]
y=[]

for i in range(len(cor)):
    for j in range(len(cor[i])):
        if j<len(cor[i])-1:
            x.append(cor[i][j]);
            y.append(cor[i][j+1]);
```

Рисунок 1. Розподілення корпусу на питання та відповіді

```
tok_x=[]
tok_y=[]
for i in range(len(x)):
    tok_x.append(nltk.word_tokenize(x[i].lower()))
    tok_y.append(nltk.word_tokenize(y[i].lower()))

sentend=np.ones((300))

vec_x=[]
for sent in tok_x:
    sentvec = [model[w] for w in sent if w in model.wv.vocab]
    vec_x.append(sentvec)

vec_y=[]
for sent in tok_y:
    sentvec = [model[w] for w in sent if w in model.wv.vocab]
    vec_y.append(sentvec)
```

Рисунок 2. Пошук векторів для кожного слова

```
model = Sequential()
model.add(LSTM(output_dim=300, input_shape=x_train.shape[1:], return_sequences=True,
               init='glorot_normal',inner_init='glorot_normal', activation='sigmoid'))
model.add(Dense(300, input_shape=x_train.shape[1:], activation='sigmoid'))
model.add(LSTM(output_dim=300, input_shape=x_train.shape[1:], return_sequences=True,
               init='glorot_normal', inner_init='glorot_normal', activation='sigmoid'))
model.add(LSTM(output_dim=300, input_shape=x_train.shape[1:], return_sequences=True,
               init='glorot_normal', inner_init='glorot_normal', activation='sigmoid'))
model.compile(loss='cosine_proximity', optimizer='adam', metrics=['accuracy'])
```

Рисунок 3. Нейронна мережа

```
model.fit(x_train, y_train, nb_epoch=500, validation_data=(x_test, y_test))
model.save('train/LSTM500.h5');
model.fit(x_train, y_train, nb_epoch=500, validation_data=(x_test, y_test))
model.save('train/LSTM1000.h5');
model.fit(x_train, y_train, nb_epoch=500, validation_data=(x_test, y_test))
model.save('train/LSTM1500.h5');
```

Рисунок 4. Збереження моделей

```
x = input("Enter the message:");
sentend = np.ones((300))

sent = nltk.word_tokenize(x.lower())
sentvec = [mod[w] for w in sent if w in mod.wv.vocab]

sentvec[14:] = []
sentvec.append(sentend)
if len(sentvec) < 15:
    for i in range(15 - len(sentvec)):
        sentvec.append(sentend)
sentvec = np.array([sentvec])

predictions = model.predict(sentvec)
outputlist = [mod.most_similar([predictions[0][i]])[0][0] for i in range(15)]
output = ' '.join(outputlist)
```

Рисунок 5. Вивід відповіді на екран

```
def __init__(self, parent=None):
    global model
    global mod
    model = load_model('train/LSTM5000.h5')
    mod = gensim.models.Word2Vec.load('train/word2vec.bin')

    super().__init__(parent)
    self.initUI()

def initUI(self):
    self.c_w = chat_window(self)
    self.c_w.resize(600, 400)
    self.c_w.move(10, 10)
    self.c_w.setReadOnly(True)

    self.e_t = enter_text(self)
    self.e_t.resize(500, 25)
    self.e_t.move(10, 420)

    self.send = QPushButton('Send', self)
    self.send.resize(90,25)
    self.send.move(520, 420)
    self.send.clicked.connect(self.send_message)

    colorborder = 'Gray'
    colorline = 'White'
```

Рисунок 6. Методи __init__ та initUI

```
def send_message(self):
    global model
    global mod

    x = self.e_t.text()
    self.c_w.append('YOU: ' +x)
    print(x)
    self.e_t.setText('')
    sentend = np.ones((300))

    sent = nltk.word_tokenize(x.lower())
    sentvec = [mod[w] for w in sent if w in mod.wv.vocab]

    sentvec[14:] = []
    sentvec.append(sentend)
    if len(sentvec) < 15:
        for i in range(15 - len(sentvec)):
            sentvec.append(sentend)
    sentvec = np.array([sentvec])

    predictions = model.predict(sentvec)
    outputlist = [mod.most_similar([predictions[0][i]])[0][0] for i in range(15)]
```

Рисунок 7. Метод send_message

word2vec.bin	29.06.2016 8:58	AceStream medi...	23 617 КБ
word2vec.bin.syn0.npy	29.06.2016 8:58	Файл "NPY"	346 895 КБ
word2vec.bin.syn1neg.npy	29.06.2016 8:58	Файл "NPY"	346 895 КБ

Рисунок 8. Word2vec модель

```
"conversations": [
    [
        "Good morning, how are you?",
        "I am doing well, how about you?",
        "I'm also good.",
        "That's good to hear.",
        "Yes it is."
    ],
    [
        "Hello",
        "Hi",
        "How are you doing?",
        "I am doing well.",
        "That is good to hear",
        "Yes it is.",
        "Can I help you with anything?",
        "Yes, I have a question.",
        "What is your question?",
        "Could I borrow a cup of sugar?",
        "I'm sorry, but I don't have any.",
        "Thank you anyway",
        "No problem"
    ],
    [
        "How are you doing?",
        "I am doing well, how about you?",
        "I am also good.",
        "That's good."
    ],
```


Рисунок 9. Діалоги для навчання

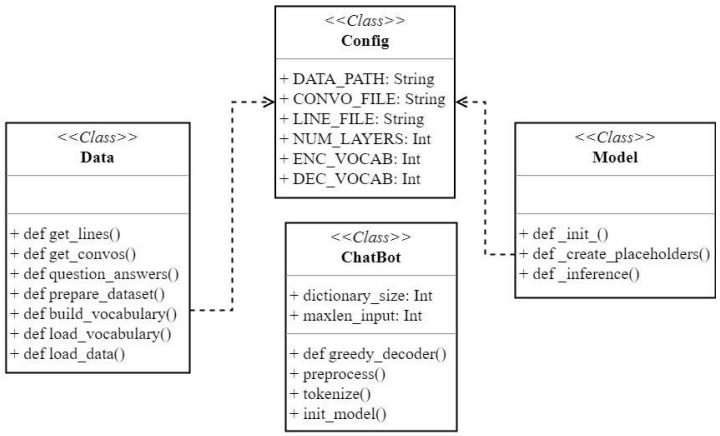

Рисунок 10. Діаграма класів прототипу інтелектуального чат-боту

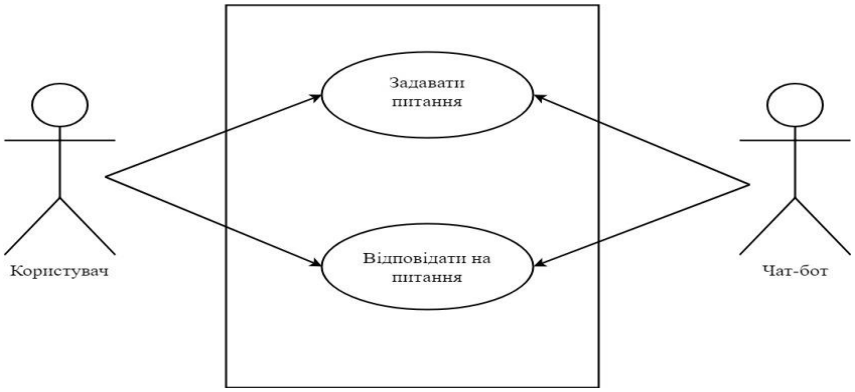

Рисунок 11. Діаграма варіантів прототипу інтелектуального чат-боту