

Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

Факультет Комп'ютерних наук
(повна назва)

Кафедра Штучного інтелекту
(повна назва)

КВАЛІФІКАЦІЙНА РОБОТА
Пояснювальна записка

рівень вищої освіти другий (магістерський)

Дослідження комбінування навчання з підкріпленням
та мовних моделей для реалізації діалогових агентів
(тема)

Виконав:
студент 2 курсу, групи СШМ-22-3
Бовдуй Р.В.
(прізвище, ініціали)

Спеціальність 122 Комп'ютерні науки
(код і повна назва спеціальності)

Тип програми освітньо-наукова
(освітньо-професійна або освітньо-наукова)

Освітня програма Системи штучного інтелекту
(повна назва спеціалізації)

Керівник проф. Кулішова Н. Є.
(посада, прізвище, ініціали)

Допускається до захисту

Зав. кафедри _____
(підпис)

В.О. Філатов
(прізвище, ініціали)

2024 р.

Харківський національний університет радіоелектроніки

Факультет _____ Комп'ютерних наук _____
(повна назва)
Кафедра _____ Штучного інтелекту _____
(повна назва)
Рівень вищої освіти _____ другий (магістерський) _____
Спеціальність _____ 122 Комп'ютерні науки _____
(код і повна назва)
Тип програми _____ освітньо-наукова _____
(освітньо-професійна або освітньо-наукова)
Освітня програма _____ Системи штучного інтелекту _____
(повна назва)

ЗАТВЕРДЖУЮ:
Зав. кафедри _____
(підпис)
« _____ » _____ 20 ____ р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

студентові _____ Бовдюю Роману Васильовичу _____
(прізвище, ім'я, по батькові)

1. Тема роботи _____ Дослідження комбінування навчання з підкріпленням та мовних моделей для реалізації діалогових агентів _____

затверджена наказом університету від 1 квітня 20 24 р. № 260Ст

2. Термін подання студентом роботи до екзаменаційної комісії 7 червня 20 24 р.

3. Вихідні дані до роботи Науково-технічні публікації, дані Інтернет-джерел та відомих наукових проектів, Python documentation, набір даних для тренування та тестування системи

4. Перелік питань, що потрібно опрацювати в роботі _____

1) Аналіз предметної галузі

2) Опис проведених теоретичних досліджень

3) Опис системи що пропонується

4) Опис проведених експериментальних досліджень

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів роботи	Терміни виконання етапів роботи	Примітка
1	Отримання завдання на кваліфікаційну роботу	01.04.2024	виконано
2	Аналіз літературних джерел, формулювання завдання дослідження	14.02.2024	виконано
3	Вивчення мовних моделей, розгляд особливостей трансформерів та моделі GPT-2	30.02.2024	виконано
4	Дослідження алгоритмів навчання з підкріпленням, аналіз властивостей алгоритмів Actor-Critic, BRAC	10.03.2024	виконано
5	Вивчення принципів дії та побудови task-орієнтованих діалогових систем	15.03.2024	виконано
6	Розробка архітектури діалогової системи на основі комбінування великої мовної моделі та алгоритму BRAC	13.04.2024	виконано
7	Експериментальне дослідження розробленої архітектури	29.04.2024	виконано
8	Аналіз отриманих результатів	05.05.2024	виконано
9	Оформлення пояснювальної записки	15.05.2024	виконано
10	Оформлення презентаційних матеріалів	20.05.2024	виконано
11	Подання на рецензування		
12	Подання кваліфікаційної роботи до захисту	07.06.2024	виконано

Дата видачі завдання 1 квітня 2024 р.

Студент _____
(підпис)

Керівник роботи _____
(підпис)

проф Кулішова Н. Є.
(посада, прізвище, ініціали)

РЕФЕРАТ

Пояснювальна записка: 65 с., 19 рис., 1 табл., 1 дод., 25 джерел.

ГЛИБОКА НЕЙРОННА МЕРЕЖА, ДІАЛОГОВА СИСТЕМА, МЕХАНІЗМ УВАГИ, МОВНА МОДЕЛЬ, НАВЧАННЯ З ПІДКРІПЛЕННЯМ, ТАСК-ОРІЄНТОВАНА, ТРАНСФОРМЕР.

Об'єкт дослідження – процес розробки архітектури таск-орієнтованої діалогової системи.

Предмет дослідження – моделі уваги, алгоритми навчання з підкріпленням.

Мета роботи – підвищити якість діалогового агенту завдяки поєднанню великих мовних моделей, таких як GPT-2, та підходів навчання з підкріпленням, створити архітектуру таск-орієнтованого діалогового агенту.

В даній роботі розглядається варіант поєднання такої мовної моделі, як GPT-2 на базі трансформеру, та алгоритмів навчання з підкріпленням, а саме – алгоритмів з необмеженою кількістю станів та безперервним простором дій, таких, як Actor-Critic та його модифікації.

В результаті досліджень вдалося побудувати архітектуру таск-орієнтованої діалогової системи, яка дозволила підвищити якість відповідей. Було доведено, що офлайнове навчання з підкріпленням може успішно використовуватися у складних людських інтерактивних завданнях, таких, як діалог. Результати роботи можуть слугувати підґрунтям для створення таск-орієнтованих діалогових систем, які можуть бути успішно примінені у різноманітних галузях і затребувані у бізнесу.

ABSTRACT

Master's thesis contains 65 pp., 19 fig., 1 tabl., 1 ann., 25 references.

ATTENTION MECHANISM, DEEP NEURAL NETWORK, DIALOGUE SYSTEM, LANGUAGE MODEL, REINFORCEMENT LEARNING, TASK-ORIENTED TRANSFORMER.

The object of research is the process of developing the architecture of a task-oriented dialogue system.

The subject of research is attention models, reinforcement learning algorithms.

The aim of the work is to improve the dialogue agent quality through a combination of large language models, such as GPT-2, and reinforcement learning approaches, to create the architecture of a task-oriented dialogue agent.

This work considers the option of combining such a language model as GTP-2 based on a transformer and reinforcement learning algorithms, namely algorithms with an unlimited number of states and a continuous space of actions, such as Actor-Critic and its modifications.

As a result of research, it was possible to build the architecture of a task-oriented dialogue system, which allowed to improve the quality of answers. It has been proven that offline reinforcement learning can be successfully used in complex human interactive tasks such as dialogue. The results of the work can serve as a basis for creating task-oriented dialogue systems that can be successfully applied in various industries and are in demand in business.

ЗМІСТ

Вступ.....	7
1 Аналіз предметної галузі.....	9
1.1 Таск орієнтовані діалогові системи.....	9
1.2 Огляд мовних моделей	15
1.2.1 Нейронні мережі прямого поширення та RNN.....	15
1.2.2 Механізм уваги.....	18
1.2.3 Популярні моделі на базі трансформерів	21
1.3 Навчання з підкріпленням.....	23
1.3.1 Загальний огляд навчання з підкріпленням	23
1.3.2 Класифікація алгоритмів навчання з підкріпленням	25
1.4 Підходи до створення таск-орієнтованих діалогових систем та постановка задачі дослідження.....	30
2 Мовна модель	33
2.1 Детальний огляд трансформерів.....	33
2.2 Мовна модель GPT-2	36
3 Алгоритм навчання з підкріпленням.....	39
3.1 Алгоритми градієнта стратегії	39
3.2 BRAC (behavior regularized actor critic).....	42
4 Побудова діалогового агента	45
4.1 Загальний огляд підходу	45
4.2 Налаштування навчання з підкріпленням	46
4.3 Побудова функції винагороди	49
4.4 Навчання діалогового агента	50
4.5 Оцінка результатів роботи моделі.....	52
Висновки	55
Перелік джерел посилання	57
Додаток А Результати роботи діалогового агента.....	61
Додаток Б Відомість кваліфікаційної роботи.....	65

ВСТУП

Мабуть, кожна людина у сучасному світі так чи інакше стикалася з різноманітними чат-ботами. Наприклад, коли клієнт звертається до банку і намагається вирішити свої питання; коли виникають проблеми, і клієнт телефонує у відповідну службу, або коли клієнт звертається в компанію, а на іншому боці слухавки чує відповідь голосового агента.

Останнім часом ринок чат-ботів продемонстрував значне зростання. Необхідність надання клієнтам цілодобового обслуговування, зниження операційних витрат стали можливими завдяки виконанню різноманітних завдань, а зростання вимог до операцій самообслуговування призвело до збільшення попиту.

Чат-боти набувають популярності, оскільки компанії продовжують автоматизувати свої продажі та обслуговування клієнтів. Використання чат-бота дозволяє організаціям надавати своєчасні послуги з меншими витратами. Чат-боти знаходять застосування у широкому діапазоні галузей, таких як охорона здоров'я, роздрібна торгівля, BFSI (Banking, financial services and insurance), медіа та розваги, подорожі та туризм, електронна комерція та інші.

Останні розробки у галузі глибокого навчання призвели до появи end-to-end підходів до побудови різноманітних моделей на базі навчання з вчителем або supervised learning, які генерують відповіді, виходячи з поточної розмови, поєднуючи висловлювання користувача та попередні відповіді. Багато з таких моделей використовують попередньо натреновані мовні моделі і потім донавчають їх на спеціалізованих датасетах під окремі задачі.

Наразі для того, щоб автоматизувати різноманітні інтерактивні задачі, такі, наприклад, як продаж товарів, допомога користувачеві у вирішенні побутових проблем, консультації з певних приводів та багато іншого, потрібно мати інтелектуальні діалогові агенти, які повинні генерувати

швидкі та натуральні відповіді, слідує меті, яка співпадає з намірами користувача. Будь-який діалог можна розглядати, як взаємодію двох людей, які намагаються вирішити деякі питання. Для цього люди ставлять певні цілі і починають рухатися до них протягом діалогу.

Незважаючи на всі переваги таких мовних моделей, як трансформери, які дозволяють надавати чіткі та розгорнуті відповіді на питання, пам'ятають контекст, можуть перекладати іншою мовою, в них немає націленості на результат для досягнення певної мети. З іншого боку, такі задачі вирішуються завдяки навчанню з підкріпленням, а саме – встановлення певної цілі і її досягнення.

Навчання з підкріпленням гарно себе зарекомендувало у таких галузях, як робототехніка, керування ресурсами в енергетиці або логістиці, в побудові різноманітних діалогових систем, автономному керуванні транспортними засобами та багатьох інших.

Виникає питання, чи можна поєднати існуючі мовні моделі такі, як трансформери, та навчання з підкріпленням, щоб побудувати модель, яка буде спілкуватися з людиною та вирішувати якусь ціль? Це може бути продаж товару за вигідною ціною чи консультація з будь-якого приводу у вирішенні різноманітних питань.

В цій роботі розглядаються саме такі питання і досліджується актуальність поєднання мовних моделей з навчанням з підкріпленням для побудови таск-орієнтованих діалогових агентів.

1 АНАЛІЗ ПРЕДМЕТНОЇ ГАЛУЗІ

1.1 Таск орієнтовані діалогові системи

Таск-орієнтовані діалогові системи (TOD) – системи, які проектуються таким чином, щоб виконувати конкретні завдання, відстежуючи стан діалогу та генеруючі адекватні відповіді користувачеві, щоб допомогти йому досягти заданої цілі [1].

В реальному світі, де спілкуються людина з людиною, розмова не завжди вважається успішною навіть коли деякі цілі були досягнуті. Якщо консультант у цій ситуації був доброзичливим, привітним і ефективним, то в користувача значно більше шансів отримати позитивний досвід і повернутися в майбутньому. Якщо консультант був грубим або некомпетентним, користувач залишиться розчарованим або навіть розгніваним і навряд чи повернеться знову.

У більшості випадків таск-орієнтовані системи мають сприяти позитивному досвіду користувача, намагаючись розв'язувати завдання найефективнішим способом. Відповідно, оцінка продуктивності таск-орієнтованої діалогової системи зазвичай враховує оптимізацію двох основних якостей: вирішення завдань і ефективність діалогу.

Системи, орієнтовані на завдання, продовжують набирати популярність, оскільки різні галузі знаходять способи ефективного використання їхніх можливостей, заощаджуючи час і гроші.

Виходячи з визначення таск-орієнтованих діалогових систем, стає зрозумілим, що галузь застосування таких діалогових агентів дуже обширна.

Вони можуть бути використані у навчанні, надаючи доступ до навчальних матеріалів, проведенні уроків та тестуванні знань. Таск-орієнтовані діалогові системи можуть надавати персоналізовані рекомендації та поради щодо вивчення певних предметів.

У сфері охорони здоров'я діалогові системи можуть допомагати у проведенні діагностики, надавати інформацію про лікування та рекомендації щодо здорового способу життя. Вони також можуть служити, як віртуальні помічники для лікарів, полегшуючи роботу з великим обсягом даних.

У бізнесі таск-орієнтовані діалогові системи можуть використовуватися для автоматизації клієнтського обслуговування, обробки замовлень, надання інформації про продукти та послуги компанії. Вони також можуть допомагати у проведенні маркетингових досліджень та управлінні бізнес-процесами.

В області розваг таск-орієнтовані діалогові системи можуть надавати рекомендації щодо вибору фільмів, музики або ігор, допомагати в організації подій та розваг, а також відігравати роль віртуальних персонажів в інтерактивних програмах.

Останнім часом навіть важче зазначити галузі та сфери де не можна використати такого рода системи. Популярність таск орієнтованих систем постійно зростає, і у цієї популярності є багато причин:

- зниження впливу людського фактору;
- економія на людській праці;
- підвищення швидкості виконання операцій;
- цілодобова доступність;
- збільшення кількості оброблених запитів клієнтів;
- персоналізація запитів та ін.

Глобальний ринок чат-ботів був оцінений у 5,132.8 мільйонів доларів США у 2022 році (рисунок 1.1) і очікується його зростання на 23% з 2023 до 2030 років (рисунок 1.2). Очікується, що використання такого роду систем призведе до зниження операційних витрат, а це, у свою чергу, призведе до поширення використання чат-ботів та зростання ринку.

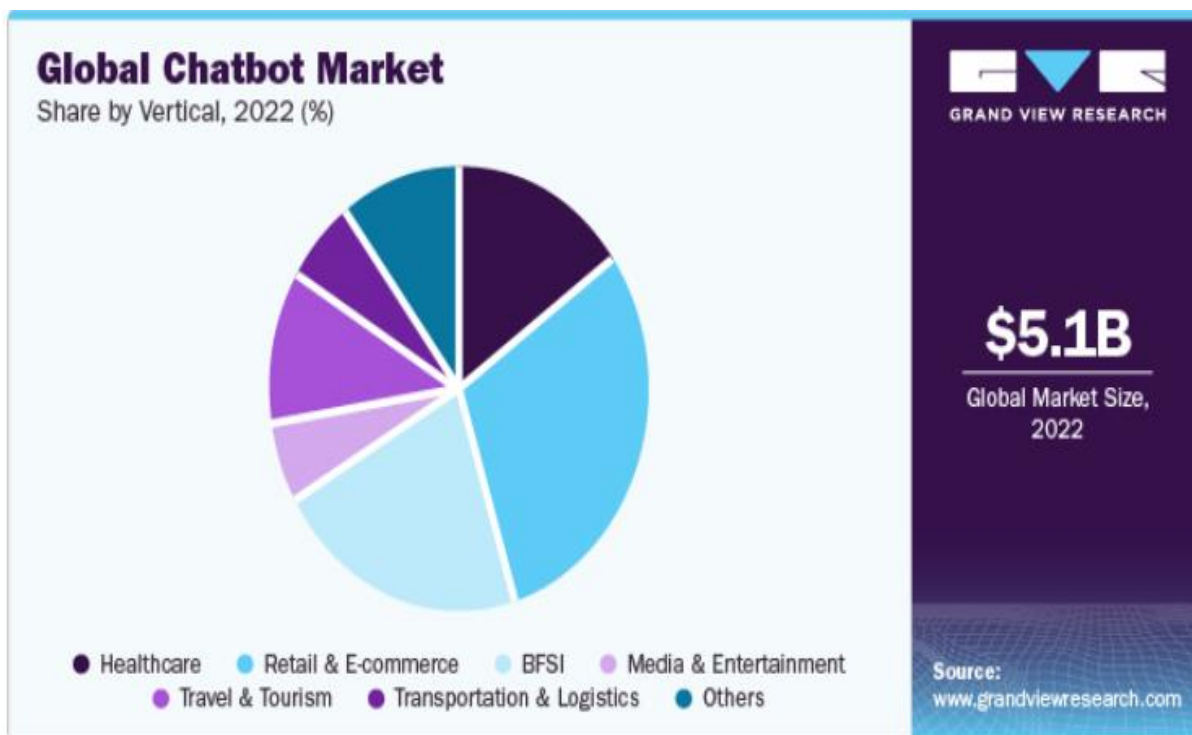


Рисунок 1.1 – Обсяг глобального ринку чат-ботів у 2022 році

Чат-боти – це програми, що реалізують підходи штучного інтелекту у вигляді спеціального набору правил. Такого рода системи призначені для взаємодії з людьми в процесі текстової розмови. Вони поєднуються з різними засобами обміну повідомленнями, таким чином підтримуючи користувачів у різних секторах. Очікується, що різноманітні інновації в області штучного інтелекту та технологій машинного навчання покращать можливості чат-ботів, що, у свою чергу, сприятиме попиту на ринку.

За останні кілька років ринок чат-ботів продемонстрував значне зростання. Необхідність надання клієнтам цілодобового обслуговування, зниження операційних витрат завдяки виконанню різноманітних завдань, зростання вимог до операцій самообслуговування призвели до збільшення попиту. Це можна пояснити тим, що користувачі надають все більше переваги програмам для обміну повідомленнями, ніж сайтам соціальних мереж.

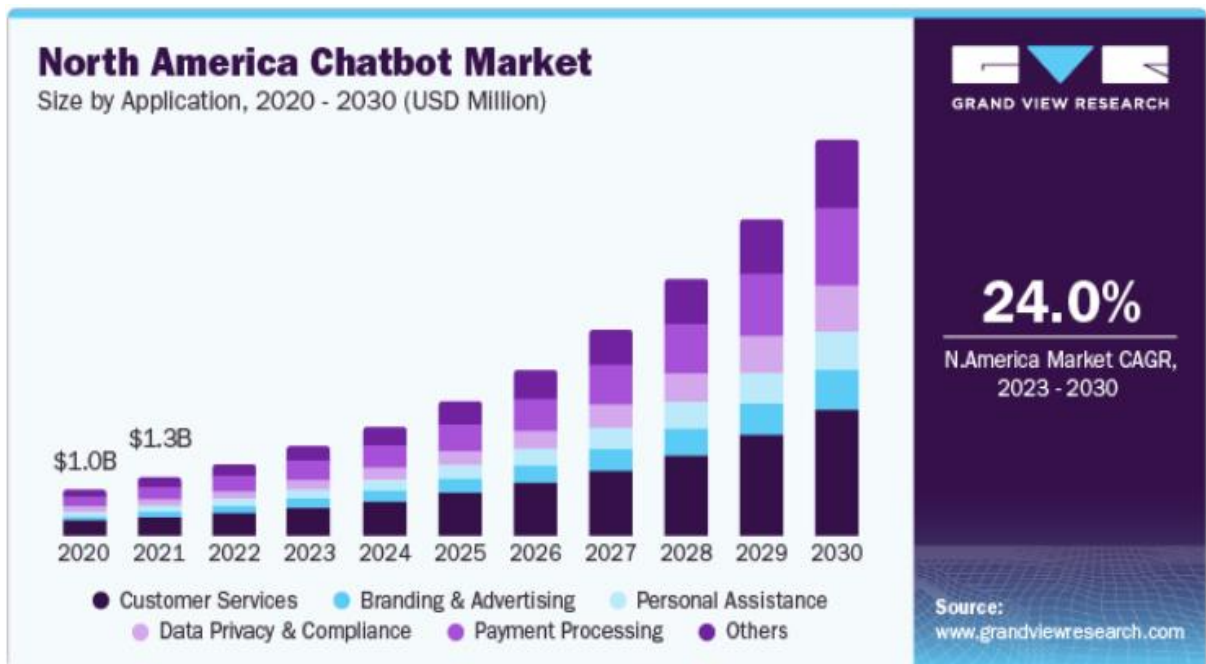


Рисунок 1.2 – Перспективи зростання глобального ринку чат-ботів до 2030 року

Чат-боти набувають популярності, оскільки компанії продовжують автоматизувати продажі та обслуговування клієнтів. Використання чат-бота дозволяє організаціям надавати своєчасні послуги з меншими витратами завдяки таким перевагам, як віддалене підключення та віртуальна присутність, що значно знижує транспортні витрати. Очікується, що галузь подорожей і туризму демонструватиме найвище зростання протягом прогнозованого періоду.

Північна Америка є головним осередком для стартапів у галузі чат-ботів, у 2022 році на нього припадала найбільша частка ринку – близько 30%. Значні перспективи зростання ринку чат-ботів можна пояснити суттєвими перевагами, які він пропонує, наприклад, скороченням операційних витрат підприємств, підвищенням задоволеності клієнтів і підвищенням операційної ефективності підприємств. За Північною Америкою слідує Азійсько-Тихоокеанський регіон, який є головним центром сфери послуг. Багато великих підприємств все частіше

використовують чат-бот у своїй рутинній діяльності з обслуговування клієнтів [2].

Побудова будь-якої сучасної діалогової системи являє собою доволі складну задачу, і для того, щоб створити дійсно корисний продукт, потрібно гарно розуміти зміст цього завдання.

Існує дуже багато задач, датасетів та моделей, на базі яких можна створити ту чи іншу діалогову систему. Якщо побудувати будь-яку модель та обрати датасет навчання, результати побудови агенту можуть бути не дуже гарними. Тому насамперед потрібно відповісти на декілька питань і зрозуміти основну мету діалогової системи.

Є багато публікацій, які допомагають зорієнтуватися у типах діалогових агентів та визначити цілі. Наприклад, у роботі [3] розглядаються питання, які потрібно вирішити розробнику перед початком створення діалогового агента: які вхідні дані очікуються від агента, до якої доменної галузі належать запити та який контекст потрібен для виконання тих чи інших потреб користувача. Потрібно також визначити, яким буде формат вхідних та вихідних даних. Наприклад, це може бути текстова переписка, голосові запити або малюнки.

Не менш важливим є розуміння того, що мають робити діалогові агенти. У тій самій статті [3] автори змогли ідентифікувати одинадцять різноманітних завдань, які вирішують діалогові агенти, а саме:

- Dialogue Rewrite (DR) – переписування різноманітних висловів, щоб вони краще підходили до контексту зі збереженням оригінального значення;
- Dialogue Summary (DS) – коротко описувати теми, ідеї та аргументи, які були обговорені протягом діалогу;
- Dialogue to Structure (D2S) – перетворення тексту на SQL або семантичне дерево;
- Question Answering(QA) – вміння ставити релевантні питання та давати адекватні відповіді на запити користувачів;

- Knowledge Grounded Response (KGR) – це використання зовнішніх знань для створення адекватних відповідей;
- Chit-Chat (CC) – вміння підтримувати розмову на будь-яку тему, без цілі або завдання;
- Task-Oriented Dialogues (TOD) – це генерування таск-орієнтованих діалогів для досягнення певних цілей;
- Intent detection (ID) – це змога ідентифікувати наміри і цілі юзера, аналізуючи контекст або розмову;
- Slot Filling (SF) – спроможність запросити необхідну інформацію у користувача, так би мовити «заповнити слоти», і часто SF йде поряд з ID;
- Dialogue State Tracking (DST) – це відстеження стану діалогу під час розмови і відслідкування цілей;
- Affect Detection (AD) – для відстеження наміру користувача потрібно відстежувати афективні атрибути, наприклад, емоції, і включати їх у відповідь агента.

Не менш важливо для кожного типу завдання мати спеціальний датасет, який краще допоможе у досягненні цілі. Наприклад, якщо обрано датасет для розмови на довільну тему, то він не буде корисним у побудові агента, що вирішує таск-орієнтовані завдання, як продаж товару за прийнятною ціною, де можна використовувати методи навчання з підкріпленням. Формат даних датасету також важливий: коли дані помічені певним чином, то їх легше використати для конкретного завдання.

Для побудови таск-орієнтованих діалогів можна використовувати різноманітні мовні моделі, алгоритми на базі навчання з підкріпленням або їхні комбінації. Щоб визначити сильні та слабкі сторони тих чи інших підходів, потрібно зробити короткий огляд цих моделей та алгоритмів, щоб зрозуміти їх можливості та останні здобутки.

1.2 Огляд мовних моделей

1.2.1 Нейронні мережі прямого поширення та RNN

Глибокі нейронні мережі прямого поширення (Feedforward Networks) досягли великих успіхів у вирішенні багатьох складних завдань в області класифікації, кластеризації, розпізнавання образів та інших.

Нейронні мережі прямого поширення також активно використовуються для обробки текстів. За їх допомогою можна класифікувати різноманітні тексти, здобувати з них інформацію, наприклад, дати або об'єкти тощо, але вони стикаються з обмеженнями при обробці довгих послідовностей. Але подібні нейронні мережі не враховують порядок вхідних даних, не здатні працювати з послідовностями різної довжини, не пам'ятають попередніх станів та не можуть обробляти довгострокові залежності.

З появою рекурентних нейронних мереж Recurrent Neural Networks (RNNs) вдалося опанувати послідовності та успішно використати такі мережі для моделювання мови, генерації текстів, машинного перекладу, передбачення часових рядів, обробки аудіо та відео потоків, генерування опису малюнків та інших завдань.

На відміну від нейронних мереж прямого поширення, які передавали інформацію через мережу без циклів, рекурентні нейронні мережі мають цикли і здатні передавати інформацію в себе знову разом з новою інформацією. Нейрони у таких мережах замінені так званими RNN нейронами, які мають прихований стан і кожен раз при надходженні нової інформації враховують цей стан (рисунок 1.3). Саме ця особливість дозволяє розширити функціональність мереж прямого поширення і брати до уваги попередню інформацію [4].

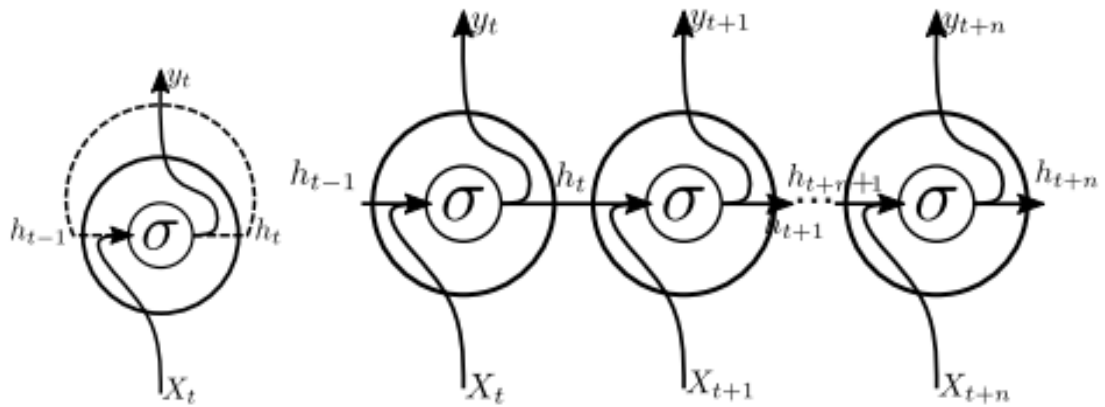


Рисунок 1.3 – Розгорнуте зображення RNN [5]

Незважаючи на те, що рекурентні нейронні мережі показали непогані результати, вони також мають свої недоліки. Як було вже згадано вище, RNN нейрони мають прихований стан (рисунок 1.4). Цей прихований стан постійно оновлюється при надходженні нової інформації і з часом може ігноруватися для дуже довгих послідовностей. Це відбувається тому, що рекурентні нейронні мережі потерпають від проблеми згасання градієнта тому, що градієнти калькулюються, враховуючи не тільки глибинний розмір мережі, а і часовий розмір [6].

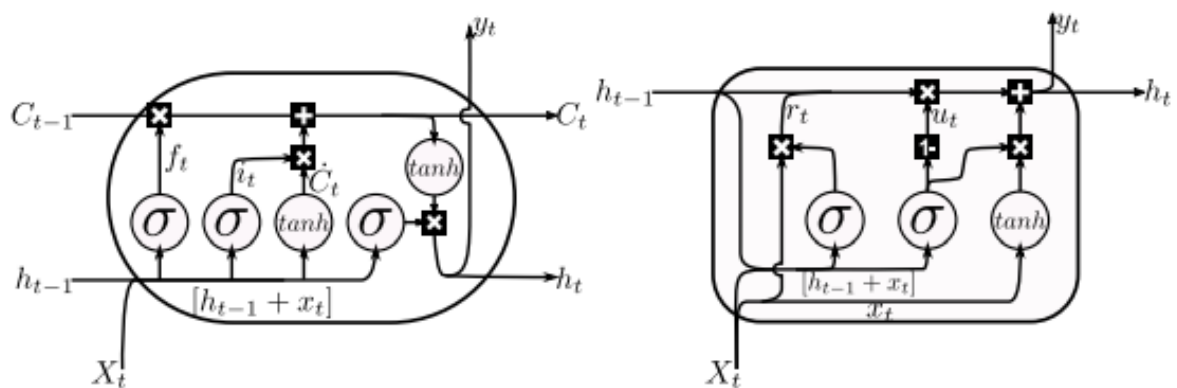


Рисунок 1.4 – Загальна структура LSTM (ліворуч) та GRU (праворуч) комірок [5]

Ці дві комірки мають доволі схожу структуру. Стандартна LSTM комірka має три гейта (gates): forget gate (f_t), який визначає, як багато попередніх даних треба забути, input gate (i_t), який визначає, скільки інформації потрібно зберігати у пам'ять комірki, та output gate (o_t), який визначає, як калькулювати вихід з наявної інформації.

GRU комірka дещо простіша, має менше параметрів і тільки два гейта: update gate (u_t) вирішує, як швидко повинне поновлюватися приховане становище, and reset gate (r_t), який вирішує, скільки минулої інформації забути, скидаючи частину пам'яті [5].

Найпростіша архітектура RNN мережі з глибоким навчанням – це Deep Recurrent Neural Network (DRNN), яка нагадує звичайну глибоку нейронну мережу, але замість звичайних нейронів використовує RNN комірki. Такі нейронні мережі дозволяють прогнозувати наступні елементи послідовності.

Наступна еволюція такої мережі – це Bidirectional Recurrent Neural Network (BRNN) (рисунок 1.5), яка дозволяє зазирнути наперед, наприклад, якщо потрібно заповнити якийсь пропуск у реченні [9].

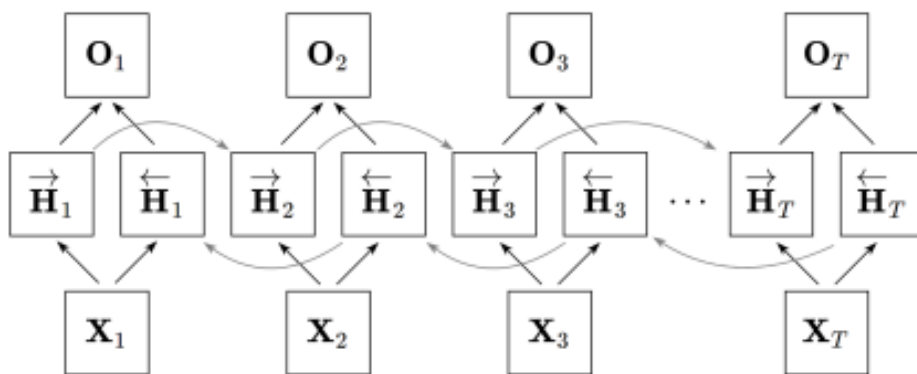


Рисунок 1.5 – Архітектура двонаправленої нейронної мережі [9]

Для того, щоб надати вищезгадану властивість мережі, додається ще один прихований шар, який пропускає послідовність у зворотному напрямку від останнього елементу до першого.

Одна з найуспішніших архітектур, яка з'явилася у 2014 році – це енкодер-декодер архітектура, також відома як Sequence-to-Sequence (seq2seq) [10]. Вона мала успішне застосування у перекладі текстів, побудові різних чат-ботів, прогнозуванні та іншому.

Sequence-to-Sequence архітектура складається з двох мереж (рисунок 1.6): енкодеру, на який подається вхідна інформація і перетворюється у внутрішнє контекстне представлення, та декодеру, який отримує контекстне представлення від енкодера, та використовується для генерації вихідної послідовності. Ця модель використовує RNN для побудови як енкодера так і декодера.

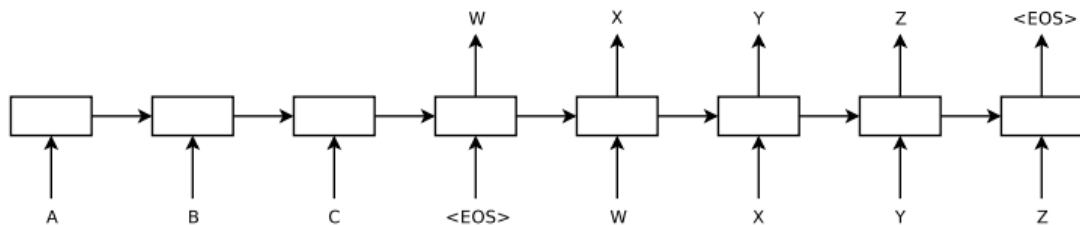


Рисунок 1.6 – Візуалізація Sequence-to-Sequence моделі [10]

Encoder Vector (або Context Vector) представляє собою останній прихований шар енкодера і агрегує всю інформацію, яка була подана на вхід енкодеру.

Одним з найбільших недоліків цієї мережі є саме прихований шар, який являє собою вузьке місце цієї архітектури, тому що повинен вміщувати всю інформацію, яка була передана до енкодера, але цей вектор має фіксовану довжину.

1.2.2 Механізм уваги

Sequence-to-Sequence моделі мають багато недоліків. Окрім фіксованого прихованого вектору, який не може вмістити весь контекст,

вони не зважають на послідовність слів у реченні, не враховують контекст на етапі навчання, зазнають впливу проблеми затухаючого або вибухаючого градієнту.

Для вирішення цього питання було запропоновано так званий концепт «уваги» [11] (рисунок 1.7). Ідея уваги була запозичена у людини. Однією з важливих властивостей людського сприйняття є те, що людина не прагне опрацювати всю сцену відразу, замість цього люди вибірково фокусують увагу на частинах візуального простору, щоб отримати інформацію, коли і де вона потрібна, і комбінують інформацію з різних фіксацій з часом, щоб створити внутрішнє уявлення про сцену [11].

Стосовно мовних моделей, механізм уваги було вдало вжито для розширення encoder-decoder моделі [12]. Було запропоновано кожного разу, коли генерується слово у перекладі, обирати, на якій частині вихідної послідовності потрібно зосередитися, тобто модель передбачає наступне слово, базуючись на векторах контексту асоційованих з позиціями слів у вихідному реченні та беруться до уваги попередньо згенеровані слова.

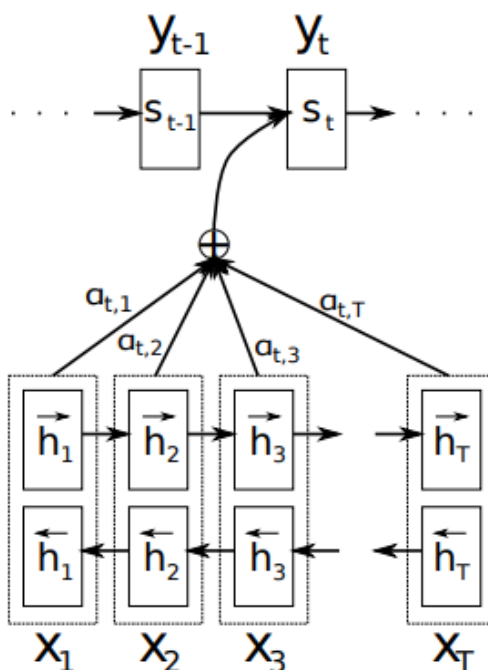


Рисунок 1.7 – Модель загальної уваги [12]

Модель більше не намагається закодувати всю вхідну послідовність у вектор фіксованої довжини, натомість вона кодує вхідне речення у послідовність векторів і потім адаптивно обирає підмножину векторів під час декодування.

Результати виявилися приголомшливими, моделі з використанням механізмом уваги виявилися більш стійкими до довжини речення (рисунок 1.8).

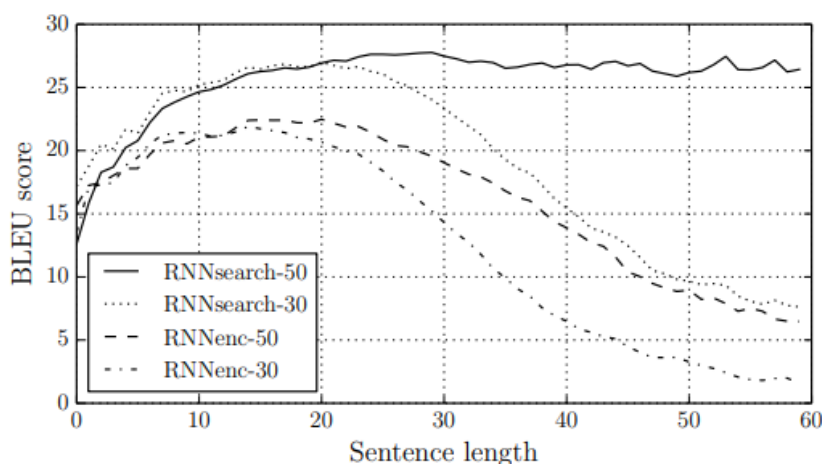


Рисунок 1.8 – Порівняння моделей з увагою RNNsearch та без уваги RNNenc [12]

Механізм уваги можна представити як матрицю кореляції, де стовпцями виступають слова з однієї мови а строками слова з іншої мови для одного й того ж самого речення. На перетині відкладаються коефіцієнти кореляції. Білі квадратики свідчать про високу ступінь кореляції (рисунок 1.9).

Наступний потужний поштовх мовні моделі отримали з появою так званих трансформерів (Transformer) [12]. Моделі, розглянуті вище, працюють послідовно, і чим довша ця послідовність, тим довше проходять розрахунки і вимагають більше обчислювальних ресурсів. Такі моделі не дозволяють обробляти дані паралельно.

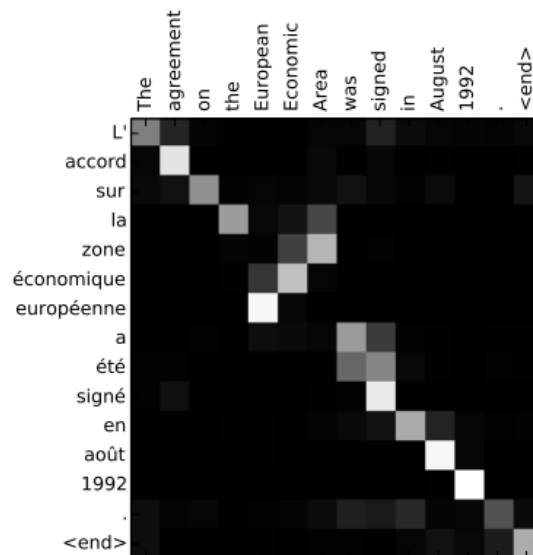


Рисунок 1.9 – Приклад матриці кореляції для моделі RNNsearch-50 [12]

Трансформери уникають використання повторень і замість використання рекурентних юнітів цілком покладаються на механізм самоуваги для покращення продуктивності. Для цього трансформери використовують позиційне кодування слів.

1.2.3 Популярні моделі на базі трансформерів

Поява механізмів уваги та трансформерів призвела до прориву у вирішенні багатьох задач, з'явилися такі популярні архітектури як GPT, BERT та T5.

BERT або Bidirectional Encoder from Transformers представляє собою поєднання дванадцяти шарів енкодерів з оригінальною архітектурою трансформерів [13] і використовує двоспрямований механізм самоуваги [14]. Архітектура BERT створена для вирішення завдань, пов'язаних з розумінням контексту в реченнях та текстах. Ця модель гарно себе зарекомендувала у задачах машинного перекладу, класифікації текстів, системах відповідей на питання, та аналізу настроїв.

GPT або Generative Pre-trained Model була створена OpenAI [15]. Вона представляє собою дванадцять шарів декодерів та використовує дванадцять маскованих причинних голів самоуваги. Модель створена для генерації текстів та розуміння контексту на рівні речення. Вона спроможна виконувати багато різноманітних завдань, таких як генерація текстів, переклад, генерація відповіді на питання на основі своїх знань та інших.

T5 або Text-to-Text Transfer Transformer [16] – ще одна популярна модель для обробки природної мови. Архітектура базується на архітектурі трансформера, та включає в себе як енкодер, так і декодер. Енкодер складається зі стеку блоків, які представляють собою два компоненти – шар самоуваги і маленьку мережу прямого поширення. Декодер блок дуже схожий на енкодер, але замість механізмів самоуваги включає стандартні механізми уваги, на вхід яких подається вихід шару самоуваги енкодера. Особливість цієї моделі в тому, що вона розглядає будь-яку проблему обробки тексту як «text-to-text» проблему, тобто бере на вхід текст і видає на вихід текст, замість того, щоб вирішувати конкретну задачу з фіксованими входом та виходом. Це дозволяє одній і тій самій моделі вирішувати широке коло завдань без будь-якої специфічної модифікації моделі під конкретне завдання (рисунок 1.12).

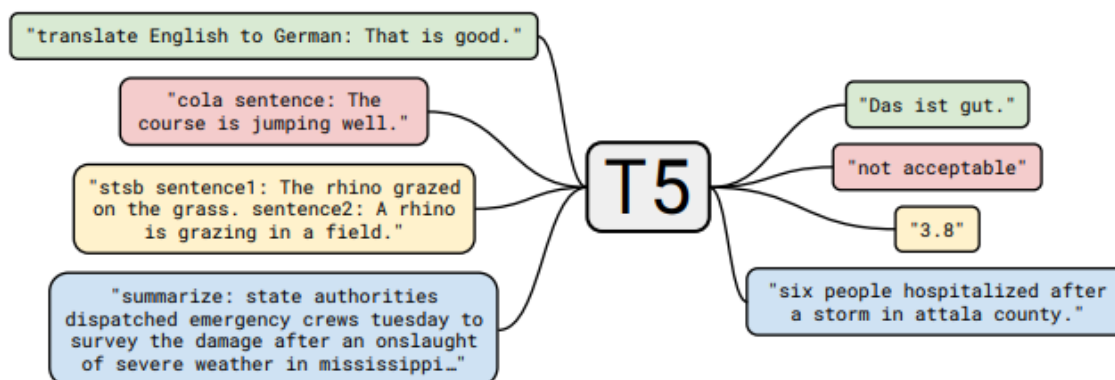


Рисунок 1.12 – Діаграма text-to-text фреймворку. Для кожної задачі використовується префікс [17]

1.3 Навчання з підкріпленням

1.3.1 Загальний огляд навчання з підкріпленням

Навчання з підкріпленням, або Reinforcement Learning, це одна з трьох парадигм машинного навчання, окрім навчання з наглядом та без нагляду. В навчанні з підкріпленням присутні так звані агент та навколишнє середовище, де агент постійно взаємодіє з середовищем, досліджує його та вивчає оптимальну стратегію поведінки методом спроб та помилок.

Навчання з підкріпленням може бути представлене Марковським процесом прийняття рішень (Markov Decision Process або MDP). Кожне середовище представлене деяким станом, яке відображає те, що коїться в цьому середовищі. Агент робить певну дію в середовищі і воно змінюється, набуває нового стану, і агент отримує винагороду як результат цієї дії (рисунок 1.13).

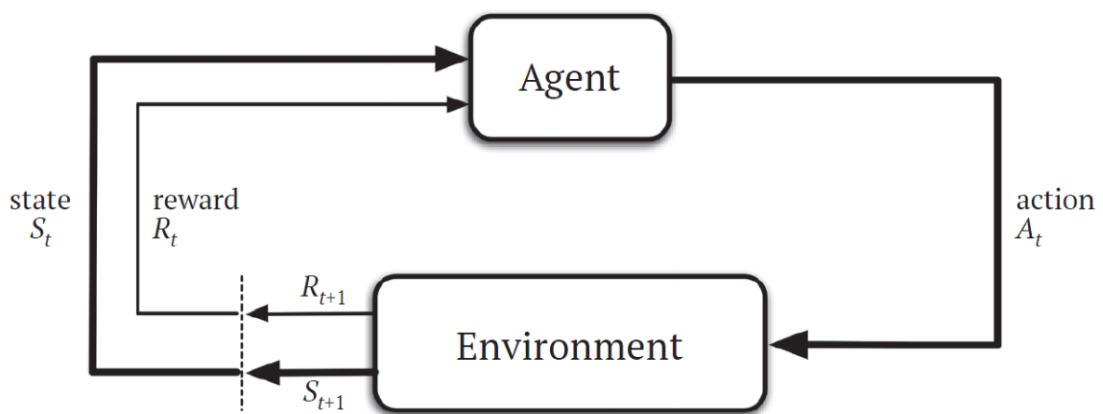


Рисунок 1.13 – Взаємодія між агентом та навколишнім середовищем в марковському процесі прийняття рішень

Агент може отримувати позитивну або негативну винагороду, дивлячись, наскільки корисною була та чи інша дія, і таким чином навчається.

Ціль агента – максимізувати винагороду, отриману на кожному кроці, що спонукає агента шукати кращі дії на протязі епізоду. Епізод, або траєкторія, це кінцева кількість дій, яка закінчується, коли агент досягає мети. Іноді дії можуть бути безперервними і не можуть бути розбиті на епізоди.

Дохід, або сумарну винагороду, можна визначити так:

$$G_t = r_0 + \lambda^1 r_1 + \lambda^2 r_2 + \dots + \lambda^n r_n = \sum_{t=0}^{\infty} \lambda^t r_t, \quad (1.1)$$

де λ – фактор знецінення, який визначає важливість майбутніх винагород і знаходиться в інтервалі $[0, 1]$.

Дохід дозволяє оцінити будь-яку траєкторію, але агент не має ніякої інформації про окремі стани. Коли є інформація про цінність станів, тоді можна обрати дію, яка приведе агента у стан з більшою винагородою, таким чином він може покращити стратегію. Тож, вводиться поняття функції цінності стану:

$$v_{\pi}(s) = E_{\pi}[G \vee s_0 = s] = E_{\pi}[\sum_{t=0}^{\infty} \lambda^t r_t \vee s_0 = s], \quad (1.2)$$

де $E_{\pi}[\cdot]$ – математичне очікування випадкової величини при старті з даного стану та дотриманні стратегії π .

За аналогією з функцією цінності станів визначається функція цінності дії. Вона визначає очікуваний дохід при старті з даного стану за умови даної першої дії і описується у такий спосіб:

$$q_{\pi}(s) = E_{\pi}[G \vee s_0 = s, a_0 = a] = E_{\pi}[\sum_{t=0}^{\infty} \lambda^t r_t \vee s_0 = s, a_0 = a]. \quad (1.3)$$

Вирішення задачі навчання з підкріпленням означає знаходження стратегії, яка допомагає агентові отримати найбільшу винагороду.

Стратегія π краща, ніж стратегія π' , якщо очікуваний дохід для першої стратегії вищий, ніж для другої, тобто $\pi \geq \pi'$, якщо $v_\pi(s) \geq v_{\pi'}(s)$ для всіх станів $s \in S$. Існує принаймні одна стратегія, яка краща або дорівнює іншим стратегіям, вона називається оптимальною і позначається π^* .

Таким чином, можна визначити оптимальну функцію цінності і оптимальну функцію цінності дії у такий спосіб:

$$v_*(s) = \max_{\pi} v_{\pi}(s) \text{ та } q_*(s) = \max_{\pi} q_{\pi}(s), \quad (1.4)$$

де для всіх $s \in S$ [17].

1.3.2 Класифікація алгоритмів навчання з підкріпленням

Існує багато різноманітних алгоритмів навчання з підкріпленням. Всі вони використовуються для різних цілей і поділяються на різні категорії. Для того, щоб обрати той чи інший алгоритм для конкретної задачі, потрібно врахувати різноманітні фактори.

Гарна класифікація розглянута у роботі [18], де алгоритми класифікуються за кількістю станів та типом дій, доступних у середовищі, і поділяються на три основні категорії (рисунк 1.14).

Перша категорія – алгоритми з обмеженою кількістю станів та дискретними діями.

Це доволі прості середовища, де є обмежена кількість дій. Найбільш відомими алгоритмами, які належать до даної групи, є Q-learning та State-Action-Reward-State-Action (SARSA).

Q-learning алгоритм знаходить оптимальну стратегію за допомогою Q-table, де зберігаються усі можливі стани та дії. Q-value оновлюється для кожної пари state-action, ітеративно використовуючи рівняння Беллмана доки Q-функція не зійдеться до оптимального значення Q-value.

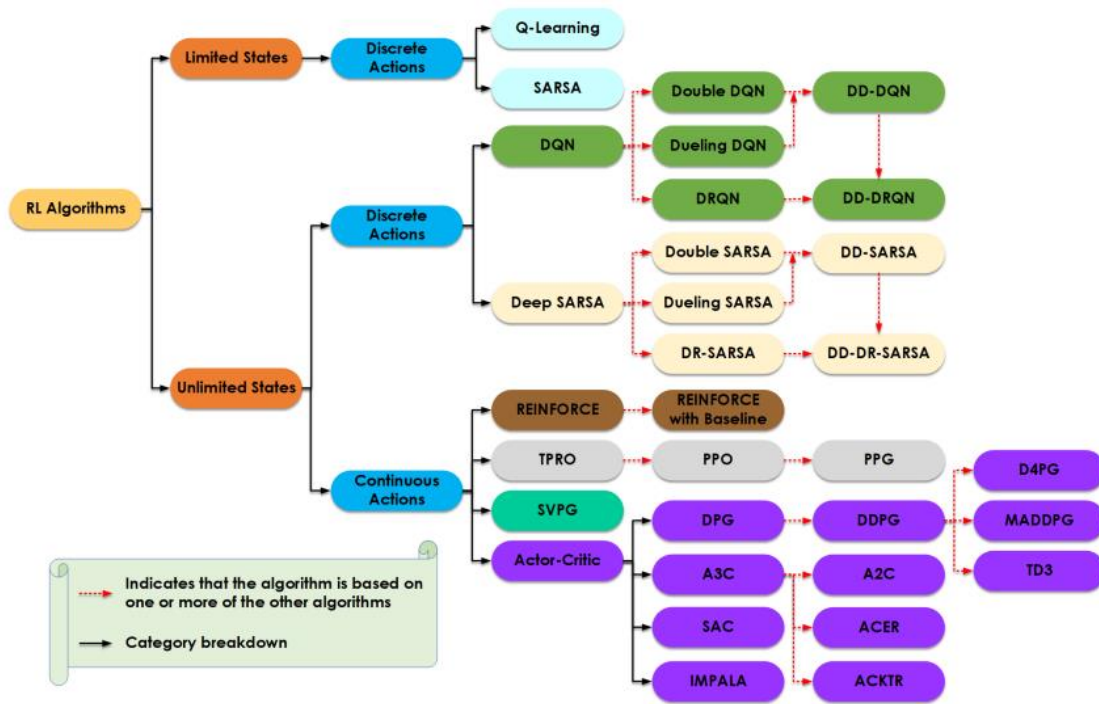


Рисунок 1.14 – Класифікація алгоритмів на основі типу середовища [18]

SARSA – алгоритм схожий на попередній, але Q-value оновлюється у інший спосіб. Щоб оновити $Q(s,a)$ значення, вираховується наступне значення state-action $Q(s',a')$, а так як наступна дія невідома, Q-learning обирає наступну дію жадібно, щоб максимізувати винагороду. Потім оновлення відбувається за формулою:

$$Q(s,a) = Q(s,a) + \alpha[r_t + \gamma Q(s',a') - Q(s,a)], \quad (1.5)$$

де α – коефіцієнт, що визначає, наскільки сильно змінилася цінність дії,

γ – коефіцієнт знецінення від 0 до 1, що зменшує цінність майбутніх дій.

Алгоритми цієї групи можуть успішно використовуватися в робототехніці, настільних іграх таких, як шахи, в управлінні різноманітними завданнями, тобто всюди, де є обмежена кількість станів та дій.

Категорія друга – алгоритми з необмеженою кількістю станів та дискретними діями.

В алгоритмах цієї групи використовуються глибокі нейронні мережі та згорткові нейронні мережі, які відповідають за обробку та здобуття різноманітних ознак зі станів, і на виході видають доступні дії. До цієї групи відносяться такі алгоритми, як Deep Q-Networks (DQN), Deep SARSA та різноманітні їхні варіації.

DQN – це один з перших алгоритмів, який вдало поєднує навчання з підкріпленням та глибокі нейронні мережі. Модель поєднує згорткову нейронну мережу і за нею слідуючу глибоку нейронну мережу. Входом цієї моделі є набір пікселів, наприклад скрін гри, який представляє собою стан середовища, а виходом цієї моделі є Q-value, що представляє кращу дію, яку потрібно виконати. Потім модель виконує цю дію, що призводить до зміни середовища і генерації винагороди.

Deep SARSA, на відміну від SARSA, придатний для роботи з необмеженою кількістю станів і для цього використовує глибокі нейронні мережі. Алгоритм схожий на DQN, але для розрахунку $Q(s,a)$ використовує $Q(s',a')$, виконуючи наступну дію.

Гарним прикладом застосування алгоритмів цієї групи є можливість навчитися іграм Atari. В цих іграх необмежена кількість станів та обмежена кількість дій.

Категорія третя – алгоритми з необмеженою кількістю станів та безперервним простором дій.

Коли кількість станів дуже велика, а простір дій безперервний, алгоритми, що навчають функцію цінності або функцію цінності дії, стикаються з великими труднощами. В таких випадках приходять на допомогу алгоритми градієнта стратегії.

Існує велика кількість алгоритмів з градієнтом стратегії. Найбільш фундаментальні з них – це REINFORCE, Trust Region Policy Optimization (TRPO), Stein Variational Policy Gradient (SVPG) та Actor-Critic [18].

REINFORCE алгоритм відноситься до алгоритмів Монте-Карло, тому що дохід оцінюється за вибірковими траєкторіями при дотриманні заданої стратегії. В функції градієнту цільової функції цінність дії замінюється доходом Монте-Карло G :

$$\nabla_{\theta} J(\theta) = E_{\pi_{\theta}}[\nabla_{\theta} \log \pi_{\theta}(a \vee s) Q_{\pi_{\theta}}(s, a)] = E_{\pi_{\theta}}[\nabla_{\theta} \log \pi_{\theta}(a_t | s_t) G_t], \quad (1.6)$$

де G_t – дохід, який обчислюється до повної траєкторії,

T – повна кількість кроків,

t – крок.

REINFORCE алгоритм відзначається відсутністю зсуву, але має велику дисперсію. Одним з недоліків простого REINFORCE алгоритму є повільна збіжність [19].

SVPG – це метод, який поєднує статистичне моделювання і градієнтний спуск для оновлення параметрів стратегії. Ідея цього алгоритму полягає в тому, щоб сформувавши розподілення стратегії агента таким чином, щоб вона була максимально наближена до оптимального розподілу стратегії. SVPG оновлює набір «часток» $\{\theta_i\}$ через формулу:

$$\theta_i = \theta_i + \alpha \phi^*(\theta_i), \quad (1.7)$$

де $\phi^*(\cdot)$ – одиничний шар RKHS (відтворювальне ядро гільбертового простору), вектор, який максимально зменшує Kullback-Leibler (KL) відстань між функціями розподілу:

$$\phi^*(\theta_i) = \max\{-\nabla_{\epsilon} D_{KL}(q'_{[\theta + \epsilon \phi(\theta)]} \vee q)\}, \quad (1.8)$$

де $\|\phi\|_H \leq 1$, $q'(\cdot)$ – розподіл $\theta + \alpha \phi(\theta)$,

q – цільовий розподіл [20].

TRPO – алгоритм, який представляє поліпшення алгоритму Natural Policy Gradient – NPG. Поліпшення полягає в обмеженні на розбіжність KL між новою і старою стратегіями, в результаті якого утворюється довірча область. Це дозволяє мережі виконувати більші кроки, залишаючись в межах довірчої області. Завдання з обмеженнями формулюється так:

$$\text{maximize}_{\theta} J_{\theta_{old}}(\theta), \quad (1.9)$$

за умови $D_{KL}(\theta_{old}, \theta) \leq \sigma$.

Тут $J_{\theta_{old}}$ – сурогатна цільова функція, $D_{KL}(\theta_{old}, \theta)$ – розбіжність КЛ між старою стратегією з параметрами θ_{old} і новою стратегією з параметрами θ , σ – верхня межа розбіжності.

Сурогатна цільова функція спроектована так, що максимізує параметри нової стратегії при використанні розподілу станів старої стратегії. Це робиться за допомогою вибірки за значимістю, що дозволяє оцінити розподіл нової стратегії (бажаний), знаючи тільки розподіл старої стратегії (відомий) [19].

Actor-Critic – це група алгоритмів, заснованих на градієнті стратегії та таких, що складаються з двох базових компонентів:

- актора, який відповідає за вибір дії на основі стану навколишнього середовища, тобто налаштовує параметр θ стратегії π_{θ} для того, щоб максимізувати винагороду;

- критика, який оцінює цінність обраної дії. Цінність дії представляється у вигляді функції $Q(s, a)$, яка представляє собою очікуваний дохід при старті з даного стану s за умови даної першої дії a .

Тобто, актора можна описати як мережу, що намагається знайти ймовірність усіх доступних дій і вибрати дію з найвищим значенням, тоді як критика можна описати як мережу, що оцінює вибрану дію шляхом оцінки значення нового стану, отриманого в результаті виконання дії.

Алгоритми цієї групи використовуються у різноманітних життєвих сценаріях, де немає заздалегідь заданих дій, а рішення задаються безперервними діями, тобто описуються певними кількісними характеристиками у різних напрямках.

1.4 Підходи до створення task-орієнтованих діалогових систем та постановка задачі дослідження

Архітектури task-орієнтованих систем можуть бути класифіковані на два типи. Перший з них – це end-to-end підхід, коли запити юзера відображаються на відповіді моделі. Зазвичай у такому підході використовуються sequence-to-sequence моделі, які навчаються за допомогою контрольованого навчання. Цей підхід більш гнучкий та має менше вимог до анотації вхідних даних, але загалом потребує більшого обсягу даних для навчання. Непрозора структура не дає змогу інтерпретувати результати, та не дозволяє жодного контролю.

Інший тип систем реалізує послідовний підхід або пайплайн, де система декомпозується у кілька послідовних під-завдань: спочатку працює NLU модуль (natural language understanding або модуль розуміння природної мови), який відповідає за розпізнавання намірів користувача та слотів із вхідної послідовності. Потім працює DST модуль (dialogue state tracking або відстеження стану діалогу), який запам'ятовує внутрішній стан діалогу. Наступний модуль DPL (dialogue policy learning або вивчення стратегії діалогу) – компонент, що виконує дії, які задовольняють вимоги користувача, і наприкінці NLG (natural language generation або генерування природної мови) відповідає за трансформацію дії у відповідь користувачу. Цей підхід вважається легшим в інтерпретації та легшим в імплементації, але в цілому систему складніше оптимізувати глобально [21].

Останнім часом зростає інтерес саме до побудування end-to-end підходу, який генерує відповіді, виходячи з поточної розмови, поєднуючи

висловлювання юзера та попередні відповіді. Багато з таких моделей використовує попередньо натреновані мовні моделі і потім донавчають їх на спеціалізованих датасетах під окремі задачі.

Якщо розглядати саме end-to-end підхід, то з огляду, проведеного вище, зрозуміло, що великі мовні моделі можуть отримати надзвичайний обсяг знань із великих текстових корпусів і можуть бути застосовані до широкого кола мовних завдань, але такі моделі не пристосовані для того щоб вирішувати конкретні завдання юзера. Модель потребує багато різних проб та помилок, щоб отримати підказки, які спрямують модель до бажаної цілі.

Навчання з підкріпленням, навпаки, поєднує алгоритми, які створені, щоб вирішувати task-орієнтовні проблеми. Через те, що діалог дуже схожий на послідовне прийняття рішень, навчання з підкріпленням може бути природно використано для навчання task-орієнтованих діалогових систем для досягнення певної цілі. Але через те, що алгоритми навчаються на абстрактних станах та діях, ці методи страждають від того, що не можуть генерувати відповіді в натуральному для людини вигляді. Ще одним великим недоліком є те, що для роботи алгоритмів навчання з підкріпленням потрібні великі анотовані датасети, але їхня кількість дуже мала порівняно з великим обсягом не анотованих даних.

Судячи з усього вищезгаданого, постає завдання поєднати мовні моделі та алгоритм навчання з підкріпленням, для того, щоб отримати переваги одного та іншого підходу, та нівелювати їхні недоліки.

Навчання з підкріпленням може використовуватися як фреймворк для спрямування мовної моделі для вирішення завдань, заданих користувачем.

Об'єкт дослідження – процес розробки архітектури task-орієнтованої діалогової системи.

Предмет дослідження – моделі уваги, алгоритми навчання з підкріпленням.

Мета роботи – підвищити якість діалогового агента завдяки поєднанню великих мовних моделей, таких як GPT-2, та підходів навчання з підкріпленням, створити архітектуру таск-орієнтованого діалогового агента.

Для того, щоб досягти мети дослідження, потрібно розв'язати кілька часткових завдань:

- детально розглянути принципи дії та архітектуру Трансформера та великої мовної моделі GPT;
- вивчити особливості реалізації алгоритмів навчання з підкріпленням, які працюють в безперервному просторі дій;
- реалізувати архітектуру агента, що навчається з підкріпленням вести діалог з клієнтом;
- дослідити ефективність розробленої архітектури для таск-орієнтованого діалогу.

2 МОВНА МОДЕЛЬ

2.1 Детальний огляд трансформерів

Загальну архітектуру трансформеру можна побачити на рисунку 2.1. Вона складається з двох основних компонентів: енкодера (зліва) та декодера (справа).

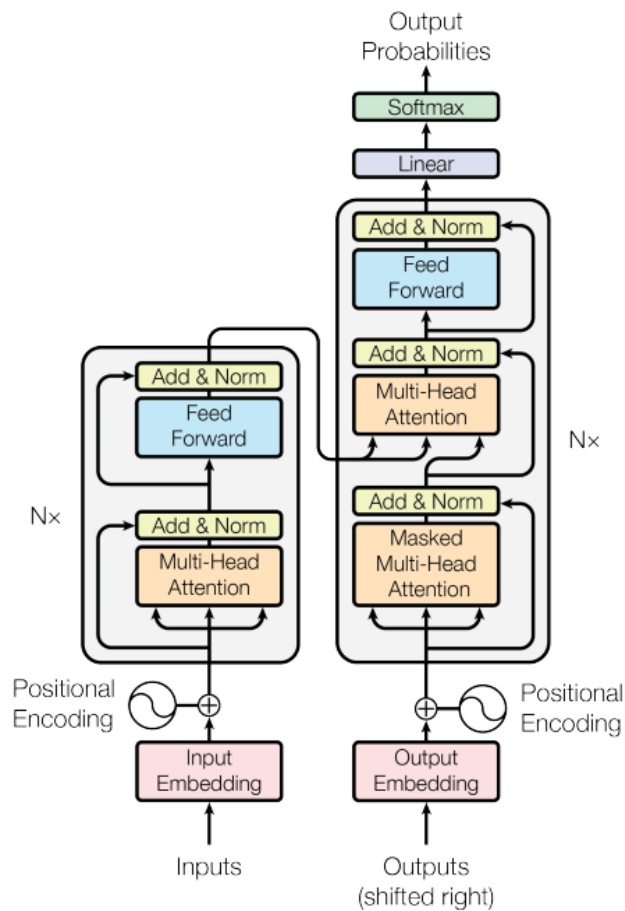


Рисунок 2.1 – Трансформер – архітектура моделі [13]

Енкодер складається з шести ідентичних шарів. Кожен шар має два суб-шари. Перший шар – це багатшаровий механізм самоуваги, другий шар – це повнозв’язна мережа прямого поширення. Також після кожного

шару використовується остаточне з'єднання (residual connection) та нормалізація, позначені на схемі як Add & Norm.

Остаточне з'єднання допомагає передавати інформацію через шар без змін, а потім додавати до цієї інформації вихідні дані шару перед подачею у наступний шар. Ця техніка допомагає запобігти проблемі затухаючого градієнту.

Нормалізація, яка використовується для вихідних даних перед додаванням до вхідних даних наступного шару, допомагає стабілізувати навчання та прискорити сходимість.

Ці техніки допомагають навчати глибокі трансформери більш ефективно та швидко.

Декодер також складається з шести ідентичних шарів. В додаток до двох суб-шарів в кожному шарі енкодера, в декодер додається третій шар, який виконує багато-головий механізм уваги «encoder-decoder attention». Він використовує вихідні дані енкодера в якості key та value, в той час як query формуються на основі попереднього вихідного шару декодера. Це дозволяє декодеру зосередитися на відповідних частинах вхідного речення при генерації наступного слова вихідного речення.

Механізм уваги в трансформерах описується як мапінг між query та key-value парою у виході, де queries, keys, values та outputs являють собою ембедінг вектори.

Outputs вираховуються як зважені суми values, де вага присвоєна кожному value, вираховується функцією сумісності query та key (рисунок 2.2).

Scaled Dot-Product Attention представляє собою з вхід з queries та keys розмірностей d_k та d_v . Вираховується Dot-Product як твір query та усіх ключів, поділених на $\sqrt{d_k}$. Потім до результату приміняється softmax та помножується на values.

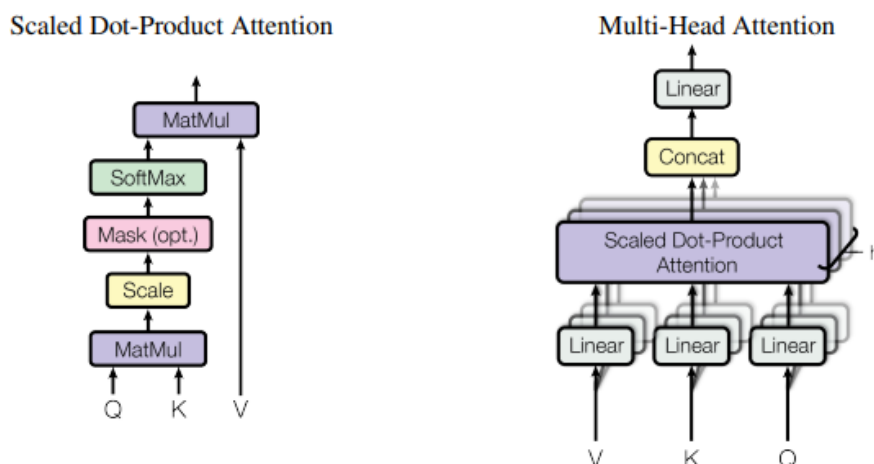


Рисунок 2.2 – Scaled Dot-Product Attention зліва та Multi-Head Attention справа

На практиці функція уваги вираховується на пачці queries одночасно, упакованих у матрицю Q. Keys та Values також складаються у матриці K та V. Матричне рівняння виглядає так:

$$\text{Attention}(Q,K,V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V. \quad (2.1)$$

Замість розрахунку однієї функції уваги з ключами, запитами та значеннями, проєціюються ключі, запити та значення кілька разів, з різноманітними вивченими лінійними проєкціями на розмірності d_k , d_v та d_q відповідно. Для кожної проєцьованої версії keys, queries та values потім паралельно вираховується функція уваги.

Тобто, замість одного механізму уваги, використовується кілька механізмів уваги, або так званий багатоголовий механізм уваги (multi-head attention), який дозволяє зосереджуватися на різних аспекти даних одночасно (рисунок 2.2).

На вхід моделі подається вектор, який отримується шляхом конвертації слів в вектори ембедінгів, вивчених при тренуванні моделі. Через те що модель немає рекурентних та згорткових компонентів, для того

щоб модель розуміла послідовність слів додається «position encodings» до вхідних ембедінгів з самого низу енкодинг та декодинг стеку.

Існує багато варіантів позиційного енкодингу, наприклад, в статті [13] використовується \sin та \cos функції різних частот:

$$PE_{(pos, 2i)} = \sin(pos/10000^{2i/d}), \quad (2.2)$$

$$PE_{(pos, 2i+1)} = \cos(pos/10000^{2i/d}), \quad (2.3)$$

де pos це позиція та i це розмірність. Тобто кожен вимір позиційного кодування відповідає синусоїді.

Трансформер є першою моделлю трансдукції, яка повністю покладається на механізм самоуваги для обчислення вкладень входу та виходу без використання рекурентних нейронних мереж або згорткових.

2.2 Мовна модель GPT-2

Архітектура GPT-2 (Generative Pre-trained Transformer 2) була розроблена OpenAI. Це модель авторегресійного типу, яка призначена для генерування тексту та виконання різноманітних задач обробки натуральної мови без необхідності специфічного налаштування [22].

GPT-2 модель складається з N трансформер декодер блоків, як показано на рисунку 2.3 зліва. Кожен декодер блок, як зображено по центру рисунку, включає багатоголовий маскувальний шар самоуваги, глибоку нейронну мережу, шар нормалізації та dropout шар (рисунок 2.3).

Як ми бачимо, модель базується на архітектурі трансформерів з деякими модифікаціями. Шар нормалізації був переміщений на початок кожного суб-блоку, а також додатковий шар нормалізації був доданий після фінального блоку само-уваги.

Модифікована ініціалізація вагів таким чином, щоб враховувати вплив накопичення сигналів в остаточних зв'язках по мірі заглиблення мережі. Такий підхід допомагає покращити стабільність навчання та прискорити сходимость, забезпечуючи більш рівномірне розподілення активацій і градієнтів вздовж мережі.

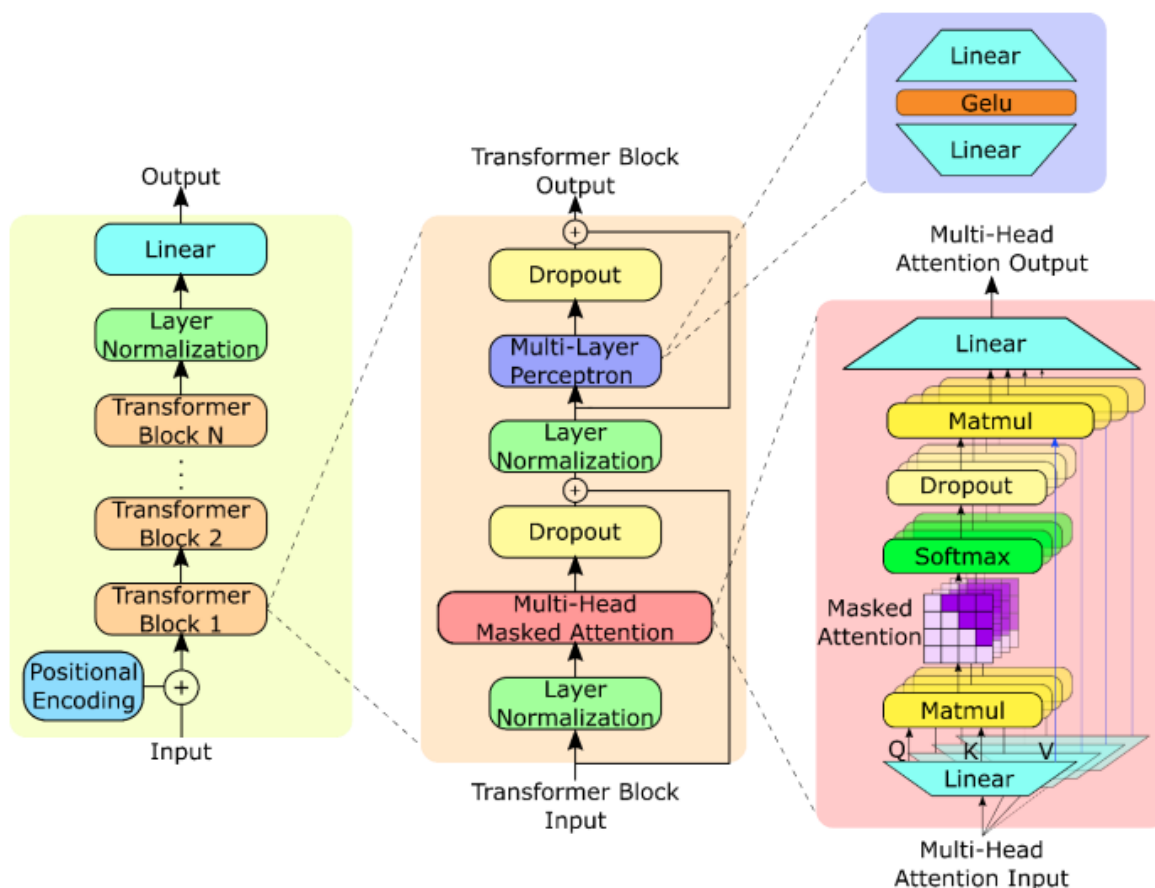


Рисунок 2.3 – GPT-2 архітектура [23]

Ваги остаточних шарів ініціалізуються, масштабуються помноженням на $1/\sqrt{N}$ остаточних шарів. Словник був розширений до 50257 слів. Також було збільшено розмір контексту з 512 до 1024 токенів, і застосовано розмір пакету, збільшений до 512.

Існує кілька версій GPT-2 моделей з різною кількістю параметрів. Їх можна бачити в таблиці 2.1.

Оригінальний трансформер навчався на паралельних корпусах текстів, наприклад, реченнях на різних мовах для машинного перекладу, з використанням методу максимізації правдоподібності.

Таблиця 2.1 – Архітектурні гіпер параметри для чотирьох моделей GPT - 2 [13]

Model	Parameters	Layers	d_{model}
GPT2 Small	117M	12	768
GPT2 Medium	345M	24	1024
GPT2 Large	768M	36	1280
GPT2 Extra Large	1524M	48	1600

Для навчання GPT-2 використовувалося неконтрольоване навчання на великих корпусах текстів без явних паралельних даних. Модель навчається передбачати наступний токен у послідовності на основі попередніх токенів, використовуючи метод максимізації правдоподібності.

GPT-2 має більше параметрів у порівнянні з оригінальним трансформером, що дозволяє їй краще запам'ятовувати та узагальнювати інформацію.

Для використання у конкретних завданнях зазвичай модель донавчається (fine-tuning). Це дозволяє використовувати великі знання, вже накопичені моделлю під час попереднього навчання, та адаптувати їх для конкретного завдання з меншим обсягом даних та меншими обчислювальними витратами.

3 АЛГОРИТМ НАВЧАННЯ З ПІДКРІПЛЕННЯМ

3.1 Алгоритми градієнта стратегії

Як ми вже коротко розглядали вище, коли кількість станів дуже велика, а простір дій безперервний, алгоритми, що навчають функцію цінності або функцію цінності дії, стикаються з великими труднощами.

Алгоритми градієнта стратегії мають великий потенціал у такого рода задачах, тому що легко адаптуються до безперервного простору дій.

Метою навчання з підкріпленням є максимізація очікуваного доходу упродовж траєкторії. Цільова функція виглядає таким чином:

$$J(\theta) = E_{\tau\pi_\theta}[R(\tau)], \quad (3.1)$$

де θ – вектор параметрів стратегії, наприклад навчені ваги глибокої нейронної мережі.

У методах градієнту стратегії для поліпшення цільової функції використовується метод градієнтного підйому, змінюють параметри функції в напрямку градієнту:

$$\theta = \theta - \alpha \nabla_\theta J(\theta). \quad (3.2)$$

Градієнт цільової функції можна описати таким чином:

$$\nabla_\theta J(\theta) = \nabla_\theta E_{\tau\pi_\theta}[R(\tau)]. \quad (3.3)$$

Якщо розписати цю формулу більш детально, одразу виявляється проблема, оскільки при такому формулюванні градієнт цільової функції залежить від розподілу станів стратегії:

$$\nabla_{\theta} J(\theta) = \nabla_{\theta} E_{\tau \pi_0} [R(\tau)] = \nabla_{\theta} \sum_s d(s) \sum_a \pi_{\theta}(a \vee s) R(s, a). \quad (3.4)$$

Можна використовувати стохастичну апроксимацію математичного очікування, але, щоб обчислити розподіл станів $d(s)$, все одно потрібна повна модель навколишнього середовища.

Теорема про градієнт стратегії дає аналітичний вираз для обчислення градієнта цільової функції за параметрами стратегії, в який не входить похідна розподілу станів. Тож, градієнт цільової функції можна записати таким чином:

$$\nabla_{\theta} J(\theta) = E_{\tau \pi_0} [\nabla_{\theta} \log \pi_{\theta}(\tau) R(\tau)] = E_{\tau \pi_0} [\nabla_{\theta} \log \pi_{\theta}(a \vee s) Q_{\pi_0}(s, a)]. \quad (3.5)$$

За допомогою цієї формули, можна оцінити градієнт цільової функції, виконуючи вибірку безпосередньо з досвіду, накопиченого в результаті проходження стратегії.

Найпростіший спосіб розрахунку градієнту цільової функції було запропоновано Вільямсом для алгоритму REINFORCE, який запропонував оцінити дохід за допомогою методу Монте-Карло. Для цього в (3.5) функція цінності дії замінюється доходом Монте-Карло:

$$\nabla_{\theta} J(\theta) = E_{\tau \pi_0} [\nabla_{\theta} \log \pi_{\theta}(a \vee s) Q_{\pi_0}(s, a)] = E_{\tau \pi_0} [\nabla_{\theta} \log \pi_{\theta}(a_i \vee s_i) G_i], \quad (3.6)$$

де G_i – дохід, який обчислюється по повній траєкторії, тоді оновити градієнтну стратегію можна буде після t кроків.

На практиці застосовується знецінений дохід в момент t , який можна також назвати майбутньою винагородою:

$$G_i = \sum_{t'=t}^T \gamma^{t'-t} r(s_{t'}, a_{t'}). \quad (3.7)$$

Формулу можна переписати у рекурентному вигляді:

$$G_t(s_t, a_t) = r(s_t, a_t) + \gamma G_t(s_{t+1}, a_{t+1}). \quad (3.8)$$

Алгоритм REINFORCE непридатний для вирішення безперервних завдань, тому що для обчислення майбутньої винагороди йому потрібно отримати всі винагороди до кінця траєкторії. Методи Actor-Critic використовують техніку бутстрепінга, яка дозволяє навчатися за неповними траєкторіями і оцінювати дохід за цінностями наступних станів.

В алгоритмі Actor-Critic виконавцем є стратегія, а критиком – функція цінності (зазвичай функція цінності станів), яка «критикує» поведінку виконавця, щоб допомогти йому швидше навчитися. Алгоритми Actor-Critic здатні навчитися цінностям дій за неповними траєкторіями.

Функція цінності дій з однокроковим бутстрепінгом визначається наступним чином:

$$Q(s,a) = r + \gamma V(s'), \quad (3.9)$$

де s' – наступний стан.

Крок поновлення в однокрокових алгоритмах Actor-Critic виглядає так:

$$\theta = \theta - \alpha (r_t + \gamma V_w(s') - V_w(s)) \nabla_{\theta} \log \pi_{\theta}(a_t | s_t). \quad (3.10)$$

Функція цінності станів використовується для оцінки цінності наступного стану, і для оцінки $Q(s, a)$ потрібно знати тільки поточну винагороду. Однокроковий Actor-Critic алгоритм повністю онлайн і інкрементний алгоритм.

Зазвичай однокроковий дохід в онлайн алгоритмах замінюють n -кроковим. Це дозволяє подолати проблему зміщення, тому що критик оцінює якість дій спираючись на один крок, а це може призвести до недооцінки або переоцінки дій, тому що інформація обмежена.

Також це дозволяє подолати високу дисперсію, проблему стабільності та інше.

Таким чином, замінюючи однокроковий дохід на $G_{t:t+n}$, ми отримуємо формулу оновлення параметрів:

$$\theta = \theta - \alpha(G_{t:t+n} + \gamma V_w(s') - V_w(s)) \nabla_{\theta} \log \pi_{\theta}(a_t | s_t), \quad (3.11)$$

де $G_{t:t+n} = r_t + \gamma r_{t+1} + \dots + \gamma^{n-1} r_{t+n-1}$. Якщо s_t – заключний стан, то $V(s_{t+1}) = 0$.

Ваги критика оптимізуються за допомогою будь-якого методу стохастичного градієнтного спуску що мінімізує середньоквадратичну помилку [17]:

$$L(w) = \frac{1}{2} \sum_i (V_w(s_i) - y_i)^2. \quad (3.12)$$

3.2 BRAC (behavior regularized actor critic)

В навчанні з підкріпленням передбачається, що агент постійно взаємодіє з середовищем online, але у реальному світі доступ до навколишнього середовища лімітований, або агент навчається на offline наборі даних, тобто на залогованому досвіді. В такому разі, навчання з підкріпленням показує не дуже гарні результати, які розходяться або дають погану продуктивність.

Таким чином, якщо максимізація дії в цільовому значенні ніяк не обмежена, то легко отримати дії, для яких Q-value передбачення хибні. В діалогах ця проблема особливо загострена, тому що алгоритм навчається на тренувальному датасеті, і в реальному діалозі передбачення може бути вкрай хибне.

Для вирішення цієї проблеми існує кілька підходів. Ми розглянемо так званий BRAC, в якому для регуляризації Q-function використовується штрафування значень, що виходять за допустимі межі.

BRAC [24] або behavior regularized actor critic – це алгоритм, який використовує архітектуру Actor-Critic і фокусується на регуляризації поведінки агента, шляхом мінімізації розбіжностей між розподілом дій даної політики та розподілом дій базової політики. Це дозволяє агенту підтримувати бажану поведінку та покращує стабільність і безпечність навчання.

Офлайн навчання з підкріпленням розглядає проблему вивчення політики π з фіксованого датасету D , яка складається з однокрокових переходів (s, a, r, s') . Така стратегія завжди чітко визначена, і стратегія поведінки $\pi_b(a \vee s)$ розглядається як розподіл $p(a/s)$, що спостерігається на датасеті D . Оскільки прямий доступ до π_b не передбачається, політика поведінки апроксимується за допомогою максимальної ймовірності над D :

$$\hat{\pi}_b := \operatorname{argmax}_{\pi} \mathbb{E}_{(s, a, r, s') \sim D} [\log \pi(a|s)], \quad (3.13)$$

де $\hat{\pi}_b$ – поведінкова стратегія, яка позначається з кришкою, щоб розрізнити зі справжньою поведінковою стратегією.

Існує два шляхи впровадження регуляризації в політику: через штрафування у функції цінності (value penalty), або через штраф, який накладається на саму політику (policy regularization).

У value penalty, додається член до цільового значення Q-value, який направляє стратегію π , яка навчається, в напрямку поведінкової стратегії π_b .

Якщо функція цінності виглядає так:

$$V^{\pi}(s) = \sum_{t=0}^{\infty} \gamma^t E_{S_t P_t^{\pi}(s)} [R^{\pi}(s_t)], \quad (3.14)$$

де P_t^{π} – перехідне розподілення для стратегії π , то функція цінності зі штрафом буде виглядати так:

$$V_D^{\pi}(s) = \sum_{t=0}^{\infty} \gamma^t E_{S_t P_t^{\pi}(s)} [R^{\pi}(s_t) - \alpha D(\pi(\cdot \vee s_t), \pi_b(\cdot \vee s_t))], \quad (3.15)$$

де D є функцією розбіжності між розподілами за діями.

Відповідно до типової схеми актор-критик, оптимізація Q -значення задається наступним чином:

$$\min_{Q,\psi} \mathbb{E}_{(s,a,r,s') \sim D} \left[\left(r + \gamma \left(\underline{Q}(s', a') - \alpha \hat{D}(\pi_{\theta}(\cdot \vee s'), \pi_b(\cdot \vee s')) \right) - Q_{\psi}(s, a) \right)^2 \right], \quad (3.16)$$

де \underline{Q} визначає цільове значення Q ,

\hat{D} визначає вибірккову оцінку функцію розбіжності D .

Ціль вивчення стратегії може бути описана так:

$$\max_{\pi_{\theta}} \mathbb{E}_{(s,a,r,s') \sim D}. \quad (3.17)$$

Відповідно, виконуються почергові оновлення градієнтів на основі (3.16) та (3.17). Цей алгоритм еквівалентний SAC, де використовується одно-вибірккова оцінка ентропії для \hat{D} .

Інший шлях додавання регуляризації – це регуляризація стратегії лише під час оптимізації стратегії. Використовуються ті ж самі цілі (3.16) та (3.17), але використовується $\alpha = 0$ при апдейті Q , тоді як $\alpha \neq 0$ при апдейті стратегії. Цей варіант регуляризації автори [24] називають policy regularization. Цей спосіб схожий на регуляризацію, яка використовується у A3C алгоритмі, якщо використовувати ентропію π_{θ} для обчислення \hat{D} .

4 ПОБУДОВА ДІАЛОГОВОГО АГЕНТА

Ціллю практичної частини є створення діалогового агента та його навчання для досягнення певних цілей через побудову функції винагорода і налаштування генерації натуральних відповідей.

4.1 Загальний огляд підходу

Навчання з підкріпленням потребує специфічних датасетів, в яких винагорода позначається тим чи іншим чином. Таких датасетів дуже мало у порівнянні з сирими текстовими даними.

Основна ідея поєднання мовної моделі та навчання з підкріпленням полягає у використанні мовної моделі, навченої на великому обсязі немаркованих даних, яка розуміє контекст та генерує натуральні відповіді, а також у використанні маркованих специфічних даних, на яких донавчається мовна модель, а алгоритм навчання з підкріпленням вивчає певну стратегію поведінки.

За основу взята мовна модель GPT-2, яка донавчається на task-специфічному датасеті [25] CraigslistBargain (рисунок 4.1).

Title: iPhone 5S 16 GB black silver - \$135

Description: Great condition. No scratches. I upgraded to iPhone 7. Please reply with phone number if you are interested. Thank you.

Buyer: Hi, I would like to buy the iPhone 5S

CHAI: Hi, great! The phone is in great condition. I'm asking \$135.00, it's in mint condition.

Buyer: Does it come with a charger?

CHAI: Yes it comes with a charger, as I upgraded. It's in mint condition.

Buyer: How long have you been using this phone?

CHAI: Just a month. It's in great condition and has a charger included.

Buyer: Did you buy this phone secondhand or from the Apple store?

CHAI: I bought it brand new from Apple store. I only used it a month, I'm happy I did not have to use this one.

Buyer: I can buy it for \$100

CHAI: I'll take that if you can pick it up in a few hours.

Buyer: Where will you be at 5pm today?

CHAI: I'm at the Apple store so about an hour.

Buyer: Okay, I can meet you at the Apple store.

CHAI: Awesome!

Buyer: *accept*

Рисунок 4.1 – Приклад оголошення з розмовою між продавцем та покупцем

Датасет містить 6682 різноманітних оголошень з Craigslist, та відповідними діалогами між покупцями та продавцями .

Потім мовна модель використовується для того, щоб згенерувати різноманітні вислови $LM(u|s_{1:t})$, виходячи з історії діалогу, $s_{1:t}$. Після того, як речення згенеровані, будується алгоритм навчання з підкріпленням, у якому навчається Q-function, відповідальна за те, щоб оцінювати гарні та погані відповіді згенеровані мовною моделлю (рисунок 4.2).

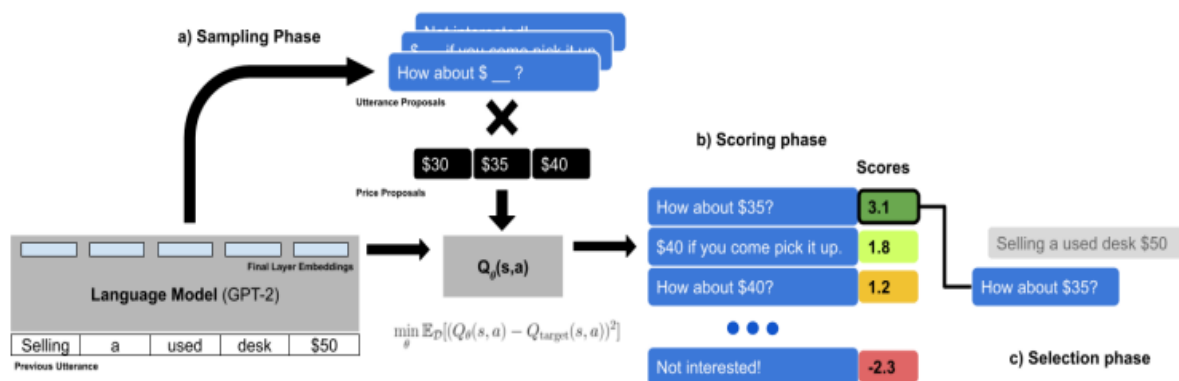


Рисунок 4.2 – Структура моделі. а) донавчена модель генерує різноманітні відповіді в залежності від історії розмови б) оцінка кожної дії критиком с) фінальна відповідь шляхом вибору оцінки Q-value

4.2 Налаштування навчання з підкріпленням

Таск-орієнтований діалог формулюється як проблема навчання з підкріпленням. Агент представляє собою buyer, а навколишнє середовище seller. Розглядається Марківський процес прийняття рішень, де задається кортеж (S, A, T, r, γ) . Простір дій A та станів S складається з чотирьох основних компонентів: type – типу дії, u – вислову, price – ціни, а також context – контексту (рисунок 4.3).

s_u	utterance	Hello!
$s_{context}$	context	Selling a new mountain bike.
s_{price}	price	0.5 (of list price)
s_{type}	action_type	message

Рисунок 4.3 – Приклад стану (State)

На кожному раунді комунікації продавець та покупець можуть відповісти чотирма різними типами відповідей S_{type} : message – дозволяє відправляти будь який вислів один одному, offer – дозволяє зробити якусь пропозицію по покупці, accept – прийняття пропозиції, reject – відмова в пропозиції, яка веде до кінця епізоду.

Стани описуються таким чином: $s = \{s_u, s_{type}, s_{price}, s_{context}\}$, а дії: $a = \{a_u, a_{type}, a_{price}\}$, де s_u та a_u представляють собою вислови, s_{type} та a_{type} – типи дій, s_{price} та a_{price} – ціни, $s_{context}$ – контекст розмови.

Розподіл переходів $T(s'|s,a)$ керує розподілом відповідей, які генерує покупець (середовище).

Ціль алгоритму навчання з підкріпленням – знайти політику $\pi(a/s)$, яка максимізує очікувану винагороду:

$$E_{\pi,T}[\sum_{t=0}^T \gamma^t r(s_t, a_t)], \quad (4.1)$$

де $r: S \times A \rightarrow R$ – функція винагороди і $\gamma \in (0, 1]$ – дисконт-фактор.

В офлайн навчанні з підкріпленням агент взаємодіє з середовищем, щоб максимізувати цю винагороду.

Як вже згадувалося вище, в алгоритмі навчання з підкріпленням навчається критик, який оцінює різноманітні відповіді моделі, основується на їх потенціалі отримати максимальну винагороду. З датасету обирається пакет переходів і оновлюються ваги критика таким чином, щоб мінімізувати Белмановську помилку:

$$J(\theta) = (Q_\theta(s,a) - Q_{target}(s,a))^2, \quad (4.2)$$

де цільове значення $Q_{target}(s,a)$ розраховується через рівняння Белмана:

$$Q_{target}(s,a) = r(s,a) + \gamma E_{s'} [\max_a Q(s',a)]. \quad (4.3)$$

Однак, якщо ми будемо використовувати офлайн датасет, то, використовуючи ці формули, ми стикаємося з проблемою out-of-distribution actions. Для вирішення цієї проблеми використовується алгоритм Behavior-regularized Actor-Critic (BRAC).

Для цього тренується додаткова мережа price proposal network $\pi_\phi(a_{price} \vee s_{1:t})$, яка видає Гаусовий розподіл цін, представлених у діалозі. Використовуючи нотацію $a' \sim \pi_{\phi, \mu}$, ми позначаємо, що ціни беруться з proposal network.

Proposal network тренується згідно з ціллю:

$$\max_{\pi_\phi} E_{s,a,D} [E_{a' \sim \pi_{\phi, \mu}} [Q(s, a')] - D_{KL}(\pi_\phi(\cdot \vee s), \pi_B(\cdot \vee s))], \quad (4.4)$$

Мережа π_ϕ розраховується як уніваріантний умовний Гаусівський розподіл поточної пропозиції за умови попередньої пропозиції, де середнє та стандартне відхилення є лінійними функціями попередньої пропозиції.

Цільова функція вираховується таким чином:

$$Q_{target}(s,a) = r(s,a) + \gamma E_{s', a' \sim \pi_{\phi, \mu}} [Q(s', a')]. \quad (4.5).$$

Тож, регуляризація поведінки контролює запити, які виходять за рамки розподілу за значенням Q-value, регуляризуючи ціни в напрямку тих, що були надані у датасеті.

4.3 Побудова функції винагороди

Одним з найцікавіших та складних етапів розробки діалогового агента є побудова функції винагороди.

Функція винагороди впливає не тільки на поведінку агента, але і на те, наскільки ефективно агент зможе навчатися та досягати певних цілей. Вона повинна чітко відображати задачі, таким чином агент буде розуміти, які дії призводять до позитивних чи негативних результатів. Винагорода повинна бути пропорційна результату і стимулювати агента виконувати ті дії, які призводять до найбільш успішного виконання завдання. Потрібно уникати дуже великих та дуже малих значень, які призводять до нестабільності навчання або ускладнюють його.

Винагорода будувалася таким чином, що якщо діалог завершується виходом одного зі співрозмовників, тобто товар не проданий, то накладається доволі високе константне негативне значення, що стимулює агента продати товар, наприклад -4 . Якщо ціна, за яку було продано товар, має дуже низьке або дуже велике значення, винагорода також визначається високим константним негативним значенням, але більшим ніж -4 , наприклад -2 . Таким чином агент буде намагатися продати товар, але дуже низьку або дуже високу ціну розглядатиме як провал. В іншому випадку винагорода вираховується за формулою $REWARD_SCALE * (price - 1)$, наприклад $2 * (0.8 - 1) = -0.4$. Отже, від нормалізованої ціни віднімають одиницю та домножують на певне значення, тобто, чим вища ціна, за яку продали товар, тим вища винагорода. Тепер агент буде намагатися продати за якомога вищу ціну.

На рисунку 4.3 представлені результати навчання трьох різноманітних критиків з різними функціями винагороди.

На графіку зелена крива – це крива навчання критику, для якого була невірно побудована винагорода, а саме, був накладений дуже великий штраф, коли діалог не призводить до продажу товару.

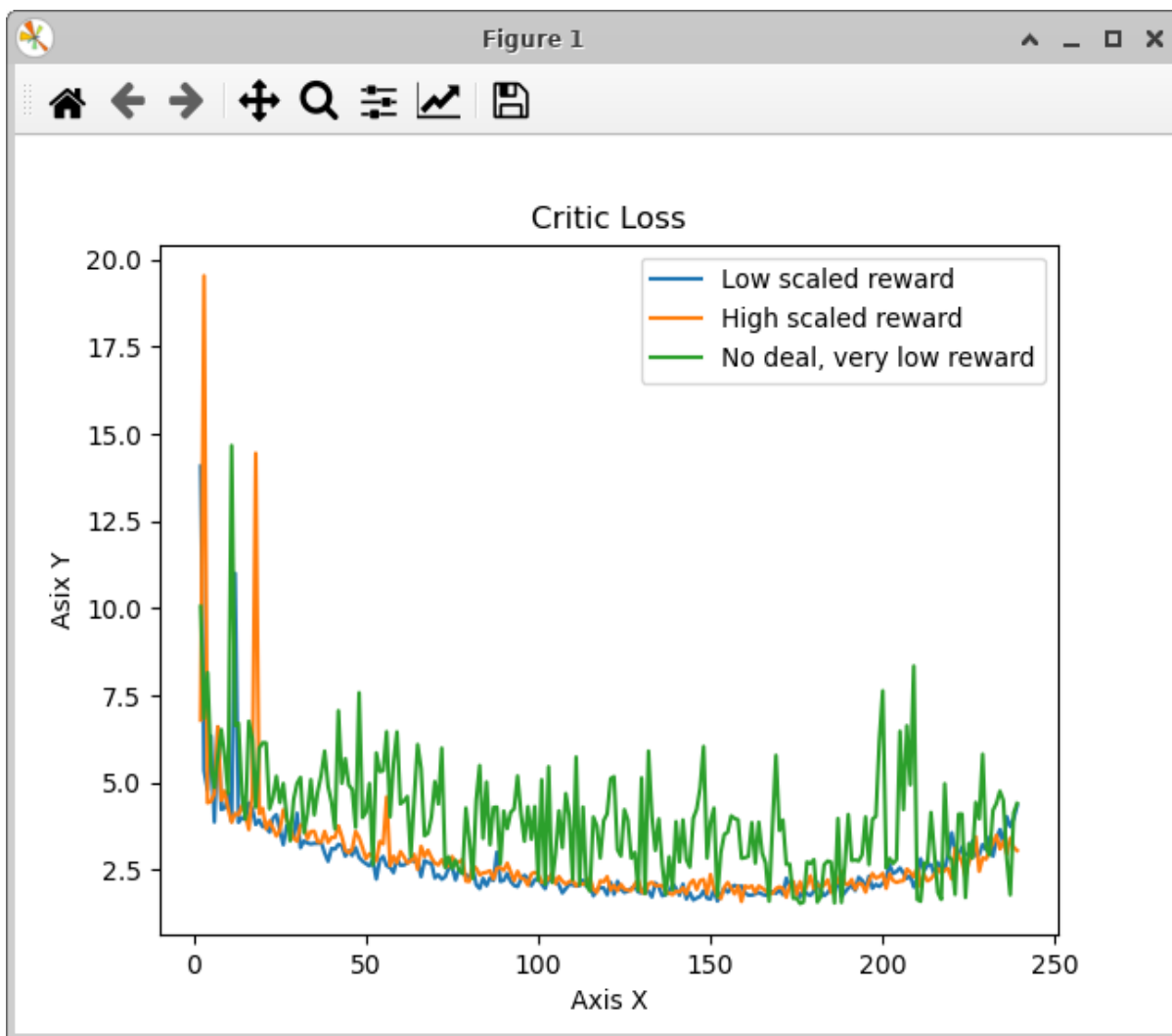


Рисунок 4.3 – Графік навчання критиків

Помаранчева крива відображає навчання критика з високим REWARD_SCALE, синя крива відображає криву навчання критика з невисоким REWARD_SCALE і помірним штрафом за вихід з діалогу без продажу товару. Можна зробити висновок, що ці функції винагороди побудовані доволі непогано.

4.4 Навчання діалогового агенту

Для створення таск-орієнтованої діалогової системи була обрана готова модель GPT2-medium. Модель була донавчена на CraigslistBargain [25] датасеті, в якому містяться контекст (заголовок, опис

оголошення) та сам діалог. Ціни у діалозі були замасковані спеціальним токеном, щоб GPT модель могла генерувати шаблони, в які можна буде підставляти ціни. Тренування відбувалося на протязі 2000 епох.

Q-function представляє собою мережу прямого поширення, яка відображає стани та дії в скалярне представлення Q-value. Стани і дії конкатенуються з цінами і типами висловів (представляється як one-hot вектор) для того, щоб отримати вектор, який подається на вхід критика.

Критик – це двошарова мережа прямого поширення з 256 прихованими вузлами та RELU активаціями.

Для того, щоб згенерувати вислови, конкатенується контекст з діалогом і передається в модель, яка генерує вислови. Процес повторюється, щоб отримати багато зразків.

Нижче описаний процес генерації входу Q-function. «STRCAT» – конкатенує контекст і вислови з префіксами «Buyer:» та «Seller:». «EMBED» – розраховує masked attention embeddings з GPT2 моделі. Де dialog представляє собою історію включно до поточного стану $[s_{u,1}, a_{u,1}, \dots, s_{u,t-1}, a_{u,t-1}]$ та candidate який позначає вислів кандидат $a_{u,t}$, який генерується мовною моделлю та буде розглядатися для оцінювання:

$$s_{\text{embed}} = \text{EMBED}(\text{STRCAT}(s_{\text{context}}, \text{dialog})), \quad (4.6)$$

$$a_{\text{embed}} = \text{EMBED}(\text{STRCAT}(s_{\text{context}}, \text{dialog} + \text{candidate})), \quad (4.7)$$

$$s = [s_{\text{embed}}, s_{\text{price}}, s_{\text{type}}], \quad (4.8)$$

$$a = [a_{\text{embed}}, a_{\text{price}}, a_{\text{type}}], \quad (4.9)$$

$$q = Q(s, a). \quad (4.10)$$

Тренування агента відбувалося у кілька етапів у Google Colab за допомогою безкоштовних ресурсів Tesla T4 з 16GB of GDDR6 пам'яті та 2,560 CUDA ядрами.

Перший етап – це донавчання GPT2-medium моделі, яке займає близько 9 годин. Другий етап – це генерування речень та ембедінгів, який також займає приблизно 9 годин. Останній етап – це навчання BRAC моделі.

При навчанні BRAC з використанням стандартних гіпер параметрів з SAC імплементації, adam оптимізатору, learning rate критика $3 \cdot 10^{-4}$, learning rate актора $3 \cdot 10^{-4}$ та soft target update rate = 0.05, навчання вже на другій-третьій ітерації виходить з під контролю, critic loss починає неймовірно швидко зростати.

При зміні оптимізатору з adam на SGD та зниженні learning rate критика до $1 \cdot 10^{-6}$ та learning rate актора $3 \cdot 10^{-5}$ вдається досягти стабільних результатів навчання. Тренування відбувається на протязі 250 епох та займає близько години, після чого проявляються ознаки перенавчання.

4.5 Оцінка результатів роботи моделі

Для того, щоб мовна модель генерувала більше різноманітних і натуральних відповідей, було підлаштовано деякі параметри моделі.

Наприклад, використано такий параметр, як beam-size = 5, який дозволяє моделі генерувати не один найбільш імовірний токен, а відразу декілька, і так продовжується далі для кожного згенерованого токена. Таким чином ми отримуємо декілька варіантів згенерованого тексту, і в результаті обирається той, в якого найкраща perplexity, тобто впевненість моделі в реалістичності тексту.

Також було задано параметри top-k = 50 или top-p = 0.85, які обмежують генерацію некоректних токенів.

На кожну репліку покупця модель генерує до п'яти різноманітних відповідей (рисунок 4.4).

buyer: Sure, tell me more about it

Q:1.292782 [It is a very nice table. It has been well kept. It is very solid Rosewood.
 Q:1.292782 [It is a very nice table. It has been well kept. It is very solid Rosewood.
 Q:1.292782 [It is a very nice table. It has been well kept. It is very solid Rosewood.
 Q:1.292782 [It is a very nice table. It has been well kept. It is very solid Rosewood.
 Q:1.292782 [It is a very nice table. It has been well kept. It is very solid Rosewood.
 Q:1.292782 [It is a very nice table. It has been well kept. It is very solid Rosewood.
 Q:1.292782 [It is a very nice table. It has been well kept. It is very solid Rosewood.
 Q:1.302622 [It is a very beautiful table. It has been well kept. It is very sturdy.
 Q:1.302622 [It is a very beautiful table. It has been well kept. It is very sturdy.
 Q:1.302622 [It is a very beautiful table. It has been well kept. It is very sturdy.
 Q:1.328688 [It is a very nice table. It has been well kept. It is very solid Rosewood.
 Q:1.328688 [It is a very nice table. It has been well kept. It is very solid Rosewood.
 Q:1.328688 [It is a very nice table. It has been well kept. It is very solid Rosewood.
 Q:1.328688 [It is a very nice table. It has been well kept. It is very solid Rosewood.
 Q:1.328688 [It is a very nice table. It has been well kept. It is very solid Rosewood.
 Q:1.328688 [It is a very nice table. It has been well kept. It is very solid Rosewood.
 Q:1.328688 [It is a very nice table. It has been well kept. It is very solid Rosewood.
 Q:1.328688 [It is a very nice table. It has been well kept. It is very solid Rosewood.
 Q:1.328688 [It is a very nice table. It has been well kept. It is very solid Rosewood.
 Q:1.333442 [It is a very beautiful table. It has been well kept. It is very sturdy.
 Q:1.333442 [It is a very beautiful table. It has been well kept. It is very sturdy.
 Q:1.364280 [It is a very nice table with lots of drawers. It is Rosewood so it has a lot of character.
 The price is 510.
 Q:1.364342 [It is a very nice table with lots of drawers. It is Rosewood so it has a lot of character.
 The price is 569.
 Q:1.410853 [It is a very nice table with lots of drawers. It is Rosewood so it has a lot of character.
 The price is 539.
 Q:1.410876 [It is a very nice table with lots of drawers. It is Rosewood so it has a lot of character.
 The price is 600.
 Q:1.410876 [It is a very nice table with lots of drawers. It is Rosewood so it has a lot of character.
 The price is 480.

Рисунок 4.4 – Відповідь моделі

Повний результат розмови з діалоговим агентом можна побачити у додатку А.

Як бачимо, конструювання функції винагороди правильним чином є одним з ключових факторів в успіху створення таск-орієнтованих діалогових систем. Правильно сконструйована функція винагороди не

тільки впливає на навчання алгоритму, але і дозволяє вирішити поставлену задачу. Згідно з нашою ціллю, ми можемо змусити систему продати товар за будь яку ціну, влаштувати так званий розпродаж, або намагатися продати товар, але при цьому отримати максимальну вигоду, тощо.

Не менш важливим фактором є налаштування мовної моделі. Від цього залежить натуральність відповідей та їх різноманіття, що також впливає на бажання користувача продовжувати розмову.

Сильною стороною побудованої системи є те, що ми розуміємо, який компонент системи ми хочемо покращити, і як це вплине на кінцевий продукт.

ВИСНОВКИ

Судячи з огляду ринків, попит на інтелектуальні чат боти найближчим часом буде тільки зростати. Системи, орієнтовані на завдання, продовжують набирати популярність, оскільки різні галузі знаходять способи ефективного використання їхніх можливостей. Вони дозволяють знизити операційні витрати, а це, у свою чергу, призведе до збільшення використання чат-ботів та зростання ринку.

Інтелектуальні чат боти можуть бути успішно використані у багатьох сферах, наприклад навчанні, де система може представляти собою персонального вчителя, який слідкує за прогресом учня, дає персональні рекомендації. Такий самий приклад можна навести у сфері медицини, де таск-орієнтовані системи можуть розглядатися як персональні лікарі, які враховують всю наявну інформацію та дають певні рекомендації. У бізнесі, де можна автоматизувати багато різноманітних процесів, підхід надасть змогу скоротити витрати.

Загалом в таск-орієнтованих діалогових системах є багато сильних сторін, таких як:

- зниження впливу людського фактору;
- економія на людській праці;
- підвищення швидкості виконання операцій;
- цілодобова доступність;
- збільшення кількості оброблених запитів клієнтів;
- персоналізація запитів та ін.

Якщо ми подивимося на ці сильні сторони, і уявімо таку сферу використання як медицина, то стає зрозумілим, що не кожен лікар може врахувати всю наявну інформацію пацієнта, допомогти серед ночі та інше.

Сучасні мовні моделі здобули значних успіхів у перекладах, підсумуванні будь-якої інформації, або відповідях на загальні питання у

натуральному для людини вигляді, але цього не достатньо для того, щоб вирішувати завдання, ціль яких співпадає з намірами користувача.

В даній роботі були розглянуті різноманітні властивості, притаманні мовним моделям та алгоритмам навчання з підкріпленням, а також варіанти їх поєднання, щоб отримати переваги одного та іншого підходу.

Зокрема, в роботі було приділено увагу end-to-end підходу побудови task-орієнтованих діалогових агентів. В ній розглядається варіант поєднання такої мовної моделі, як GPT-2 на базі трансформеру, та алгоритмів навчання з підкріпленням, а саме алгоритмів з необмеженою кількістю станів та безперервним простором дій, таких як Actor-Critic та його модифікації.

В результаті цього дослідження було успішно поєднано офлайн-методи навчання з підкріпленням з мовними моделями для створення реалістичних агентів діалогу, які краще досягають цілей завдань.

Результати роботи можуть слугувати підґрунтям для створення task-орієнтованих діалогових систем, які можуть бути успішно застосовані у різноманітних галузях і затребувані у бізнесу.

В результаті практичної роботи вдалося побудувати task-орієнтовану діалогову систему, яка показала доволі непогані результати. Встановлено, що офлайнове навчання з підкріпленням може успішно використовуватися у складних людських інтерактивних завданнях, таких як діалог.

Дизайн винагороди сам по собі може бути складною проблемою, але він надає більш прямий важіль впливу на поведінку агента, ніж той, що доступний у стандартних мовних моделях, які можуть бути спрямовані або через вибір навчальних даних, або через інші непрямі механізми.

Подальшим розвитком цієї роботи може бути більш детальний розгляд побудови функції винагороди для різноманітних цілей, або розгляд поєднання навчання з підкріпленням з іншими мовними моделями такими як BERT, T5 та іншими.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Balaraman V., Sheikhalishahi S., Magnini B. Recent Neural Methods on Dialogue State Tracking for Task-Oriented Dialogue Systems: A Survey // Proceedings of the 22nd Annual Meeting of the Special Interest Group on Discourse and Dialogue, м. Singapore and Online. Stroudsburg, PA, USA, 2021. URL: <https://doi.org/10.18653/v1/2021.sigdial-1.25> (дата звернення: 23.03.2024).
2. Chatbot Market Size, Share & Trends, Analysis Report By Application (Customer Services, Branding & Advertising), By Type, By Vertical, By Region (North America, Europe, Asia Pacific, South America), And Segment Forecasts, 2023–2030.
3. Kumar S., Bhatia S., Aggarwal M., Chakraborty T. Dialogue Agents 101: A Beginner’s Guide to Critical Ingredients for Designing Effective Conversational Systems. 14 Jul 2023. arXiv.org e-Print archive. URL: <https://arxiv.org/pdf/2307.07255.pdf> (дата звернення: 28.03.2024).
4. Kjærø A., Bugge E.S., Vennerød C.B. Long Short-term Memory RNN. Department of Computer Science. May 2021. arXiv.org e-Print archive. URL: <https://arxiv.org/pdf/2105.06756.pdf> (дата звернення: 28.03.2024).
5. Cahuantzi R., Chen X., Güttel S. A comparison of LSTM and GRU networks for learning symbolic sequences. Jan 2023. arXiv.org e-Print archive. URL: <https://arxiv.org/pdf/2107.02248.pdf> (дата звернення: 28.03.2024).
6. Bird G., Polivoda M. Backpropagation Through Time For Networks With Long-Term Dependencies. January 2021. arXiv.org e-Print archive. URL: <https://arxiv.org/pdf/2103.15589.pdf> (дата звернення: 13.04.2024).
7. Hochreiter S., Schmidhuber J. Long Short-term Memory // *Neural computation*. 1997. Vol. 9, No. 12. P. 1735–1780.
8. Chung J., Gulcehre C., Cho K.H., Bengio Y. Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling. Dec 2014. arXiv.org

e-Print archive. URL: <https://arxiv.org/pdf/1412.3555.pdf> (дата звернення: 10.04.2024).

9. Schmidt R.M. Recurrent Neural Networks (RNNs): A gentle Introduction and Overview. Nov 2019. arXiv.org e-Print archive. URL: <https://arxiv.org/pdf/1912.05911.pdf> (дата звернення: 05.04.2024).

10. Sutskever I., Vinyals O., Le Q.V. Sequence to Sequence Learning with Neural Networks. Dec 2023. arXiv.org e-Print archive. URL: <https://arxiv.org/pdf/1409.3215.pdf> (дата звернення: 06.04.2024).

11. Mnih V., Heess N., Graves A., Kavukcuoglu K. Recurrent Models of Visual Attention. Jun 2014. arXiv.org e-Print archive. URL: <https://arxiv.org/pdf/1406.6247.pdf> (дата звернення: 06.04.2024).

12. Bahdanau D., Cho K.H., Bengio Y. Neural machine translation by jointly learning to align and translate. May 2016. arXiv.org e-Print archive. URL: <https://arxiv.org/pdf/1409.0473.pdf> (дата звернення: 08.04.2024).

13. Vaswani A. et al. Attention is All you Need // *Advances in Neural Information Processing Systems*. Ed. by I. Guyon et al. Curran Associates, Inc., 2017. Vol. 30. P. 5998–6008.

14. Devlin J., Chang M.-W., Lee K., Toutanova, K. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding // *NAACL-HLT*, 24-May-2019. arXiv.org e-Print archive. URL: <https://arxiv.org/abs/1810.04805> (дата звернення: 09.04.2024).

15. Radford A., Wu J., Child R., Luan D., Amodei D., Sutskever I. Language Models are Unsupervised Multitask Learners, 2019. Available: <https://paperswithcode.com/paper/language-models-are-unsupervised-multitask> (дата звернення: 10.04.2024).

16. Raffel C., Shazeer N., Roberts A., Lee K. Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer. Sep 2023. arXiv.org e-Print archive. URL: <https://arxiv.org/pdf/1910.10683.pdf> (дата звернення: 10.04.2024).

17. Sutton R.S., Barto A.G. Reinforcement Learning: An Introduction // *IEEE Transactions on Neural Networks*. 1998. Vol. 9, No. 5. P. 1054. URL: <https://doi.org/10.1109/tnn.1998.712192> (дата звернення: 23.05.2024).
18. AlMahamid F., Grolinger K. Reinforcement Learning Algorithms: An Overview and Classification. – Department of Electrical and Computer Engineering Western University. 29 Sep 2022. arXiv.org e-Print archive. URL: <https://arxiv.org/pdf/2209.14940.pdf> (дата звернення: 10.03.2024).
19. Sutton R.S. Learning to predict by the methods of temporal differences // *Machine Learning*. 1988. Vol. 3, No. 1. P. 9–44. URL: <https://doi.org/10.1007/bf00115009> (дата звернення: 23.05.2024).
20. Liu Y., Ramachandran P., Liu, Q., Peng, J. Stein Variational Policy Gradient. Apr 2017. arXiv.org e-Print archive. URL: <https://arxiv.org/pdf/1704.02399.pdf> (дата звернення: 05.03.2024).
21. Kwan W.-C., Wang H., Wang H., Wong K.-F. A Survey on Recent Advances and Challenges in Reinforcement Learning Methods for Task-Oriented Dialogue Policy Learning. The Chinese University of Hong Kong. Jul 2022. arXiv.org e-Print archive. URL: <https://arxiv.org/pdf/2202.13675.pdf> (дата звернення: 03.04.2024).
22. Radford A., Wu J., Child R., Luan D., Amodei D., Sutskever I. Language Models are Unsupervised Multitask Learners. OpenAI blog, 2019.
23. Yang S.D., Ali Z.A., Wonga B.M. FLUID-GPT (Fast Learning to Understand and Investigate Dynamics with a Generative Pre-Trained Transformer): Efficient Predictions of Particle Trajectories and Erosion. Materials Science & Engineering Program, University of California-Riverside, Riverside, CA 92521, USA.
24. Wu Y., Tucker G., Nachum O. Behavior Regularized Offline Reinforcement Learning. 26 Nov 2019. arXiv.org e-Print archive. URL: <https://arxiv.org/pdf/1911.11361.pdf> (дата звернення: 03.04.2024).
25. Decoupling Strategy and Generation in Negotiation Dialogues / Н. Не та ін. Proceedings of the 2018 Conference on Empirical Methods in Natural

Language Processing, м. Brussels, Belgium. Stroudsburg, PA, USA, 2018. URL:
<https://doi.org/10.18653/v1/d18-1256> (дата звернення: 23.05.2024).