Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

Факультет навчально-науковий центр заочної форми навчання
(повна назва)

Кафедра електронних обчислювальних машин
(повна назва)

АТЕСТАЦІЙНА РОБОТА

Пояснювальна записка

Рівень вищої освіти другий (магістерський)

Модель автоматичного навчання чат-бота
на основі системи обробки природньої мови LUIS

(тема)

Виконав:
студент II курсу, групи СПзм-18-2
Гожа І.М.
(прізвище, ініціали)

Спеціальність
123 – Комп’ютерна інженерія
(код і повна назва спеціальності)

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

Освітня програма
Системне програмування
(повна назва освітньої програми)

Керівник: проф. Смеляков С.В.
(посада, прізвище, ініціали)

Допускається до захисту

Зав. кафедри ЕОМ _____ Коваленко А.А.
(підпис) (прізвище, ініціали)

2020 р.

Харківський національний університет радіоелектроніки

Факультет навчально-науковий центр заочної форми навчання

Кафедра електронних обчислювальних машин

Рівень вищої освіти другий (магістерський)

Спеціальність 123 – Комп'ютерна інженерія
(код і повна назва)

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

Освітня програма Системне програмування
(повна назва)

ЗАТВЕРДЖУЮ:
Зав. кафедри _____
(підпис)
"____" _____ 20___ р.

ЗАВДАННЯ

НА АТЕСТАЦІЙНУ РОБОТУ

студентові Сидорову Дмитру Сергійовичу
(прізвище, ім'я, по батькові)

1. Тема роботи Методи та засоби аналізу мережного трафіку

затверджена наказом по університету від "30" березня 2020 р. № 43 Стз

2. Термін подання студентом роботи до екзаменаційної комісії 18 травня 2020 р.

3. Вхідні дані до роботи

1. Перелік існуючих технологій реалізації чат-ботів.
2. Існуючи моделі розподіленого програмного забезпечення чат-ботів.
3. Методи створення чат-ботів.
4. Множина основних параметрів чат-ботів

4. Перелік питань, що потрібно опрацювати в роботі

1. Аналіз предметної області і постановка мети та завдань дослідження.
2. Загальна модель багаторівневого чат-бота
3. Розробка програмних засобів підтримки чат-ботів

5. Перелік графічного матеріалу із зазначенням креслеників, схем, плакатів, комп'ютерних ілюстрацій (слайдів) 12 слайдів

6. Консультанти розділів роботи (заповнюється за наявності консультантів згідно з наказом, зазначеним у п.1)

Найменування розділу	Консультант (посада, прізвище, ім'я, по батькові)	Позначка консультанта про виконання розділу	
		підпис	дата

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів роботи	Термін виконання етапів роботи	Примітка
1	Огляд предметної області та постановка	31.03.20-10.04.20	
	завдання		
2	Вибір та обгрунтування методики	11.04.20-20.04.20	
	дослідження		
3	Вибір інструментальних засобів	21.04.20-25.04.20	
4	Проведення експериментів	26.04.20-05.05.20	
	Оформлення ПЗ	06.05.20-11.05.20	
6	Подання атестаційної роботи	12.05.20-13.05.20	

Дата видачі завдання 30 березня 2020 р.

Студент _____
(підпис)

Керівник роботи _____ проф. Смеляков С.В.
(підпис) (посада, прізвище, ініціали)

РЕФЕРАТ

Пояснювальна записка атестаційної роботи: 82 с., 13 рис.., 1 дод., 42 джерела.

РОЗПОДІЛЕНЕ ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ, ЧАТ-БОТ, СЕРВЕР, МАШИННЕ НАВЧАННЯ, ВЕЛИКІ ДАНІ.

Метою магістерської атестаційної роботи є розробка моделі і метода адаптації багаторівневого чат-бота, які дозволять автоматизувати процес створення бота з урахуванням індивідуальних особливостей користувача.

У роботі було вирішено ряд задач: проведено аналіз сучасного стану програмних засобів створення та імплементації чат-ботов, а також інформаційних технологій, які можливо використовувати для їх впровадження; створення моделі багаторівневого чат-бота; розроблено метод адоптації багаторівневого чат-бота к користувачу; розроблені програмні засоби, які реалізують взаємодію розподілених компонентів чат-бота.

ABSTRACT

Master’s thesis: 82 pages, 13 figures,1 appendics, 42 sources.

DISTRIBUTED SOFTWARE, CHAT BOT, SERVER, MACHINE LEARNING, BIG DATA.

The purpose of the master’s certification work is to develop a model and method of adaptation of a multilevel chatbot, which will automate the process of creating a bot taking into account the individual characteristics of the user.

A number of tasks were solved in the work: the analysis of the current state of software tools for creating and implementing chatbots, as well as information technologies that can be used for their implementation; creation of a multilevel chatbot model; developed a method of adapting a multilevel chatbot to the user; developed software that implements the interaction of distributed components of the chatbot.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

NLP – natural language processing

SDK – набір засобів розробки

SQL – мова структурованих запитів

ВСТУП

Чат-боти – це вид програмного забезпечення, вони дуже різні з точки зору очікувань користувачів. Чат-боти спрямовані на підтримання діалогу з людиною. Ця здатність обробляти мову змушує проектувати їх свого роду людську особистість і інтелект, незалежно від того, хочемо ми цього чи ні. Розробити програмне забезпечення з особистістю і інтелектом досить складно.

Метою магістерської атестаційної роботи є розробка моделі і метода адаптації багаторівневого чат-бота, які дозволять автоматизувати процес створення бота з урахуванням індивідуальних особливостей користувача.

Для реалізації мети було вирішено ряд задач:

- проведено аналіз сучасного стану програмних засобів створення та імплементації чат-ботов, а також інформаційних технологій, які можливо використовувати для їх впровадження;

- створено модель багаторівневого чат-бота;

- розроблено метод адоптації багаторівневого чат-бота к користувачу;

- розроблені програмні засоби, які реалізують взаємодію розподілених компонентів чат-бота.

Об’єктом дослідження є процес створення та функціонування програмного забезпечення багаторівневого чат-бота.

Предметом дослідження є моделі, методи та інформаційні технології управління процесом створення та функціонування розподіленого програмного забезпечення чат-бота.

До використаних методів дослідження можна віднести: критичний аналіз теоретичних і прикладних розробок вітчизняних і зарубіжних вчених у галузі розробки моделей та методів розподіленого програмного забезпечення хмарних систем. Для вирішення поставлених завдань використані методи системного аналізу, теорії множин (для розробки моделі багаторівневого чат-

бота), загальної теорії систем (метода адоптації багаторівневого чат-бота к користувачу), теорії імітаційного моделювання (для моделювання обчислювальних процесів).

Практична значення отриманих результатів підтверджується наявним програмним забезпеченням, яке демонструє можливість реалізації запропонованих теоретичних досліджень при розробці чат-ботів. Як приклад, у роботі наведені компоненти, які дозволяють використовувати дві різні платформи для створення чат-ботів для відомого сервісу обміну повідомленнями Telegram.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ І ПОСТАНОВКА ЦІЛІ І ЗАДАЧ ДОСЛІДЖЕННЯ

1.1 Аналіз програмних інструментальних засобів розробки

У даному підрозділі описано деякі програмні інструменти, які використані в роботі щодо створення, розміщення та функціонування чат–ботів.

Heroku – це хмарна платформа, яка дозволяє розміщувати веб-додатки в хмарі. Щоб почати роботу, необхідно зареєструватися з безкоштовною обліковим записом на сайті www.heroku.com і встановити консольний інструмент Heroku на сторінці devcenter.heroku.com/articles/heroku-cli.

Git – це інструмент управління версіями. Цій інструмент буде інтенсивно використовувати Git, оскільки це частина процесу Heroku. Потрібно встановити консольний інструмент Git в системі. Його можна знайти на сторінці git-scm.com/downloads.

Ядро Git є набором утиліт командного рядка з параметрами. Всі налаштування зберігаються в текстових файлах конфігурації. Така реалізація забезпечує Git легко портіруемость на будь-яку платформу і дає можливість легко інтегрувати Git в інші системи (зокрема, створювати графічні git-клієнти з будь-яким бажаним інтерфейсом).

Репозиторій Git є каталогом файлової системи, в якому знаходяться файли конфігурації сховища, файли журналів, що зберігають операції, що виконуються над репозиторієм, індекс, що описує розташування файлів, і сховище, що містить власне файли. Структура сховища файлів не відображає реальну структуру зберігається в репозиторії файлового дерева, вона орієнтована на підвищення швидкості виконання операцій з репозиторієм. Коли ядро обробляє команду зміни (неважливо, при локальних змінах або при отриманні патча від іншого вузла), воно створює в сховище нові файли,

що відповідають новим станам змінених файлів. Істотно, що ніякі операції не змінюють вмісту вже існуючих в сховище файлів.

Node.js – це середовище виконання JavaScript, в якій знадобиться створювати веб-додатки в більшості проектів. Знайти її дистрибутив можна на сторінці nodejs.org/en/download/.

Java SDK і Eclipse: Java 1.8 використовується для створення чат бота. Потрібно встановити Java SDK і середовище розробки Eclipse, щоб виконувати інструкції. Крім того, можна запрограмувати чат-бота, використовуючи Node.js або іншу мову програмування, підтримуваний Heroku. Знайти дистрибутив можна на сайті www.oracle.com/technetwork/java/javase/ downloads / index.html, а Eclipse – на сайті www.eclipse.org/downloads/.

1.2 Загальна характеристика предметної області

Зараз настала ера інтелектуальних машин, в якій лідирують діалогові інтерфейси. За останні пару років усі були охоплені безліччю нових видів машин і програмного забезпечення, загальновідомих як боти. Боти – це автоматизовані апаратні або програмні засоби, які засновані на технологіях штучного інтелекту (ІІ). Недавні розробки в алгоритмах машинного навчання, такі як глибоке навчання і поглиблене закріплення, поліпшили продуктивність завдань ІІ, таких як автоматичне розпізнавання мови (ASR), розпізнавання природної мови (NLU), синтез тексту в мову (TTS) та розпізнавання зображень. Це скоротило час шляху людства до технологічної сингулярності, момент, коли штучний інтелект перевершує природний людський розум не по днях, а по годинах.

Однією з довгострокових цілей в галузі ШІ є створення комп’ютерних систем, які зможуть проводити людиноподібні бесіди з користувачами. Завдяки недавнім досягненням в області технологій ІІ ми зараз на один крок ближче до досягнення цієї мети. Тепер вже не за горами, що ми можемо

взаємодіяти з пристроями і гаджетами в наших будинках і офісах, використовуючи тільки голос. Нам ще належить пройти довгий шлях зі створення стандартів і цифрових сутностей, здатних безперешкодно розмовляти на природній мові. Однак недавній сплеск інтересів і величезні інвестиції в просування цих ідей свідчать про те, що ми знаходимося на шляху до розробки такого глобального стандарту. Дизайнер Марк Кертіс ввів поняття розмовної сингулярності, коли діалогові пристрої зникають.

Сьогодні прості кнопкові чат-боти можуть бути створені без необхідності написати хоч один рядок коду. Інші мають більш складну і гнучку архітектуру, канали для використання, такі як Facebook Messenger, SMS і Twitter, інструменти для розпізнавання природної мови і управління розмовами в міру продовження, голосовий зв’язок на таких платформах, як Amazon Alexa і Google Assistant.

1.3 Діалогові інтерфейси для користувача

Діалогові призначені для користувача інтерфейси так само старі, як і зі тимчасові комп’ютери. ENIAC, перший програмований універсальний комп’ютер, був побудований в 1946 році. У 1950 році Алан Тьюринг, британський комп’ютерний вчений, запропонував виміряти рівень інтелекту в машинах з використанням діалогового тесту, званого тестом Тьюрінга. Тест полягав у тому, що машина конкурувала з людиною як партнером по діалогу і суддів-людей (ще одну людину). Судді повинні були взаємодіяти з кожним з двох учасників (людиною і машиною), використовуючи інтерфейс текстового типу, який не схожий на більшість сучасних додатків чату обміну повідомленнями. В ході бесіди судді повинні були визначити, хто з двох учасників був машиною. Якщо, щонайменше, 30% суддів не змогли виявити різницю між двома учасниками, вважалося, що машина пройшла тест. Це була одна з найбільш ранніх концепцій по діалоговим інтерфейсів і їх відношенню до рівня інтелекту машин, які мають такі можливості. Однак

спроби створення подібних інтерфейсів були успішними протягом кількох наступних десятиліть.

Протягом приблизно 40 років, з 1980-х років, графічні інтерфейси (GUI) були домінуючим способом взаємодії з машинами. З недавнім розвитком в області ІІ і зростаючими обмеженнями, такими як скорочення розмірів гаджетів (від ноутбуків до мобільних телефонів), скорочення екранних об’єктів (смарт-годинник), і необхідністю того, щоб інтерфейси стали невидимими (розумний будинок і роботи), діалогові призначені для користувача інтерфейси знову стають реальністю.

Наприклад, кращим способом взаємодії з мобільними роботами, використовуваними в розумних будинках, буде голос. Таким чином, система повинна розпізнавати запити користувачів і відповідати на природну людську мову. Такі можливості систем допоможуть зменшити наші зусилля у вивченні і розумінні сучасних складних інтерфейсів.

Діалогові призначені для користувача інтерфейси відомі під декількома назвами: інтерфейси природної мови, розмовні діалогові системи, розмовні чат-боти, інтелектуальні віртуальні агенти, віртуальні помічники і т. Д. Фактична різниця між цими системами пов’язана з інтеграцією в них серверних елементів (наприклад, баз даних або модулів завдань / управління), модальностями (наприклад, текстовими, голосовими і візуальними аватарами) і каналами, на яких вони розгорнуті. Однак одним із загальних моментів для цих систем є їх здатність взаємодіяти з користувачами в розмовній манері з використанням природної мови.

1.4 Історія розвитку чат-ботів

Витоки сучасних чат-ботів можна простежити до 1964 року, коли Джозеф Вайзенбаум в Масачусетському технологічному інституті (MTI) розробив чат-бота під назвою Eliza. Він використовував прості правила розмови і перефразував більшу частину того, що користувачі сказали, щоб

імітувати роджеріанского терапевта. Хоча це показало, що наївних користувачів можна обдурити, якщо вони думають, що розмовляють з фактичним терапевтом, сама система не розуміла проблеми користувача. Після цього в 1991 році була присуджена премія Лобнера, яка закликала дослідників ШІ створювати чат-ботів, які зможуть пройти випробування Тьюринга і допомогти в просуванні ІІ.

Незважаючи на те що вони не пройшли тест аж до 2014 року, багато відомих чат-боти виграли призи за перемогу в інших обмежених завданнях. До них відносяться ALICE, JabberWacky, Rose і Mitsuku.

Однак в 2014 році на конкурсі Тьюринга, присвяченому 60-річчю смерті Алана Тьюринга, чат-бот під назвою Eugene Goostman, що імітує 13-річну дитину, зумів обдурити 33% суддів, тим самим виконавши тест. Мова розмітки штучного інтелекту (Artificial Intelligence Markup Language – AIML) і ChatScript були розроблені як спосіб для позначення знань і діалогового контенту для більшості цих чат-ботів. Скрипти, розроблені з використанням цих сценарних мов, можуть потім передаватися в інтерпретатори для створення діалогового поведінки. Чат-боти, розроблені для проходження тесту Тьюринга, були в основному балакучими з однією метою – перемогти тест Тьюринга. У більшості вони не розглядалися як будь-яке просування в ІІ або крок до створення корисних діалогових помічників.

З іншого боку, дослідження в галузі штучного інтелекту, зокрема в області машинного навчання і обробки природної мови, привели до появи різних діалогових інтерфейсів, таких як системи відповідей на питання, інтерфейси природної мови для баз даних і розмовні діалогові системи. На відміну від чат-ботів, побудованих для проходження тесту Тьюринга, ці системи мали дуже чіткі цілі. Системи відповіді на питання обробляли природні мовні питання і знаходили відповіді в наборах неструктурованих текстових даних.

Інтерфейси природної мови для баз даних (Natural Language Interfaces to Database Systems – NLIDBS) були інтерфейсами для великих баз даних

SQL, які інтерпретували запити до бази даних, створювані на природній мові, такому як англійська, перетворювали їх в SQL і повертали відповідь. Розмовні системи (Spoken Dialogue Systems – SDS) – це системи, які могли б підтримувати контекстні розмови з користувачами для обробки діалогових задач, таких як бронювання квитків, контроль інших систем і навчання учнів. Це були попередники сучасних чат-ботів і діалогових інтерфейсів.

У 2011 році компанія Apple випустила інтелектуального помічника під назвою Siri в складі своїх пристроїв iPhone / iPad. Siri була змодельована як персональний помічник користувача, виконуючи такі завдання, як виклик, він читав текстові повідомлення та налаштування аварійних сигналів і нагадувань. Це одне з найбільш значних подій в недавньому минулому, які перезавантажили історію діалогових інтерфейсів. У перші дні роботи користувачі Siri використовували її лише кілька разів на місяць для виконання таких завдань, як пошук в Інтернеті, відправка SMS і вчинення телефонних дзвінків. Незважаючи на новизну, Siri призначалася для роботи з набагато більшою кількістю функцій, які були додані в наступні роки. У перші дні у Siri було багато клонів і послідовників на Android і інших платформах для смартфонів.

Більшість з них було змодельовані в якості помічників і доступно у вигляді мобільних додатків.

У тому ж році (2011 р) IBM представила Watson, систему відповідей на питання, яка брала участь в ігровому шоу під назвою Jeopardy і виграла його у попередніх переможців, Бреда Раттера і Кена Дженнінгса. Це ознаменувало собою віху в історії II, оскільки Watson зміг обробляти питання природної мови і відповідати на них в режимі реального часу. З тих пір Watson був перероблений в набір інструментальних засобів з набором інструментів когнітивного обслуговування для розпізнавання природної мови, аналізу настроїв, управління діалогами і т. Д.

Дотримуючись Siri і Watson, наступне велика подія з’явилося в Microsoft в 2013 році, коли вони представили Cortana як стандартну функцію

на смартфонах під управлінням операційної системи Windows, а потім в 2015 році і на платформі Windows 10. Як і Siri, Cortana була персональним помічником, який керував такими завданнями, як настройка нагадувань, відповіді на питання і т. д.

У листопаді 2014 року Amazon запросив своїх прем'єр-членів випробувати свого персонального помічника на ім'я Alexa. Alexa була доступна на власному продукті Amazon під назвою Echo. Echo був першим у своєму роді розумним динаміком, який розміщував в ньому помічника, як «привид» в машині. Хоча він на зувати динаміком, насправді це була крихітна комп'ютер з голосом в якості єдиного інтерфейсу, на відміну від смартфонів, планшетів і персональних комп'ютерів. Користувачі могли говорити з Alexa, використовуючи голос, просити її виконувати такі завдання, як установка нагадувань, відтворення музики і т. д.

У квітні 2016 року соціальна мережа Facebook оголосила, що відкриває свою популярну платформу Messenger для чат-ботів. Це був радикально інший підхід до діалоговим інтерфейсів, в порівнянні з Siri, Alexa та Cortana. На відміну від цих персональних помічників, оголошення Facebook призвело до створення призначених для користувача і фірмових чат-ботів. Ці боти дуже схожі на Siri, Cortana і Alexa, але можуть бути налаштовані на потреби їх створює бізнесу. Тепер чат-боти могли порушити роботу кількох ринків, включаючи обслуговування клієнтів, продажу, маркетинг, технічну підтримку і т. Д. Багато платформи обміну повідомленнями, такі як Skype, Telegram і інші, також стали доступні для інтеграції в чат-боти в той же час.

У травні 2016 року Google анонсувала Assistant, свою версію персонального чат-бота, доступну на декількох платформах, таких як додаток Allo і Google Home (розумний динамік, такий як Echo). Всі помічники, такі як Siri, Cortana, Alexa і Google Assistant, також відкрилися в якості каналів для діалогових можливостей сторонніх розробників. Таким чином, тепер можна персоналізувати програмне забезпечення Alexa і Google Assistant, додавши діалогові можливості (звані навичками або діями) з бібліотеки рішень

сторонніх розробників.

Так само, як бренди можуть створювати свої власні чат-ботів для різних служб обміну повідомленнями (наприклад, Skype і Facebook Messenger), вони можуть розвивати навички для Alexa або дії для Асистента Google. У 2018 році був випущений смарт динамік компанії Apple, Homepod, що працює під управлінням голосової платформи Siri.

Паралельно з цими подіями також спостерігається значне зростання з точки зору інструментів, доступних для створення і розміщення чат-ботів. В останні два роки відзначений експоненціальне зростання інструментів для розробки, моделювання, складання, розгортання, управління і монетизації чатів. Це призвело до створення екосистеми, яка розробляє і створює для користувача діалогові інтерфейси для підприємств, благодійних організацій, урядових та інших організацій по всьому світу.

Архітектура діалогового призначеного для користувача інтерфейсу

У цьому розділі ми розглянемо базову архітектуру діалогового інтерфейсу (рисунок 1.1.).

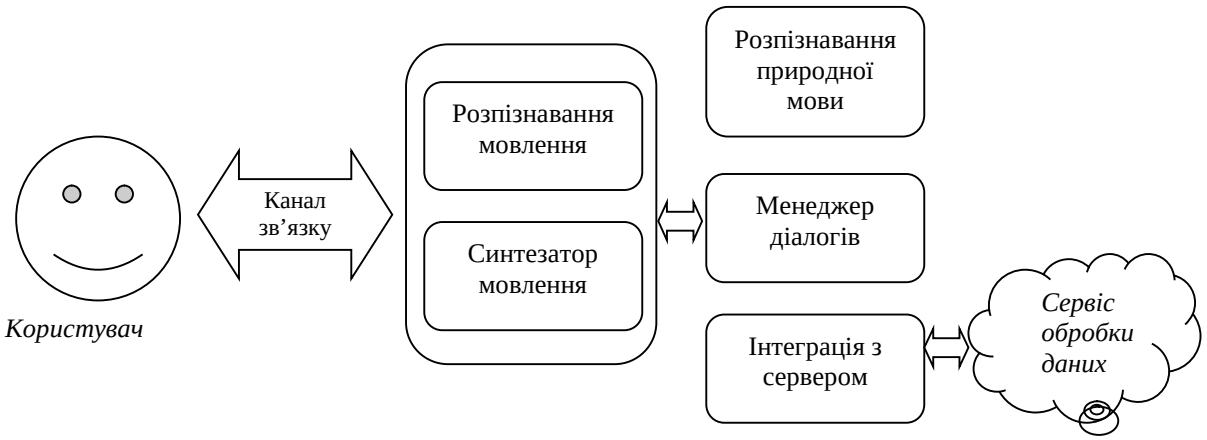

Рисунок 1.1 – Базова архітектура діалогового інтерфейсу

Основним модулем діалогового інтерфейсу є менеджер діалогів. Цей модуль контролює потік розмови. Він бере семантичне уявлення того, що

користувач говорить, як введення і вирішує, яким має бути відповідь системи. Він підтримує уявлення діалогового контексту в тій чи іншій формі, наприклад, набір пар значень ключа для виконання змістовної розмови протягом декількох обмінів між користувачем і системою.

Семантичне уявлення призначеного для користувача введення можливо безпосередньо після натиснення кнопки. У системах, які розпізнають мову, призначені для користувача висловлювання переводяться в семантичне уявлення, що складається з призначених для користувача цілей і параметрів (слотів і сутностей), за допомогою модуля розпізнавання природної мови. Можливо, цей модуль необхідно попередньо навчити, щоб розпізнати набір призначених для користувача цілей, визначених розробником, що відносяться до доступним діалоговим завданням.

Інтерфейси з підтримкою голосу, які приймають голосові інструкції користувача, також потребують модулі розпізнавання мови, який може транслювати промову в текст, перш ніж передавати його в модуль розпізнавання природної мови.

Симетрично, з іншого боку, необхідний модуль синтезу мови (або модуль перетворення тексту в мову), який перетворює текстову відповідь системи в мова.

Менеджер діалогів повинен взаємодіяти з серверними модулями. Це може бути база даних або онлайнове джерело даних, що запитується для відповіді на питання користувача (наприклад, телевізійне розклад), або онлайн-служба для виконання інструкцій користувача (наприклад, бронювання квитка).

Канал – це місце, де чат-бот очікує користувача. Залежно від каналу може бути один або кілька модулів, з яких складається цей шар. Наприклад, якщо чат-бот знаходиться в Facebook Messenger, цей шар складається з Facebook-сторінки і Facebook-додатку, який з’єднується з іншими модулями чат-бота, оформленими у вигляді веб-додатку.

1.5 Класифікація чат-ботів

Діалогові призначені для користувача інтерфейси застосовуються в різних сценаріях. Їх програми можна розділити на дві категорії: корпоративні помічники і персональні помічники.

Корпоративні помічники – це чат-боти та інші діалогові призначені для користувача інтерфейси, змодельовані після опитування представників служби підтримки клієнтів і помічників магазину. Як і представники обслуговування клієнтів, боти залучають клієнтів до розмови, виконуючи маркетингові, продажні і допоміжні завдання. Більшість чат-ботів, розгорнутих по каналах, таким як Facebook Messenger, Skype, Slack і багато інших, є корпоративними помічниками. Вони спроектовані і побудовані для виконання завдань, які будуть виконувати помічники магазину і представники з обслуговування клієнтів.

Корпоративні помічники розробляються в багатьох секторах бізнесу, автоматизуючи різні діалогові завдання. З іншого боку, персональні помічники, наприклад Alexa, Siri і Cortana, які грають роль особистого помічника користувача, виконують такі завдання, як управління календарем, відправка текстів, прийом викликів і відтворення музики. Можливості цих особистих помічників можуть бути розширені. Наприклад, Alexa допускає таке розширення, дозволяючи розробникам створювати навички, які користувачі можуть вибрати для додавання в свою власну Alexa. Таким чином, бренди можуть розвивати навички Alexa або дії для Асистента Google, які дозволять Alexa і Асистенту взаємодіяти з IT-послугами бренду і виконувати такі завдання, як розміщення замовлень, перевірка статусу доставки і багато іншого.

Хоча чат-боти розроблялися протягом, принаймні, кількох десятиліть, вони не стали основними каналами для взаємодії з клієнтами до недавнього часу. За останні два роки завдяки серйозним зусиллям таких гігантів індустрії, як Apple, Google, Microsoft, Facebook, IBM і Amazon, а також їх

подальшим інвестиціям в розробку наборів інструментів, чат-ботів і діалогових інтерфейсів вони стали серйозним суперником інших каналів зв’язку з клієнтами. За цей час чат-боти були застосовані в різних секторах і в різних діалогових сценаріях в цих секторах: роздрібному, банківському та фінансовому, урядовому, медичному, юридичному і незалежному секторах і багато іншого.

У роздрібній торгівлі чат-боти застосовуються для діалогів з приводу маркетингу продукту, залучення бренду, підтримки продукту, продажів і підтримки. Браузерні чат-боти пропонують поради і рекомендації лояльним клієнтам бренду, пов’язані з використанням продуктів, що продаються брендом. Наприклад, чат-бот Sephora радить користувачам підібрати ідеальну помаду. Точно так же чат-бот TK-Maxx допомагав користувачам вибирати подарунки для своїх друзів і сім’ї під час Різдва 2016 року. Одним з перших роздрібних продавців для вивчення чатів для продажів був H & M. Чат-бот H & M допоміг користувачам переглянути каталог продуктів і додати продукти в свої кошики. Виробники автомобілів, такі як Tesla, Kia і Mercedes, розробили чат-ботів, які допомагають водіям отримати інформацію про своїх автомобілях.

Чат-боти дуже ефективні в банківській і фінансовій індустрії. Банкінг був одним з перших секторів, які експериментували з діалоговими інтерфейсами. Банківські чат-боти відповідають на загальні питання про фінансові продукти, безпечному банківському обслуговуванні і т. Д., А також надають конкретну і персоналізовану інформацію по рахунках користувачів. Багато найбільші банки і постачальники фінансових послуг, включаючи Bank of America, ICICI bank, HSBC, Royal Bank of Scotland, Capital One, MasterCard і т. і., Розгорнули чат-боти, щоб допомогти своїм клієнтам. Багато фінансові інститути створюють чат-ботів, які виступають в якості фінансового помічника для користувачів. Ernest.ai і Cleo – це чат-боти, які з’єднують вас з вашими банківськими рахунками і говорять про ваші витрати, балансах, а також дають поради щодо економії грошей.

Чат-боти застосовуються в юридичному, медичному, урядовому та незалежному секторах. Чат-бот під назвою DoNotPay допоміг людям оскаржити паркувальні квитки в Лондоні і Нью-Йорку в більш ніж 160 000 випадків. Після цього було розроблено багато нових чат-ботів, що дозволяють допомогти людям отримати доступ до судових та юридичних послуг: оцінка злочинності (LawBot), реєстрація бізнесу (LawDroid), допомога орендарям (RentersUnion), допомога в юридичних питаннях і документації (Lisa, LegaliBot, Lexi, DocuBot), пошук адвокатів (BillyBot).

У незалежному секторі чат-боти використовувалися для поширення інформації про проблеми благодійності. Stoptober1 – це чат-бот Facebook, який був розроблений Національними службами охорони здоров'я (NHS) у Великобританії, щоб допомогти курцям кинути палити. Інший чат-бот, Yeshi2, був розроблений, щоб привернути увагу до водної кризи в Ефіопії. Чат-боти починають вносити свій вклад в охорону здоров'я. Чат-боти Your.MD і HealthTap були розроблені для діагностики проблем зі здоров'ям на основі симптомів. Emily3 – це чат-бот, розроблений LifeFolder, щоб допомогти прийняти рішення після смерті (наприклад, юридична документація, підтримка життєзабезпечення, донорство органів і багато іншого).

Чат-боти використовуються не тільки для того, щоб допомогти клієнтам, але і всередині, для співробітників. Чат-боти в деякому сенсі стають колегами, допомагаючи співробітникам в роботі з повторюваними, рутинними і нудними завданнями. Служби обміну повідомленнями, такі як Slack і Microsoft Teams, заохочують чат-ботів на своїх платформах для автоматизації службового зв'язку. Ці боти націлені на залучення співробітників в чат для вирішення цікавих і важливих завдань. Наприклад, є боти, здатні допомогти колегам поділитися знаннями (Obie.ai), отримати доступ до інших служб, таким як GDrive (WorkBot), налаштувати зустрічі (Meekan), обговорити обід (LunchTrain) і навіть допомогти в прийнятті рішень (ConcludeBot, SimplePoll).

1.6 Інструменти розробника чат-ботів

За останні кілька років екосистема інструментів і послуг навколо ідеї діалогових інтерфейсів зросла. Існує безліч інструментів, які ми можемо використовувати для дизайну, розробки та управління чат-ботами.

Інструменти прототипирования використовуються, щоб показати клієнтам, як буде виглядати і вести себе чат-бот. Це інструменти, які ви можете використовувати під час конструювання діалогу, після того як запишете ідею і приклади діалогів між користувачем і ботом на зворотному боці серветки. Інструменти прототипирования дозволяють візуалізувати бесіду між користувачем і ботом і продемонструвати динаміку розмови в обох напрямках.

BotSociety.io (botsociety.io) і BotMock.com (botmock.com) – приклади популярних інструментів прототипирования. Деякі з цих інструментів дозволяють експортувати прототип і створювати відео.

Канали – це місця, де користувачі можуть взаємодіяти з чат-ботом. Існує кілька каналів розгортання, за якими ваші боти можуть бути доступні користувачам. До них відносяться служби обміну повідомленнями, такі як Facebook Messenger, Skype, Kik, Telegram, WeChat і Line; офісні та командні чати, такі як Slack, Microsoft Teams і багато інших; традиційні канали, такі як веб-чат, SMS і голосові дзвінки; і розумні динаміки, такі як Amazon Echo і Google Home. Виберіть канал, базуючись на ваших користувачів і вимогах проекту. Наприклад, якщо ви створюєте чат-бота для споживачів, Facebook Messenger може бути кращим вибором через зростаючого числа користувачів, які застосовують послугу, щоб підтримувати зв’язок з друзями та родиною. Додати вашого чат-бота в їх список контактів може бути простіше, ніж змусити їх завантажити ваш додаток. Якщо користувачеві необхідно взаємодіяти з ботом голосом в домашній або офісному середовищі, канали розумних динаміків можуть бути ідеальним вибором.

І нарешті, є інструменти, які можуть підключати чат-ботів до багатьох

каналах одночасно (наприклад, інтеграція Dialogflow, служба MS Bot і Smooch.io і т. Д.).

Існує безліч інструментів, які можна використовувати для створення чат-ботів без необхідності роботи з кодом безпосередньо: Chatfuel, ManyChat, Dialogflow і т. Д. Chatfuel дозволяє дизайнерам створювати потік діалогу з використанням візуальних елементів. За допомогою ManyChat ви можете вибудувати потік, використовуючи візуальну карту, яка називається FlowBuilder. Діалогові елементи, такі як висловлювання бота і кнопки відповіді користувача, можуть бути налаштовані шляхом простого перетягування елементів інтерфейсу. Dialogflow можна використовувати для створення чат-ботів, які вимагають розширених методів розпізнавання природної мови для взаємодії з користувачами.

З іншого боку, існують мови сценаріїв, такі як мова розмітки штучного інтелекту (Artificial Intelligence Markup Language – AIML), ChatScript і RiveScript, які застосовуються для створення чат-ботів. Ці сценарії будуть містити діалоговий контент і потік, які потім повинні бути відправлені програмі інтерпретатора або механізму правил, щоб оживити чат-бота.

Інтерпретатор вирішує, як продовжувати діалог, зіставляючи призначені для користувача висловлювання з шаблонами в сценаріях. Хоча за допомогою цього підходу легко створювати діалогових чат-ботів, стає складно створювати транзакційні чати, чи не генеруючи явних семантичних уявлень призначених для користувача висловлювань. PandoraBots – популярна веб-платформа для створення чат-ботів AIML.

В якості альтернативи існують бібліотеки SDK, які можна використовувати для створення чатів: MS Bot Builder, BotKit, BotFuel і т. Д. SDK надається на одному або декількох мовах програмування, щоб допомогти розробникам у створенні основного діалогового модуля управління. Можливість настройки діалогового менеджера дозволяє розробникам гнучко форматувати бесіди і інтегрувати в бота завдання резервного копіювання, без зайвого коду і сценарних платформ. Після

створення менеджер діалогів може бути підключений до інших служб, таким як розпізнавання природної мови, щоб сприймати призначені для користувача висловлювання.

2 ЗАГАЛЬНА МОДЕЛЬ БАГАТОРІВНЕВОГО ЧАТ-БОТА

2.1 Аналіз компонентів і вибір інструментів

Як і інші цифрові рішення, чат-боти можуть отримати вигоду з збору та аналізу статистики їх використання. Хоча ви можете Інструменти розробника використовувати готові інструменти, які зараз широкодоступним.

Існують багато готові інструменти для аналітики, які можна підключити до чат-боту, використовуючи вхідні і вихідні повідомлення, які можуть бути зареєстровані і вивчені. Ці інструменти повідомляють розробникам і менеджерам чат-бота про типах розмов, які фактично відбуваються між користувачами і чат-ботом. Дані нададуть корисну інформацію, таку як популярні діалогові інструкції, місця, де діалог переривається, висловлювання, які бот не пізнав, і запити, для яких як і раніше потрібно масштабування чат-ботів.

Dashbot.io, BotAnalytics і Chatbase Google – це кілька аналітичних інструментів, які ви можете використовувати для аналізу продуктивності вашого чат-бота.

Розпізнавання природної мови Чат-боти можуть бути побудовані без необхідності розпізнавати висловлювання користувача. Однак додавання можливостей розпізнавання природної мови не дуже складно. Це одне з відмінних якостей, яке відрізняє чат-ботів від їх цифрових аналогів, таких як веб-сайти і додатки з візуальними елементами. Існує безліч модулів розпізнавання природної мови, які доступні як хмарні служби. Великі IT-гравці ринку, такі як Google, Microsoft, Facebook і IBM, створили інструменти, які ви можете підключити до свого чат-боту. Google Dialogflow, Microsoft LUIS, IBM Watson, SoundHound і Facebook Wit.ai - ось деякі з NLU-інструментів (Natural Language Understanding – розпізнавання природної мови), які ви можете протестувати.

Одне із завдань бота – змусити користувачів знайти і використовувати його. Чат-боти не так популярні, як веб-сайти і мобільні додатки, тому потенційний користувач може не знати, куди перейти, щоб знайти бота. Як тільки ваш чат-бот буде розгорнуто, ви повинні допомогти користувачам знайти його. Є каталоги, в яких перераховані боти в різних категоріях. Chatbots.org – одна з 31 найстаріших служб каталогів, яка з 2008 року публікує списки чат-ботів і віртуальних помічників. Іншими популярними службами є Botlist.co, BotPages, BotFinder і ChatBottle. Ці каталоги класифікують ботів з точки зору мети, сектора, підтримуваних мов, країн і т. Д. На додаток до цих такі канали, як Facebook і Telegram, мають свої власні каталоги для ботів, розміщених на їх каналі.

Чат-боти створені для багатьох цілей: для інформування, підтримки клієнтів після продажу товарів / послуг, надання платних послуг та багатьох інших. На додаток до всього цього чат-боти з цікавим контентом можуть довгий час залучати користувачів і застосовуватися для того, щоб заробити гроші за рахунок цілеспрямованої персоналізованої реклами. Такі послуги, як CashBot.ai і AddyBot.com, можуть інтегруватися в вашого чат-бота для відправки націлених рекламних оголошень і рекомендацій користувачам, і коли користувачі будуть брати участь, ваш чат-бот буде заробляти гроші.

Вищезазначене – не вичерпний список інструментів, а також послуг. Ці інструменти розвиваються з часом, у міру того як чат-боти знаходять свою нішу на ринку. Цей список повинен дати вам уявлення про те, як багатовимірна екосистема, допомогти вам досліджувати її і дати їжу вашому розуму.

Діалогові призначені для користувача інтерфейси дають найкраще з обох світів: людиноподібна природне взаємодія в поєднанні з перевагами цифрових технологій.

Доступність: як і будь-які інші автоматизовані цифрові технології, діалогові інтерфейси недорогі і доступні цілодобово і без вихідних. Це схоже на те, що хтось постійно сидить в веб-чаті, щоб клієнти завжди мали

можливість отримати відповіді.

Персоналізований досвід: на відміну від веб-сайтів і додатків для смартфонів, чат-боти забезпечують вкрай персоналізований підхід завдяки розмовної природі взаємодії. Індивідуальні настройки бесіди надають широкі можливості для розпізнавання і адаптації до цілей, перевагам і обмеженням користувача.

Низька вартість: чат-боти – це цифрові рішення, отже, вони надають послуги підтримки клієнтів, щонайменше, в десять разів дешевше, ніж люди, що виконують ті ж самі завдання.

Стабільність: чат-боти можуть безперервно працювати в службах, чого важко досягти за допомогою операторів-людей і що може бути дуже важливо в певних секторах.

Швидкий час відгуку: на відміну від людських систем, час відповіді в чат-ботах набагато менше. Користувачам більше не потрібно чекати, поки їх дзвінок буде прийнятий, і під час розмови чат-бот буде відповідати швидше людини, особливо коли людським операторам доручено одночасно вести більш одного чату (іноді до п’яти). Здатність чат-ботів обробляти одночасні діалоги також усуває вузьке місце на обмежену пропускну спроможність підтримки користувачів і, отже, допомагає підприємствам масштабироваться.

Масштабування: чат-боти можуть легко масштабироваться, щоб обробляти зростаючий і сезонний трафік, що нелегко зробити при використанні штату людей-радників. Святковий сезон може особливо підвищити попит на підтримку клієнтів. У такі моменти чат-боти можуть використовуватися для обробки фонових і простих завдань, тим самим зменшуючи навантаження на живих консультантів, і застосовуватися розумно для обробки особливо цінних розмов.

Технології діалогового призначеного для користувача інтерфейсу в даний час є однією з головних тенденцій в технологічному бізнесі. Більшість великих брендів почали формулювати свою стратегію чат-ботів в рамках своєї більшої стратегії ІІ і автоматизації. Такі інноваційні рішення, як чат-

боти, розумні динаміки і безпілотні автомобілі, призводять до важливих політичних рішень.

Світ готується до наступу технологій автоматизації, які готові замінити людей в повторюваних і структурованих завданнях.

Недавнє зростання популярності чат-ботів був викликаний багатьма факторами: представники покоління Y неухильно рухалися до чатах як свого пріоритетним носієм для взаємодії з брендами. Опитування клієнтів показують, що люди частіше хочуть використовувати канали веб-чату, коли вони доступні, в порівнянні з іншими традиційними каналами, такими як електронна пошта і телефон, для зв'язку з підприємствами;

- зростання додатків обміну повідомленнями в чатах на смартфонах і інших пристроях перевершив використання додатків для соціальних мереж, таких як Facebook і Twitter. Тепер люди витрачають більше часу на додатки для обміну повідомленнями, розмовляють з друзями, родиною, колегами і навіть з співробітниками компаній;

- зростаючий купівельний попит на чати чинить величезний тиск на бренди. Відсутність достатніх кваліфікованих людських ресурсів для обробки зростаючого трафіку чатів також є важливим фактором зростання популярності чат-ботів;

- наявність інструментів віртуального обслуговування для розпізнавання природної мови, розпізнавання мови, синтезу мови, управління діалогами, аналітики і т. д. зробило дизайн і розвиток чат-ботів простіше, ніж це було кілька років тому;

- відкриття каналів обміну повідомленнями та інноваційні нові можливості, такі як інтелектуальні динаміки, зробили доставку послуг через чат-боти реальністю. Той факт, що зростає інтерес до додатків і пристроїв обміну повідомленнями, таким як розумні динаміки, дає привабливу можливість для брендів створювати чат-боти користувачам, які вже використовують ці канали.

Існує кілька оглядів і статистичні дані, які демонструють, що діалогові

інтерфейси нікуди не подінуться.

У списку нижче перераховані деякі з найбільш переконливих результатів опитувань і прогнозів, які показують, що у чат-ботів довгострокова перспектива:

- компанія Gartner прогнозує, що до 2021 року бренди, які розробляють свої веб-сайти з урахуванням голосового і візуального пошуку, підвищать свій дохід на 30% і що більш 50% компаній будуть витрачати більше на чат-боти, ніж на традиційні мобільні додатка 1;

- в опитуванні Oracle 80% респондентів (керівники рівня С) заявили, що планують впровадити чат-ботів до 2020 року, якщо вони ще не були введені;

- компанія Juniper прогнозує, що використання чат-ботів призведе до щорічної економії витрат в 8 млрд доларів США до 2022 року, в порівнянні з 20 мільйонами доларів США в 2017 году2;

- опитування Hubspot показав, що близько 47% споживачів готові робити покупки предметів через чат-бот, а близько 40% не хвилює, чи розмовляють вони з чат-ботом або людиною, поки вони отримують допомогу легко і бистро3;

- нарешті, згідно з недавнім звітом Grand View Research, глобальний ринок чат-ботів досягне приголомшливих 1,25 млрд дол. США до 2025 року, збільшивши сукупний середньорічний темп зростання на 24,3% з 190 млн дол. США в 2016 році.

2.2 Формалізація та структура чат-ботів

Чат-бот (віртуальний співрозмовник або просто бот) – це комп’ютерна програма, яка, імітуючи живого співрозмовника, розмовляє з користувачем-людиною (найчастіше через Інтернет). Стрімке поширення інтелектуальних роботів обумовлено поєднанням декількох факторів. По-перше, користувачі вже втомилися завантажувати і встановлювати додатки. Це явище перевтоми

від додатків. По-друге, люди масово переходять на служби повідомлень і постійно знаходяться в чатах. Таким чином, замість чергового додатка можна створити сервіс, який працює в додатку, яке вже встановлено на пристрої користувача. Чат-боти стають затребуваним і бажаним варіантом взаємодії завдяки досягненням у сфері штучного інтелекту. Створені потужні алгоритми машинного навчання,

Штучний інтелект (ШІ) – основа чат-ботів. Поняття штучного інтелекту (ІІ) існує досить давно. Навіть в грецьких міфах згадуються механічні люди, здатні імітувати поведінку живої людини. Ранні європейські комп'ютери були задумані як «логічні машини». Прагнучи відтворити такі функції, як пам'ять і основні арифметичні операції, інженери того часу по суті намагалися створити механічний мозок. Поняття «штучний інтелект» має більш широке охоплення і має на увазі здатність машин вирішувати завдання розумним, з нашої точки зору, спосіб.

У міру розвитку технологій і еволюції наших поглядів на пристрій людського розуму змінювався і зміст поняття «штучний інтелект». Найбільш значущі сучасні досягнення в цій галузі стали можливі завдяки вдосконаленню технологій машинного навчання (клас методів штучного інтелекту, за рахунок яких машина в теорії здатна вирішити будь-яке завдання).

Машинне навчання має на увазі навчання комп'ютера на основі аналізу даних і подальше побудова відповідних прогнозів при обробці нових (не зустрічалися раніше) даних. Артур Семюел (Arthur Samuel), один з основоположників науки про штучний інтелект, запропонував термін «машинне навчання» в 1959 р і визначив його як галузь науки, яка присвячена наділення комп'ютерів здатністю навчатися без явного програмування.

В основі машинного навчання лежать складні статистико-математичні алгоритми розпізнавання. Методи машинного навчання вже зараз широко застосовуються і поступово проникають в нові сфери діяльності. Приклади: безпілотні автомобілі, розпізнавання мови, розпізнавання рукописного тексту,

розпізнавання осіб, оптичне розпізнавання символів, виявлення спаму, сегментація ринку, прогнозування, астрономія (створення теорій виникнення Всесвіту), виявлення шахрайства, прогнозування відмов реактивних двигунів. Серед технологій машинного навчання особливе місце займають сучасні нейронні мережі, які стали найважливішим технічним засобом для вирішення широкого кола завдань.

Штучні нейронні мережі імітують людський розум і здатність мозку до зміни зв’язків між нейронами для придбання нових навичок. Моделюючи інтелект, цілком логічно спробувати зімітувати людський мозок з його дивовижною здатністю до навчання. Крім того, існує теорія, згідно з якою наш мозок функціонує за єдиним алгоритмом і набуває різні навички в залежності від вхідних сенсорних даних. Експерименти показали, що ми можемо натренувати слухову зону кори головного мозку на здатність бачити, і так само з зоною кори, що відповідає за відтворення мови.

Архітектури машинного навчання на основі нейронних мереж влаштовані подібно біологічному мозку, тобто мають форму багатошарових мереж, що складаються з нейронів, дендритів, аксонів і синапсів. Як і людський мозок, штучна нейронна мережа дуже пластична. Вона може вирішувати завдання машинного навчання, раніше представляли величезну складність. Нейронні мережі застосовуються при вирішенні складних нелінійних задач машинного навчання в таких напрямках, як комп’ютерний зір, безпілотні автомобілі, розпізнавання емоцій за голосом і виразом обличчя, розпізнавання мови, розпізнавання осіб і ін.

Штучний інтелект визначається сукупністю різних інтелектуальних сервісів, які показані на наступній схемі (рисунок 2.1).

Нижче описані три важливих прориву, які зумовили розвиток технологій машинного навчання і стимулювали інтерес до нейронних мереж в 1980-1990-х рр., Створивши умови для удосконалення методів штучного інтелекту.

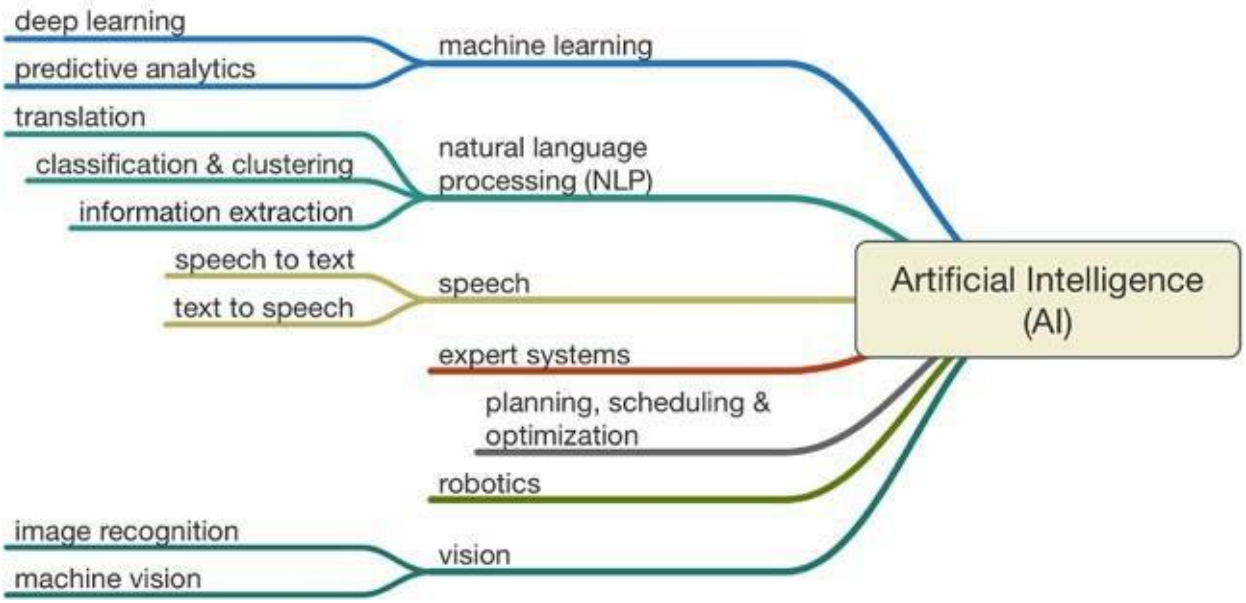

Рисунок 2.1 – Інтелектуальні сервіси

Доступність даних. З'явився Інтернет. Обсяг що генеруються, збережених і доступних для аналізу цифрових даних кардинально збільшився. За оцінками IBM, 90% світового обсягу даних створені в останні два роки. Дані являють собою їжу для статистичного аналізу, який є ключовим засобом підвищення точності в моделях машинного навчання.

Великомасштабні обчислення. Складні алгоритми машинного навчання і багаторівневі нейронні мережі споживають величезний обсяг обчислювальних потужностей. Новітні обчислювальні методи, включаючи модель MapReduce, винайдену фахівцями Google Джеффом Діном (Jeff Dean) і Санджаєм Гемаватом (Sanjay Ghemawat) в 2004 р, забезпечують великомасштабну параллелизация обчислень, необхідну для навчання багаторівневих нейронних мереж.

Закон Мура та спеціалізовані II-чіпи. Безперервний експоненціальне зростання обчислювальної потужності в поєднанні з методами прискорення обчислень за допомогою графічних процесорів також створив сприятливі умови для практичного застосування складних алгоритмів машинного навчання. В останні роки великі компанії, включаючи Nvidia, Intel і AMD,

працюють над створенням мікропроцесорів з вбудованими графічними процесорами, які були б пристосовані до завдань машинного навчання. У поєднанні з великомасштабними високопродуктивними технічними інструментами математичних обчислень ці нові чіпи допоможуть значно скоротити нинішні багаторічні інтервали між ітераціями навчання і створити неможливі раніше алгоритми і нейронні мережі з високою інтенсивністю обчислень.

Додатки машинного навчання здатні прочитати текст і розпізнати намір його автора, т. Е. Зрозуміти, наприклад, що автор тексту пред’являє претензію або вітає адресата. Подібна система зуміє прослухати музичну композицію, розпізнати її емоційне забарвлення (наприклад, сумна або весела) і підібрати інші композиції під той же настрій. У деяких випадках така система здатна самостійно скласти музику, яка, за припущенням системи, буде приємна тим, кому сподобалася вихідна композиція.

Вище ми розглянули приклади можливостей систем машинного навчання і нейронних мереж. Однак існує й інша задача, в формулюванні якої не останню роль зіграла наукова фантастика, – розробити засоби, за допомогою яких будь-яка людина зможе взаємодіяти з електронними пристроями і цифровою інформацією так само природно, як і з іншими людьми. В останні роки для вирішення цього завдання розроблені дивовижні інноваційні методи обробки природної мови (natural language processing, NLP) – однієї з областей штучного інтелекту, яка багато в чому спирається на машинне навчання.

NLP додатки аналізують природну людську мову (письмову або усну) і видають результати, користуючись тим же природним мовою. Тут машинне навчання використовується для розпізнавання величезної кількості відтінків людської мови і дозволяє навчити машину реагувати зрозумілим для аудиторії способом.

Один з перших сценаріїв використання обробки природної мови відкрив дорогу для чат-ботів – нового покоління систем розмовного взаємодії з комп'ютерами.

Oracle Intelligent Bots

Oracle розробив мобільний сервіс, який є невід'ємною частиною Oracle Cloud і забезпечив чудові результати для наших замовників в усьому світі і у всіх галузях. З огляду на швидкість поширення, успіх серед замовників з усього світу, функціональні можливості продукту, бачення і стратегію, експерти Gartner і Forrester назвали Oracle Mobile Cloud Service (MCS) лідером ринку. Лідируючи в сегменті послуг класу «платформа як послуга» (platform as a service, PaaS), ми прагнемо надати нашим замовникам кошти для створення зручних механізмів цифрового взаємодії з внутрішніми і зовнішніми замовниками. Ми створили MCS, щоб спростити розгортання мобільних додатків на смартфонах, планшетах, носяться пристроях і веб-сторінках.

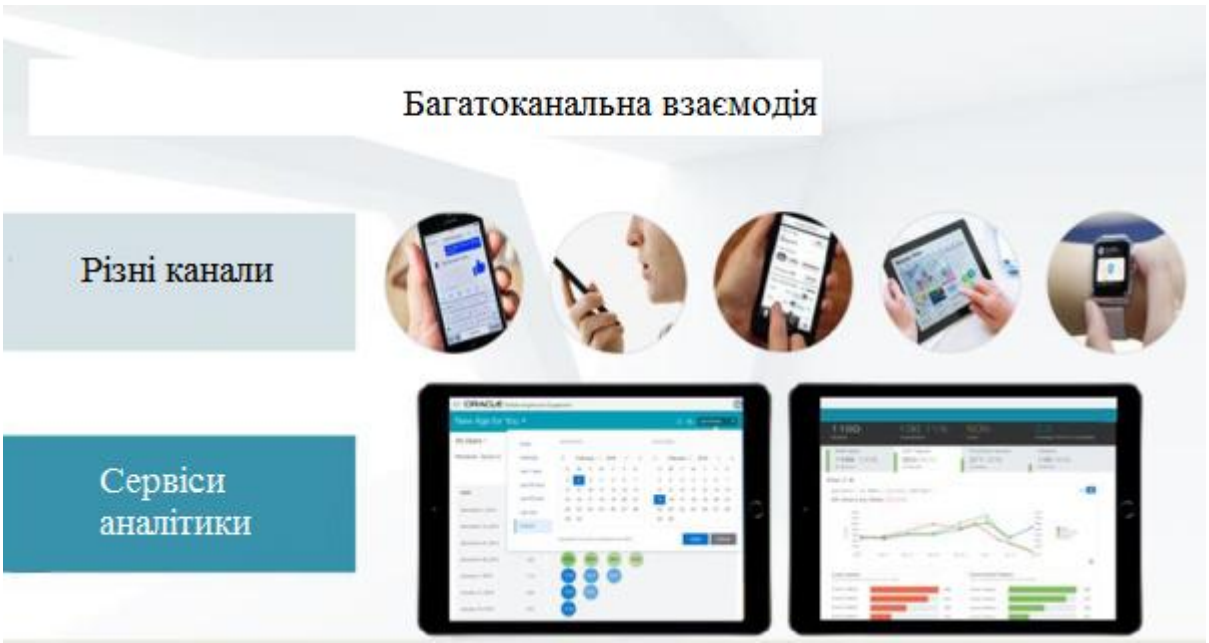

Рисунок 2.2 – Взаємодія через різні канали та аналітика

Тепер платформа Oracle Mobile Cloud Enterprise (OMCe) включає рішення Oracle Intelligent Bots, яке функціонує на основі алгоритмів машинного навчання і допомагає нашим замовникам взаємодіяти зі споживачами і співробітниками. У цьому рішенні застосовується комплекс просунутих алгоритмів машинного навчання для обробки природної мови користувачів і миттєвого формування очікуваних реакцій як при пошуку інформації (наприклад, користувач запитує інформацію про банківський рахунок), так і при здійсненні транзакцій (наприклад, користувач бажає замовити товар). Завдяки потужній технології штучного інтелекту наші інтелектуальні боти самовдосконалюються в міру збільшення обсягу оброблених розмов. Oracle Intelligence Bots дозволяє розпізнати тональність / емоції користувача,

Крім того, платформа OMCe також містить потужний засіб Customer Experience Analytics (CxA) для поглибленого аналізу взаємодії з користувачами через різні канали з метою подальшої персоналізації взаємодії з охопленням веб-сайтів, мобільних пристроїв і спамерських пошукових роботів.

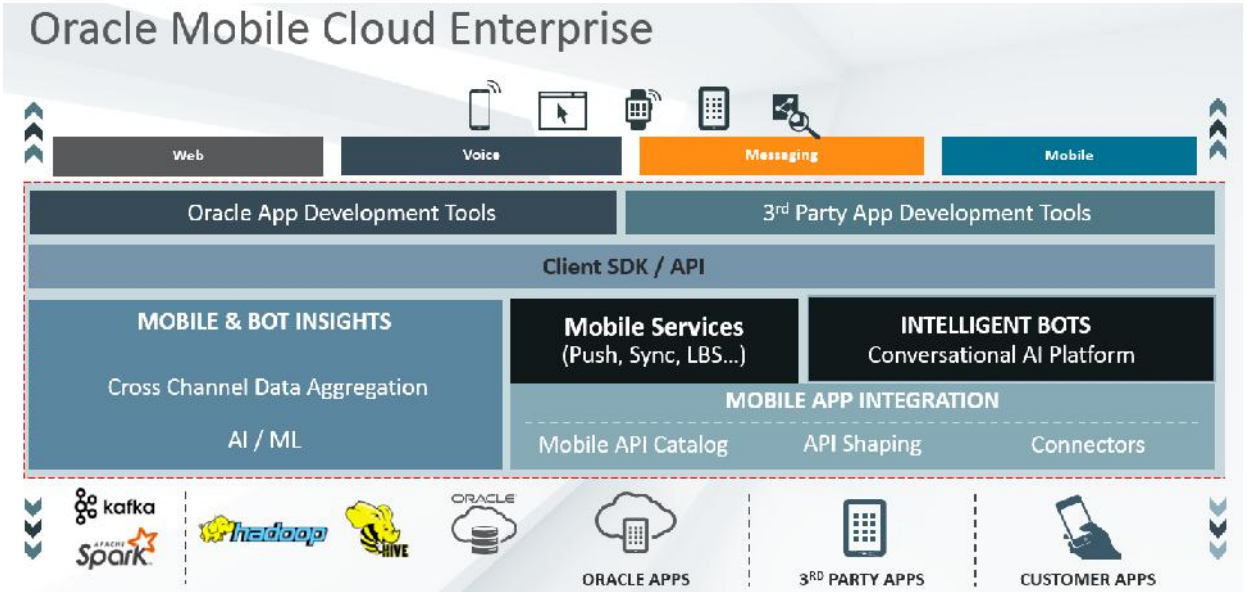

Рисунок 2.3 – Єдина контекстна платформа, що охоплює мобільні пристрої, веб-сайти і чат-боти

Oracle пропонує унікальне комплексне рішення в хмарі, яке помітно спрощує і прискорює впровадження чат-ботів для взаємодії зі споживачами та співробітниками. Як і у випадку з веб-сайтами та мобільними технологіями, механізми взаємодії зі споживачами досить швидко поширюються і на взаємодію з співробітниками. Ця тенденція була врахована при розробці Oracle Intelligent Bots – передбачена підтримка як зовнішніх (B2C), так і внутрішніх (B2E) сценаріїв використання в різних галузях і напрямках.

Мобільні технології помітно вплинули на життя і роботу кожного, торкнувшись абсолютно всі сфери діяльності. Сервіс Oracle Mobile Cloud Service застосовується в багатьох галузях, включаючи роздрібну торгівлю, фінансові послуги, ЖКГ, виробництво, автомобільну промисловість і будівництво. У Oracle прогнозують аналогічну динаміку поширення чат-ботів для сценаріїв використання B2B, B2E і B2C в більшості галузей. Уже зараз чат-боти для сценарію використання B2C впроваджуються в таких галузях, як банківські послуги, туристичні послуги, роздрібна торгівля, комунальні послуги та ресторанно-готельний бізнес, при цьому чат-боти використовуються для ведення діалогів як при здійсненні трансакцій, так і під час обговорення послуг. Крім того, є приклади впровадження чат-ботів для HR-підрозділів і відділів продажів (впровадження в CRM-системи для вирішення таких завдань, як твердження угод, а також облік робочого часу і відпусток в ERP-системах). Одне з головних переваг рішення на основі чат-ботів полягає в тому, що будь-який, хто вміє відправляти текстові повідомлення або користуватися Facebook Messenger / WhatsApp / Slack, зможе легко взаємодіяти з ботами Oracle Intelligent Bots – нічого нового вивчати не доведеться.

Oracle Intelligent Bots: ключові компоненти.

Рішення Oracle Intelligent Bots складається з 4 основних компонентів.

Рисунок 2.4 – Компоненти Oracle Intelligent Bots

Конфігуратор каналів

Кінцеві користувачі вважають за краще різні канали обміну повідомленнями. Популярність цих каналів в різних регіонах неоднакова. Існують різні категорії таких каналів. У першому наближенні можна використовувати таку класифікацію.

OTT-канали: OTT (over the top) – найпопулярніші канали обміну текстовими повідомленнями, наприклад Facebook Messenger, WhatsApp, WeChat, Line, Kik, Telegram, Talk, Skype, Slack, SMS.

VPA: Віртуальні персональні помічники (Virtual Private Assistant, VPA), наприклад Google Home, Apple HomePod і пристрої Echo, Dot і Show від Amazon.

Розширення для мобільних і веб-додатків: Додавання функцій чату в нативні, гібридні / адаптивні або веб-додатки.

Голосовий введення: Спеціалізовані пристрої або програми з інтерфейсами взаємодії з Siri, Cortana, Google Voice або іншими засобами мовного взаємодії.

Підтримка такого широкого кола каналів являє собою досить складну задачу через істотних відмінностей у функціональності цих каналів і відповідної інфраструктури – управління чергами, маршрутизація, обмеження на повідомлення, механізми резервування, кількість повторів, доступність, обробка помилок і виключень. Як і у випадку з мобільними додатками, одним з найважливіших критеріїв, з точки зору кінцевого користувача, є зручність, проте призначені для користувача інтерфейси різних текстових каналів неоднакові. І відповідальність за компенсацію цих відмінностей лягає на розробників. Рішення Oracle Intelligent Bots істотно полегшує завдання, надаючи універсальні готові інтеграційні механізми для каналів, т. Е. Компенсує різницю. Організації отримують доступ до більш широкої аудиторії і зможуть зосередити зусилля розробників безпосередньо на розмовному взаємодії, а не на складнощі інтеграції з потрібними каналами. Крім прискорення розгортання, такий механізм забезпечує і інше важлива перевага – зниження вартості володіння (TCO), оскільки дозволяє легко адаптуватися до змін. Сьогоднішній лідер може в будь-який момент поступитися місцем перспективному новачку.

2.3 Розробка сценарію діалогу

Користувачі взаємодіють з платформою Oracle Intelligent Bots за допомогою діалогу. Цей механізм взаємодії, що іменується також розмовною призначеним для користувача інтерфейсом, має на увазі звичайний людський діалог між кінцевим користувачем і чат-ботом. Наприклад, кінцевий користувач говорить: «Привіт!», На що чат-бот відповідає: «Здрастуйте!» І запитує, чим може допомогти користувачеві. Користувач може запросити перевести гроші або іншу транзакцію за допомогою банківського чат-бота, дізнатися у чат-бота відділу кадрів суму відпускних або задати питання чат-боту роздрібної мережі про те, як повернути товар. Таким чином, чат-бот

приймає на себе функції першої лінії підтримки, надаючи миттєві відповіді на питання в режимі 24/7.

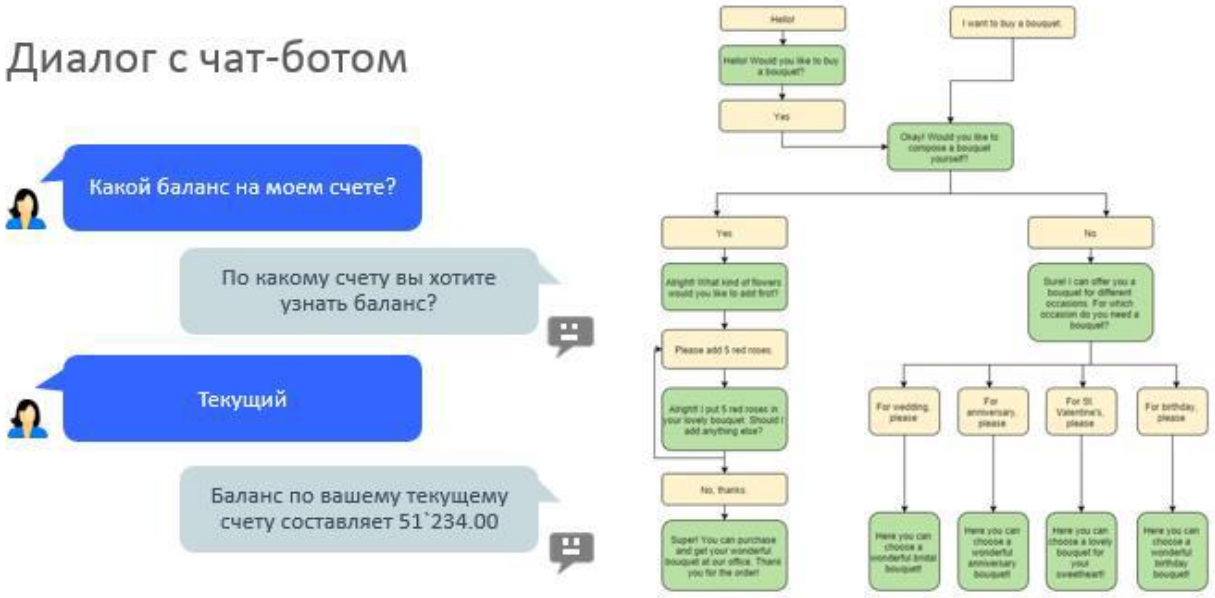

Рисунок 2.5 – Діалог з інтелектуальним ботом і його сценарій

Редактор сценарію діалогу Oracle Intelligent Bots і модуль виконання спрощують моделювання сценаріїв розмови і одночасно забезпечують високий рівень контролю над його логічними шляхами. Сценарій діалогу має кілька складових, включаючи: кроки або стану, рішення, наміри, сутності, змінні.

Розмова з чат-ботом йде за певним сценарієм, що залежить від різних станів і контексту. Цей сценарій визначає таку обставину в залежності від вхідної інформації і внутрішніх правил та реалізується за допомогою алгоритму кінцевого автомата, який можна розглядати як робочий процес. Під кроками маються на увазі стану робочого процесу, через які система проводить кінцевого користувача в процесі розмови. Така система переходу до певних станів називається кінцевим автоматом. Щоб визначити сценарій діалогу для чат-бота, необхідно задати конкретні стану і призначити дії (компоненти) для виконання відповідної роботи в цих станах. Ці кроки або

стану слідують різним логічним напрямками в залежності від введеної користувачем інформації,

Наміри і сутності представляють собою конфігуруються користувачем компоненти алгоритму обробки природної мови на основі машинного навчання. Цей алгоритм використовується для управління станами або кроками в складі сценарію діалогу чат-бота. Змінні служать для зберігання значень, які можуть змінюватися в залежності від вхідної інформації чи інших умов.

Схема процесу чат-бота Oracle Intelligent Bots є сценарій діалогу, оскільки чат-боти призначені для взаємодії з користувачами за допомогою ведення діалогу. Сам по собі процес реалізується через алгоритм кінцевого автомата з використанням набору правил і визначається в консолі платформи Oracle Intelligent Bots на мові розмітки BotsML (Oracle Intelligent Bots Markup Language). Мова BotsML є варіацією YAML. YAML - це мова розмітки, призначений для представлення даних і взаємозв’язків між ними в максимально зрозумілому для людини форматі.

Одним з вигідних переваг цього механізму проектування та виконання діалогів є тісна інтеграція з механізмом обробки природної мови на основі машинного навчання. Розмова з живою людиною часто носить нелінійний характер. В ході розмови кінцевий користувач може легко перевести діалог в інший стан або змінити контекст. Припустимо, користувач має намір перевести кому-то гроші зі свого рахунку А. Користувач звертається до чат-боту з наступним запитом: «Заплатити Тому за вечерю». Чат-бот відповідає: «З якого рахунку?» Припустимо, користувач вибирає чековий рахунок, але розуміє, що не пам’ятає, скільки на цьому рахунку коштів. Користувач змінює контекст, запитуючи баланс рахунку, історію транзакцій за останній час і т. Д. Інакше кажучи, користувач змінює стан, переходячи від переказу коштів до перевірки балансу і потім до перегляду історії транзакцій по рахунку. У якийсь момент користувач приймає рішення провести грошовий переказ Тому, т. Е. Повертає розмову в початковий стан. Платформа Oracle Intelligent

Bots дозволяє легко змоделювати такий сценарій за допомогою вбудованих функцій управління станами - готового рішення, яке заощадить час розробників. Це прискорює розгортання і знижує вартість володіння.

Механізм штучного інтелекту.

Засіб Oracle Intelligent Bots надає кілька моделей для розуміння природної мови (NLU). Ці моделі дозволяють розпізнати намір користувача шляхом аналізу вхідного запиту, отриманого чат-ботом, і точно реалізувати відповідний сценарій діалогу. Це досягається за допомогою декількох алгоритмів обробки природної мови та машинного навчання в поєднанні з наступними підходами до класифікації намірів користувача. Ці підходи описані нижче:

- модель Trainer Ht – висока швидкість. При малих обсягах даних застосовується модель Trainer HT, яка формує комбінацію правил за методами генеративної NLP-лінгвістики і дозволяє почати роботу дуже швидко, навчивши модель на невеликому корпусі даних. Ця модель забезпечує високу точність розпізнавання наміри користувача для вхідних запитів, схожих з фрагментами, що містяться в корпусі даних.

- Модель Trainer Tm – інтелект з узагальненням. При великих обсягах наборів даних, що вимагають більшого узагальнення для класифікації наміри користувача (тобто підвищення точності класифікації намірів при аналізі вхідних запитів, відсутніх в корпусі даних), і при наявності високоякісного вихідного набору даних, пристосованого до сценаріїв діалогів через Trainer HT, платформа розпізнає намір користувача за допомогою просунутої моделі на основі машинного навчання. У другій моделі (Trainer Tm) застосовується набір алгоритмів неконтрольованого навчання, які аналізують великі обсяги нерозміченого тексту (наприклад, The Wall Street Journal, The New York Times, Вікіпедія, Reuters і ін.) І формують контекстно залежні зіставлення або вектори розпізнавання наміри користувача на основі корпусу даних, механізм обробки природної мови налаштовується через наміри і сутності. Налаштування не вимагає навичок розробника і виконується так, щоб

механізм обробки природної мови зміг розуміти і класифікувати розмови з кінцевим користувачами, витягуючи необхідну інформацію для точного виконання відповідних дій (наприклад, проведення транзакції або вилучення даних з цільової (бекенд) системи).

У першому наближенні намір – це те, чого хоче досягти кінцевий користувач (наприклад, дізнатися залишок на рахунку, зробити покупку). Концепція наміри передбачає встановлення відповідності між вхідною інформацією, що надійшла від користувача, і одиницею роботи, яку повинні виконати цільові системи. Таким чином, по фразам, які користувач вводить в розмові з чат-ботом, необхідно визначити конкретний сценарій використання або одиницю роботи. «Перевірити баланс», «перевести гроші», «відстежити витрати» – це сценарії використання. Чат-бот повинен підтримувати всі необхідні сценарії використання і точно визначати, яку одиницю роботи необхідно виконати, виходячи з тексту, введеного кінцевим користувачем у вільній формі на природній мові. Розпізнавання наміри по тексту у вільній формі проводиться за допомогою механізму обробки природної мови. Механізм обробки природної мови функціонує на основі відповідної моделі і починає роботу з навчання на невеликому корпусі найбільш поширених фраз для кожного наміри. Наприклад, для наміри «перевірити баланс» адміністратор проводить навчання механізму, вводячи в Oracle Intelligent Bots Builder кілька простих висловлювань: «Перевірити баланс», «Скільки у мене грошей» і т. П.

Суті можна розглядати як асоційовані з намірами параметри, точно визначають або уточнюють дію, яке повинні виконати цільові системи. Суті витягуються з висловлювань користувача і передаються в якості параметрів в інтеграційний компонент, або використовуються для прийняття рішень щодо просування по сценарієм діалогу. Наприклад, при наявності декількох рахунків, намір «перевірити баланс» потрібно уточнити, тобто з’ясувати, який саме рахунок (наприклад, чековий або ощадний) цікавить користувача. Якщо користувач просить «дізнатися баланс ощадного рахунку», чат-бот витягує

сутність «ощадний» і ставить її у відповідність зі змінною AccountType (Тип рахунку). За допомогою сутностей можна визначити елементи, що вводиться, які слід вважати значеннями змінних.

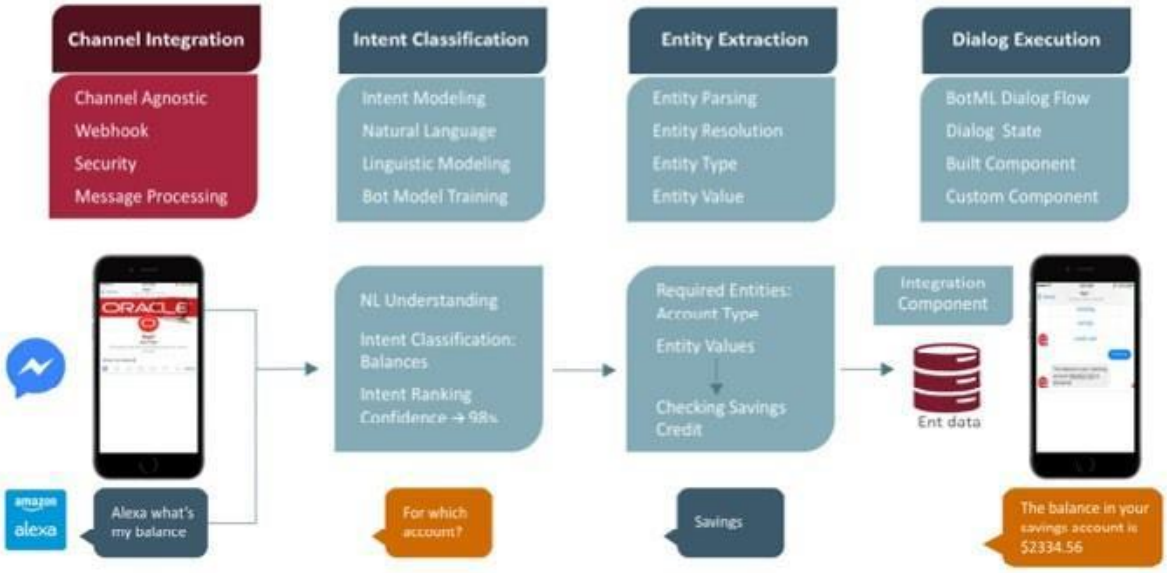

Рисунок 2.6 – Класифікація намірів і витяг сутностей - загальна картина

Цей подвійний підхід дозволяє негайно почати роботу і доповнити вихідний корпус найпотужнішою моделлю машинного навчання. Точність розпізнавання наміри буде рости в міру збільшення обсягу наборів даних. Крім алгоритмів обробки природної мови, в Oracle намір впровадити додаткові функціональні можливості штучного інтелекту. Зокрема, мова йде про такі можливості, як ведення об’ємних діалогів в форматі «питання-відповідь» і застосування засобів автозавершения, розпізнавання тональності і аналізу зображень безпосередньо в ході розмови. Таким чином, рішення Oracle вийде далеко за рамки обробки природної мови,

2.4 Спеціалізовані компоненти чат-ботів

Чат-боти приносять користь лише за умови якісної інтеграції з системами, з яких необхідно отримувати інформацію для цільових сценаріїв

використання. Як правило, це корпоративні серверні системи, де зберігаються банківські дані, інформація про співробітників (HCM-системи) або продажах (CRM-системи відділів продажів). Oracle є лідером в області інтеграції з корпоративними і хмарними джерелами даних, причому така інтеграція доступна в формі готового механізму, який заощадить сили, час і кошти на підготовку даних для чат-ботів. Завдяки вбудованим засобам інтеграції з Oracle Mobile Cloud, Enterprise (OMCe), платформа Oracle Intelligent Bots може використовувати відповідні мобільні API-інтерфейси, а також мобільні сервіси (наприклад, сервіс push-повідомлень для передачі асинхронних повідомлень з серверних систем чат-боту). Ці спеціалізовані компоненти також сумісні з будь-якими REST-сервісами і розгортаються в формі мікросервісов.

Oracle Intelligent Bots Builder

Платформа Oracle Intelligent Bots містить засіб Oracle Intelligent Bots Builder, що дозволяє будувати чат-боти в веб-браузері інструментами візуального програмування. За допомогою цих інструментів можна створити новий чат-бот, визначити наміри, сутності, схему діалогу, спеціалізовані компоненти і налаштувати інтеграцію чат-бота з потрібними каналами. Засіб Oracle Intelligent Bots Builder також забезпечено функціями тестування системи в режимі реального часу – можна вводити фрази в тестове поле і тут же отримувати відповіді від механізму обробки природної мови. Аналізуючи результати тестування, розробники можуть налаштовувати і заново навчати бот в режимі реального часу. Крім того, за допомогою засобу Intelligent Bots Builder можна тестувати процес розмовного взаємодії в повному обсязі,

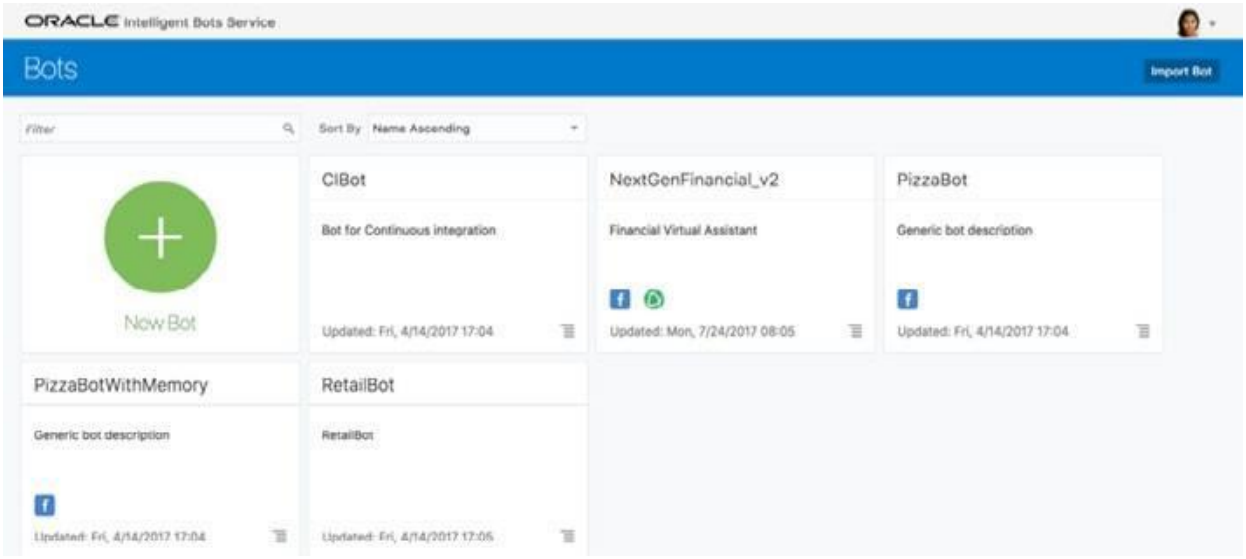

Рисунок 2.7 – Початкова сторінка Oracle Intelligent Bots - створення, клонування, імпорт і експорт існуючих інтелектуальних роботів

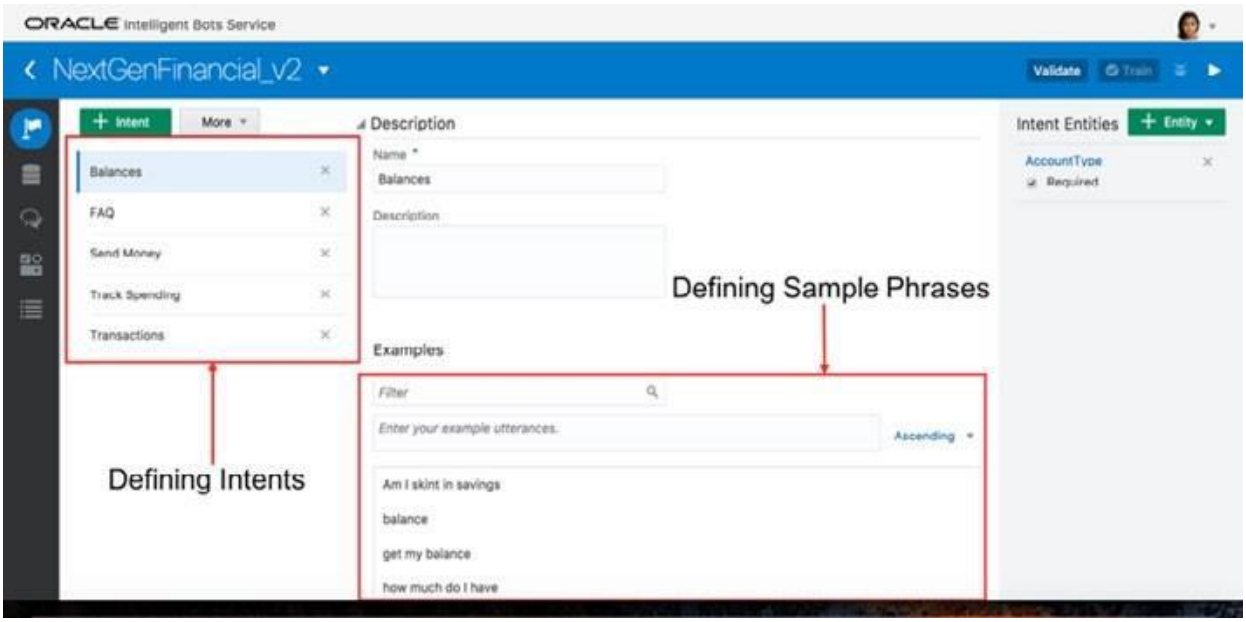

Рисунок 2.8 – Визначення намірів і введення зразків фраз (висловлювань)

Аналітика взаємодії з інтелектуальними ботами

Платформа Oracle Mobile Cloud Enterprise (OMCe) включає засіб Customer Experience Analytics (CxA), що забезпечує поглиблений аналіз взаємодії з користувачами через різні канали, включаючи веб-сайти, мобільні

пристрої і боти. З його допомогою можна отримати відповіді на складні аналітичні питання: який канал використовується найчастіше, які сценарії використання найбільш популярні при взаємодії з мобільними додатками / ботами, чи користуються кінцеві користувачі декількома каналами і, якщо так, за якими характеристиками (час, особисті уподобання) вони вибирають той чи інший канал. Канали є джерелами цінних знань, необхідних для персоналізації взаємодії з кінцевим користувачем. За допомогою СхА можна формувати сегменти користувачів, виконувати когортний аналіз і будувати деталізовані воронки.

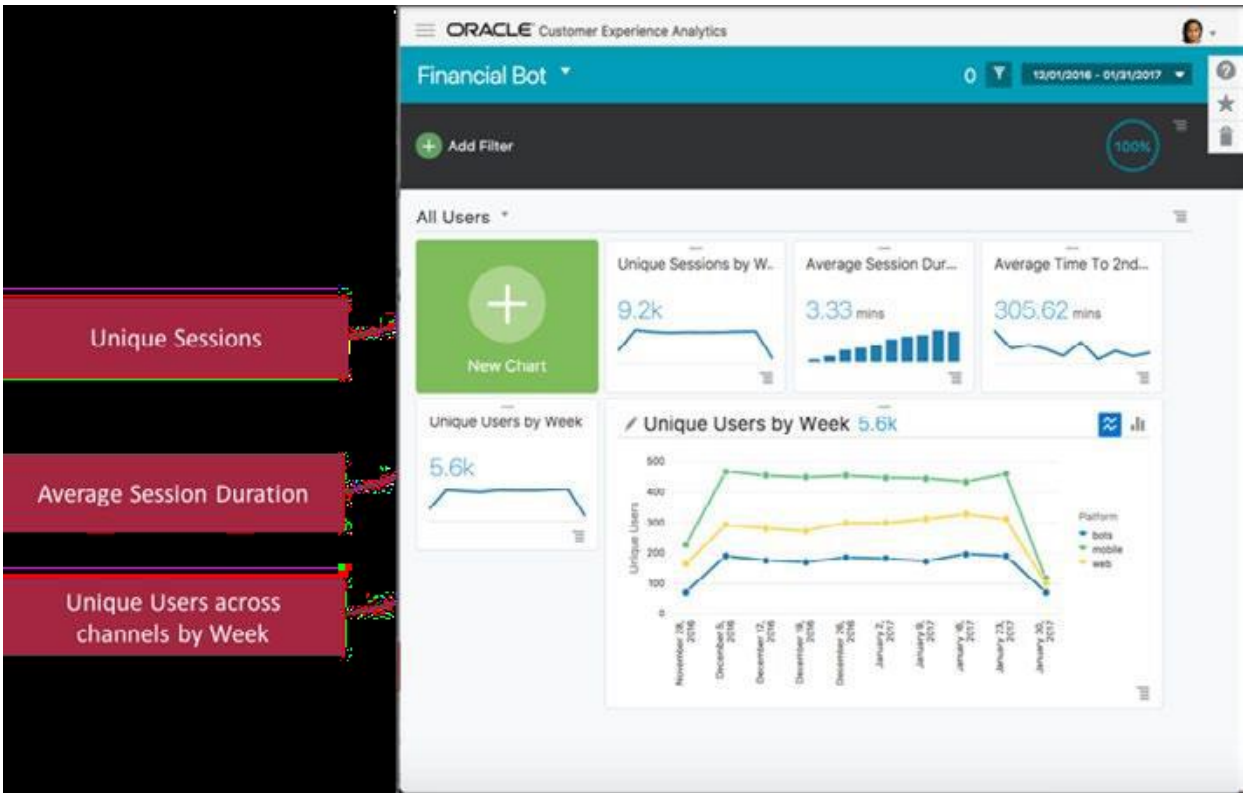

Рисунок 2.9 – Аналітика популярності і використання різних каналів

Оперативний моніторинг інтелектуальних пошукових роботів. СхА надає аналітику про функціонування ботів, оцінювати ступінь їх зручності для користувачів, а також навчати модель, тестуючи різні напрямки розмови. І все це в режимі реального часу. Безпосередньо для інтелектуальних роботів в

СхА передбачені готові звіти та інструменти для оптимізації ефективності ботів і шляхів кінцевого користувача. Зокрема, за допомогою СхА корпоративні користувачі зможуть отримати наступну інформацію:

- загальна інформація про використання інтелектуальних пошукових роботів.
- ключові метрики використання інтелектуальних роботів і взаємодії з ними.
- динаміка точності визначення намірів.
- популярні фрази, використовувані користувачами.
- популярні шляху проходження розмови за сценарієм діалогу, найпоширеніші варіанти вибору.
- звіти про користувачів: діючі, нові, неактивні; розбиття по регіонах, пристроїв і каналах.
- звіти про повідомлення: кількість сеансів інтелектуальних роботів з розбивкою по каналах, написав.

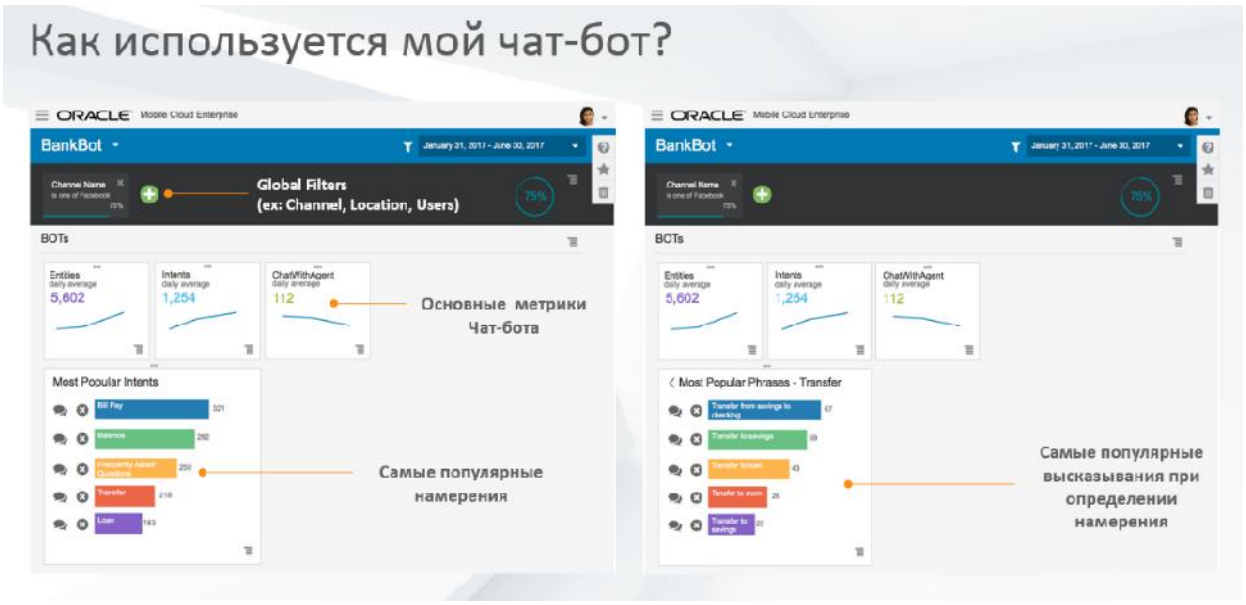

Рисунок 2.10 – Популярність і використання інтелектуальних роботів по різних каналах

Автоматично формуються діагностичні звіти для розробників дозволяють усувати проблеми в роботі чат-бота. Нижче наведені приклади таких звітів:

- де виникли проблеми і як їх виправити?
- повністю пройдені напрямки: найбільш (зелені) і найменш (помаранчеві) популярні напрямки.
- незавершені напрямки: історія всіх покинутих сеансів (червоні) з показниками помилок і текстами листування для аналізу і вирішення проблеми.
- набір відображуваних метрик включає помилки слоттінга, системні помилки, помилки схеми діалогу, помилки компонентів, помилки каналів, тональність.

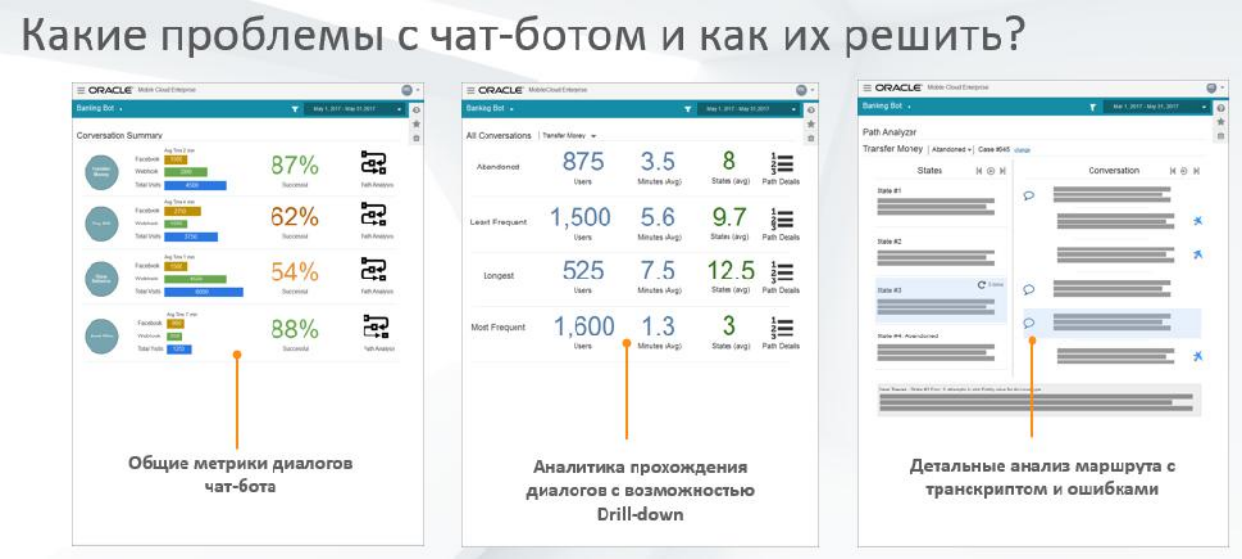

Рисунок 2.11 – Діагностика інтелектуальних роботів - детальна інформація про напрямки діалогів

Що далі? Платформа штучного інтелекту для розробників.

У сфері штучного інтелекту ми застосовуємо підхід, заснований на сценарії використання. Розуміючи імпульс індустрії і усвідомлюючи великі можливості для використання чат-ботів, ми застосували алгоритми

машинного навчання, що дозволяють будувати новий клас розмовних програм (чат-ботів) на основі штучного інтелекту. Структуру наших сервісів в області штучного інтелекту можна представити у вигляді наступних п’яти ключових компонентів: машинне навчання, когнітивні сервіси, сервіси знань, діалог і контекст, дані і аналітика.

Платформа штучного інтелекту Oracle

Об’єднавши просунуті методи машинного навчання, обробку природної мови, діалоги, засоби перекладу та систему ведення розмов у форматі «питання-відповідь» на основі машинного навчання, ми розробили механізм для створення ботів зі штучним інтелектом. Цей механізм дозволяє будувати потужні програми і відразу ж приносити користь нашим замовникам. Ми продовжуємо роботу над Oracle Intelligent Bots і інвестуємо значні кошти в розгортання безлічі нових функцій штучного інтелекту, що розширюють можливості платформи. Зокрема, мова йде про такі функції, як триваюче контрольоване самонавчання, розпізнавання тональності, обробка зображень і розпізнавання об’єктів, прогнозне моделювання, рекомендаційні можливості, поглиблення аналітики, витяг іменованих сутностей (named entity extraction, NER) з використанням машинного навчання, а також автозавершення. І це далеко не повний список. Згодом планується впровадження онтологій, графів знань і семантичних технологій для подальшого удосконалення можливостей обробки та розуміння природної мови. Поєднуючи широке коло джерел даних, доступних через додатки Oracle, з загальнодоступними джерелами даних і застосовуючи новітні засоби контрольованого і неконтрольованого машинного навчання, обробки природної мови та семантичні методи, ми можемо побудувати і розширити високоякісний граф знань спеціально для конкретної організації. графів знань і семантичних технологій для подальшого удосконалення можливостей обробки та розуміння природної мови. Поєднуючи широке коло джерел даних, доступних через додатка Oracle, з загальнодоступними джерелами даних і застосовуючи новітні засоби контрольованого і неконтрольованого

машинного навчання, обробки природної мови та семантичні методи, ми можемо побудувати і розширити високоякісний граф знань спеціально для конкретної організації. графів знань і семантичних технологій для подальшого удосконалення можливостей обробки та розуміння природної мови. Поєднуючи широке коло джерел даних, доступних через додатки Oracle, з загальнодоступними джерелами даних і застосовуючи новітні засоби контрольованого і неконтрольованого машинного навчання, обробки природної мови та семантичні методи, ми можемо побудувати і розширити високоякісний граф знань спеціально для конкретної організації.

Платформа Oracle Intelligent Bots будується на архітектурі мікросервісов. Кожен з механізмів машинного навчання спроектований і функціонує як мікросервіс. У платформу Oracle Intelligent Bots вбудований фреймворк для інтеграції з різними каналами, включаючи Facebook Messenger, WeChat, WhatsApp, нативні мобільні додатки, веб-додатки та ін. Спілкування з клієнтами здійснюється через платформу Oracle Intelligent Bots, тому функцію розпізнавання мови бере на себе мікросервіс обробки природної мови. Цей же мікросервіс доступний кожному розробнику, який бажає впровадити відповідні можливості в власний додаток. Завдяки високій гнучкості архітектури ми готові запропонувати розробникам «II як послугу» (AI as a Service, AIaaS),

Ми застосували підхід «зверху вниз» до розробки складових платформи штучного інтелекту. Інтелектуальні боти стали вихідною точкою, що визначила ключові пріоритети стосовно інфраструктурі, алгоритмам, моделям і фреймворк. Ці II-компоненти будуть доступні в складі Oracle PaaS у вигляді мікросервісов для побудови високоінтелектуальних додатків.

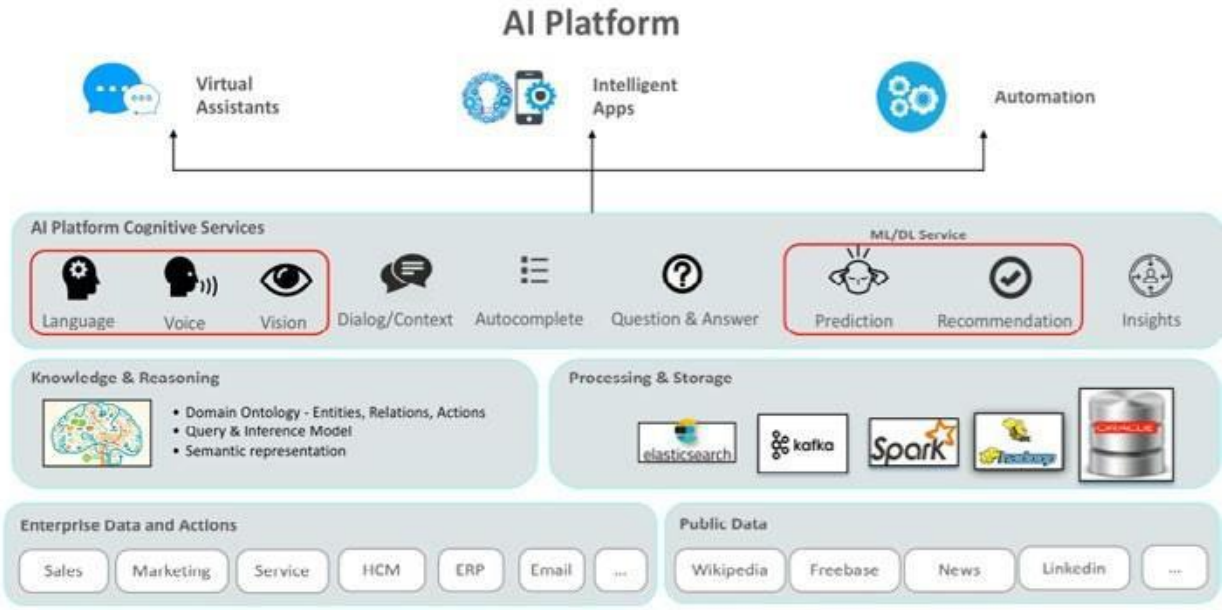

Рисунок 2.12 – Платформа Oracle Intelligent Bots

Oracle Intelligent Bots.

Єдиний комплексне рішення. Наше рішення містить все необхідне для створення чат-ботів. Ми пропонуємо інтеграцію з каналами, управління сценаріями діалогів, штучний інтелект, кошти інтеграції з платформою і зручний призначений для користувача інтерфейс Oracle Intelligent Bots Builder, який об’єднує всі ці можливості. відміну від інших пропозицій

Єдиний багатоканальне рішення. Рішення Oracle Intelligent Bots створено на основі зрілої та успішної мобільної платформи Oracle. Чат-боти відмінно підходять для побудови розмовного інтерфейсу, але не призначені для збору структурованих даних і введення даних через форми. Такі завдання вирішуються за допомогою мобільного додатку. Максимальний ступінь зручності для користувача досягається за умови, що платформа надає прості і потужні засоби для забезпечення взаємодії чат-ботів, мобільних додатків і веб-сайтів. На відміну від нас наші конкуренти пропонують набір окремих компонентів. Тільки ми готові запропонувати рішення, ефективно об’єднує збір структурованих даних і неструктурований розмовний призначений для користувача інтерфейс.

Інтеграція з корпоративними системами. Завдання платформи Oracle Intelligent Bots полягає в тому, щоб з високою точністю видати потрібні кінцевому користувачеві дані. Oracle є лідером в області інтеграції з корпоративними і хмарними джерелами даних завдяки готовому механізму, який заощадить сили, час і кошти на підготовку даних для чат-ботів. Oracle PaaS охоплює повний комплекс засобів, включаючи сервіс Oracle Integration Cloud Service (ICS) (забезпечує інтеграцію з більш ніж 75 корпоративними та іншими серверними системами), хмарний сервіс Apiary (спрощує проектування API-інтерфейсів) і API-сервіс для захисту і настройки API-політик.

Аналітика. До складу платформи входить засіб Customer Experience Analytics (CxA) – єдине вікно для доступу до результатів аналізу взаємодії з користувачами через різні канали, включаючи мобільні пристрої, чат-боти і веб-сайти. З його допомогою можна отримати відповіді на складні аналітичні питання: «який канал використовується найчастіше», «які сценарії використання найбільш популярні при взаємодії з мобільними додатками / ботами», «чи користуються кінцеві користувачі декількома каналами і, якщо так, за якими характеристиками (час, особисті уподобання) вони вибирають той чи інший канал». Канали є джерелами цінних знань, необхідних для персоналізації взаємодії з кінцевим користувачем.

Оперативний моніторинг інтелектуальних пошукових роботів. Оцінка якості роботи чат-бота, виявлення потенційних проблем і відстеження діалогу для навчання моделі в режимі реального часу.

Від чат-бота до людини і навпаки. Виявлення ситуацій, в яких чат-бот не здатний адекватно відреагувати на запит кінцевого користувача, а також миттєве перемикання на співробітника, який може таким же чином передавати діалоги чат-ботів.

Управління життєвим циклом. Комплексне управління життєвим циклом чат-бота, що охоплює розробку, тестування, виробництво, управління

версіями, інтеграцію, а також управління за схемою CI/CD (безперервна інтеграція/безперервне розгортання).

Безперервна еволюція. Ми постійно впроваджуємо додаткові алгоритми, що спрощують розробку чат-ботів. Незабаром будуть додані алгоритми для розпізнавання тональності введеного користувачем тексту, аналізу зображень, перекладу, самонавчання, ведення розмов у форматі «питання-відповідь», аналізу поведінки та ін. Розробники чат-ботів зможуть легко задіяти потрібні додаткові алгоритми, не витрачаючи час і сили на вивчення, відбір, розробку і настройку відповідних базових моделей можливості Oracle PaaS. Розробники можуть створювати власні додатки, користуючись потрібними сервісами машинного навчання, наприклад, засобами обробки природної мови або розпізнавання зображень.

Oracle Cloud. Платформа пропонується у вигляді комплексного вирішення в Oracle Cloud, що дозволяє скоротити витрати і знизити трудомісткість і складність завдань, пов’язаних з установкою патчів і оновлень. Надійне і безпечне хмара забезпечує резервне копіювання, високу доступність і відмовостійкість. Oracle Cloud відкриває величезні можливості, починаючи з потужних графічних процесорів на рівні IaaS і закінчуючи доступом до різних функцій PaaS і вбудованим засобам інтеграції з додатками Oracle SaaS і готовими чат-ботами.

Досвід розробки вертикальних додатків. Oracle забезпечує найбільш повну підтримку вертикальних рішень, оскільки має відповідне спеціалізоване підрозділ, що розробляє вертикальні рішення на основі чат-ботів.

Досвід замовників. У нас вже є кілька замовників (B2C і B2E) в різних ринкових сегментах. Ці компанії порівняли наше рішення з рішеннями конкурентів і зробили вибір на користь Oracle Intelligent Bots.

Штучний інтелект може зробити істотний вплив на всі галузі, відкривши нові способи взаємодії з замовниками і нові інструменти ефективної роботи по всьому світу. Задумайтесь, чат-боти ніколи не сплять,

не змушують чекати і ефективно персоналізують спілкування з клієнтом. Новітні з них – чат-боти зі штучним інтелектом - здатні навчатися і будувати відносини. Навіть якщо штучний інтелект не зробить глобальної революції, він залишиться важливим інструментом для оптимізації взаємодій, підвищення ефективності і зниження витрат. Будь-який сучасний лідер IT-індустрії, що відповідає за стратегію автоматизації, рано чи пізно зіткнеться з необхідністю впровадження штучного інтелекту.

2.5 Розробка моделі багаторівневого чат-бота

Пропонується нова теоретико-множинну модель багаторівневого бота:

$$BL = \bigcup_{i=1}^{L} BL_i, BL_i =< P_i^j >, j = \overline{1, K_i} \quad (2.1)$$

де L – кількість рівнів програмного забезпечення, K_i - кількість програмних компонент чат бота на і-му рівні; P_i^j - програмний компонент забезпечення чат-бота.

Програмні компоненти реалізують певну функціональність та об’єднуються за допомогою управляючої програми або за допомогою хмарних сервісів. Якщо не існує протоколу взаємодії програмного компонента, тобто відсутні стандартні програмні інтерфейси, розробляється програма-адаптер, яка виконує функції запуску, підтримки виконання та організують взаємодію з іншими програмними компонентами. Таким чином з усієї множини програмних компонент P можливо виділити наступні підмножини:

$$P =< Pm, Pf, Pi, Pa > \quad (2.2)$$

Де Pm – множина програмних компонент управляння, Pf – множина програмних компонент, які виконують конкретну функціональну дію, Pi – інтерфейсні програмні компоненти, які забезпечують передачу інформації, тобто канали зв’язку, Pa – допоміжне програмне забезпечення (програми-адаптери), яке розробляється для включення у систему керування компонентів, що не мають стандартних інтерфейсів.

Сформулюємо декілька властивостей програмного забезпечення, які необхідно виконувати при реалізації чат-ботів у випадку трьохрівневої моделі:

1. $(Pm \neq \varnothing) \wedge (Pf \neq \varnothing) \wedge (Pi \cup Pa \neq \varnothing).$
2. $Pm \cap Pf \cap Pi \cap Pa = \varnothing$
3. BL = <BF, BB, Br>,

де BF – підмножина програмних компонент, що забезпечують FrontEnd, BB – підмножина програмних компонент, що забезпечують BackEnd, Br – підмножина зовнішніх компонент, що забезпечують взаємодію з розподіленими ресурсами.

2.6 Розробка методу адаптації чат-бота

Метод адаптації чат-бота дає змогу реконфігурації як складу програмного забезпечення так і властивостей візуального, слухового, інтелектуального сприйняття користувачем.

Метод містить наступні етапи:

Ініціалізація управляючої програми Pm^{global} на сервері, загрузка конфігураційного файлу, який містить інформацію о наявних програмних компонентах $Pf \cup Pi \cup Pa$ та наявних обчислювальних ресурсах $R = \bigcup_{i-1}^{M} R_i$, де M – кількість наявних комп’ютерів.

Встановлення зв’язку з доступними ресурсами з множини R та завантаження на кожному з них термінального клієнта Pm_i^{local}, $i = \overline{1, M'}$, де $M' \le M$. Остання нерівність відображає той факт, що кількість доступних ресурсів може буди менша за заплановану.

Встановлення зв’язку з програмними компонентами з множини $Pf \cup Pi \cup Pa$, що вже виконуються на віддалених обчислювальних ресурсах або їх завантаження та запуск за допомогою термінальних клієнтів Pm_i^{local}.

Ідентифікація клієнта *C*, пошук профілю клієнта C^{prof} в базі даних. Якщо профіль знайдений, перехід до етапу 7.

Встановлення профілю за замовчуванням.

Завантаження програмних компонент з множини *Pf*, які відповідають за початковий збір інформації о користувачі (історія пошуків, перелік сайтів та програмного забезпечення користувача, аналіз часу використання комп’ютера та інш.).

Завантаження резидентних програмних компонент з множини *Pf*, які слідкують за діями користувача. Дані програми функціонують увесь час, коли клієнт взаємодіє з чат-ботом, але можлива їх робота у фоновому режимі.

Статистичні дані *St*, які отримуються від резидентних програм зберігаються у базі даних з атрибутами профілю користувача C^{prof}.

При наявності достатніх даних, активуються віддалені програмні компоненти з множини *Pf*, які виконують аналіз профілю користувача та модифікують параметри чат-бота.

Етапи 8 та 9 методу виконуються постійно за умови взаємодії користувача з чат-ботом або за умови функціонування комп’ютера користувача.

При завершенні роботи, виконується збереження отриманих статистичних даних та інформації про реконфігурування чат-боту в базі даних з атрибутами профілю користувача C^{prof}.

Даний метод дозволяє постійно слідкувати за користувачем та робити модифікацію чат-бота згідно початкових даних та отриманих у процесі спілкування.

3 РОЗРОБКА ПРОГРАМНИХ ЗАСОБІВ ПІДТРИМКИ ЧАТ-БОТІВ

3.1 Чат-бот на базі мови програмування Java

Аналіз предметної області в пошукових системах як правило дає уривки знань з розробки ботів, наприклад, як зробити клавіатуру, обробити CallbackQuery тощо. Повного і вичерпного керівництва для розробки ботів на Java не існує. В Інтернеті багато сайтів, де можна створити свого бота з уже готовим деплоем. Але справа в тому, що в своїй більшості створюються боти, які можуть надати довідкову інформацію та інше. Наш бот – повноцінний веб-додаток, до якого можна прив’язати базу даних, виконувати запити на різні API, парсити сайти, виконувати складні обчислення та інше. Справа обмежена тільки вашою фантазією. Бота в Telegram зареєструвати дуже просто, цей процес детально описаний в документації. Для нашого додатка необхідно знати тільки ім’я бота і токен, який ви отримаєте при реєстрації. По суті бот – просто консольний веб-додаток. Ніякого фронтенда, чиста обробка команд (бекенд) з використанням Hibernate та JSON. Щоб підключити залежність в pom.xml (мається на увазі що використовується Maven). Зробити це можна так:

```
<dependency>
 <groupId>org.telegram </ groupId>
 <artifactId>telegrambots </ artifactId>
 <version>3.5</ Version>
</ Dependency>
```

Приклад 3.1 – Підключення основного модуля

Потім створюємо класBot, Успадкуємо його від класу TelegramLongPollingBot, перевизначивши його методи:

```
public class Bot extends TelegramLongPollingBot {
 / **
 * Метод для прийому повідомлень.
 * @Param update Містить повідомлення від користувача.
 * /
 @Override
 public void onUpdateReceived(Update update) {
String message = update.getMessage().getText();
sendMsg(update.getMessage().getChatId().(), message);
 }

 / **
 * Метод для налаштування повідомлення і його відправки.
 * @Param chatId id чату
 * @Param s Рядок, яку необходимоти відправити у вигляді повідомлення.
 * /
 public synchronized void sendMsg(String chatId, String s) {
 SendMessage sendMessage = new SendMessage();
 sendMessage.enableMarkdown(true);
 sendMessage.setChatId(chatId);
 sendMessage.setText(s);
 try {
 sendMessage(sendMessage);
 } catch (TelegramApiException e) {
 log.log(Level.SEVERE, "Exception:", e.());
 }
 }
 / **
 * Метод повертає ім’я бота, вказане при реєстрації.
 * @Return ім’я бота
 * /
 @Override
 public String getBotUsername() {
 return "BotName";
 }

 / **
 * Метод повертає token бота для зв’язку з сервером Telegram
 * @Return token для бота
 * /
 @Override
 public String getBotToken() {
 return "BotToken";
 }
}
Ну і вміст методу main:
public static void main(String[] args) {
 ApiContextInitializer.init();
 TelegramBotsApi telegramBotsApi = new TelegramBotsApi();
 try {
 telegramBotsApi.registerBot(Bot.getBot());
 } catch (TelegramApiRequestException e) {
 e.printStackTrace();
 }
}
```

Приклад 3.2 – Методи основного класу

Вписавши в методи getBotUsername() і getBotToken() запускаємо бота. Поки він тільки перенаправляє нам будь-які повідомлення, які ми відправимо йому. Працює це все наступним чином: коли ви запускаєте додаток, додаток починає раз в n кількість секунд відправляти на сервер Telegram GET запит за наступним URL: https://api.telegram.org/BotToken/getMe, де BotToken – токен вашого бота, отримуючи у відповідь JSON, в якому знаходяться всі повідомлення. Кожне таке повідомлення обробляється бібліотекою і приходить в методOnUpdateReceived (Update update) об’єктом Update. З ним то ми і працюємо. У цьому вся принадність Telegram-ботів, вони можуть працювати на будь-якому комп’ютері, для тестування потрібно просто запустити додаток, не потрібно деплоіть його на хостинг після кожної зміни. Це дуже зручно. Само собою бота можна налаштувати на роботу по вебхуку, керівництво можна знайти на просторах Інтернету, ми будемо для простоти працювати по LongPolling. Те як обробляти повідомлення і що відправляти у відповідь обмежена тільки лише засобами мови і бібліотекою, все інше на ваш розсуд. Ви можете зробити бота, який буде шукати для вас відео на YouTube, можете зробити бота, який кожен день буде надсилати вам те, що ви відправите собі, наприклад, за рік, таку собі капсулу часу. А можете навчитися інтегруватися до CRM-систем і робити ботів для малого бізнесу, все обмежено вашою фантазією. Йдемо далі. ті,«/», наприклад / start. Але є спосіб зручніше - кнопки. Є два види кнопок: ті, що з’являються під полем введення,ReplyKeyboardMarkup і кнопки, які знаходяться безпосередньо під повідомленням, до якого прив’язані, InlineKeyboardMarkup. У документації можливо поверхнево ознайомитися з їх описом. ReplyKeyboardMarkup. По суті це - масив масивів кнопок,List <KeyboardRow <KeyboardButton >>. Ось приклад коду, який створює клавіатуру.

У методі sendMsg() викликається цей метод, передаючи йому повідомлення, таким чином встановлюючи для такого повідомлення клавіатуру. Коли відправимо це повідомлення користувачу, то він побачить текст повідомлення, який ми встановили, а також 2 кнопки, на яких буде

написано Привіт і Допомога, один під одним.

```
public synchronized void setButtons(SendMessage sendMessage) {
 // Створюємо клавіуатуру
 ReplyKeyboardMarkup replyKeyboardMarkup = new ReplyKeyboardMarkup();
 sendMessage.setReplyMarkup(replyKeyboardMarkup);
 replyKeyboardMarkup.setSelective(true);
 replyKeyboardMarkup.setResizeKeyboard(true);
 replyKeyboardMarkup.setOneTimeKeyboard(false);

 // Створюємо список рядків клавіатури
 List<KeyboardRow> keyboard = new ArrayList <>();

 // Перший рядок клавіатури
 KeyboardRow keyboardFirstRow = new KeyboardRow();
 // Додаємо кнопки в перший рядок клавіатури
 keyboardFirstRow.add(new KeyboardButton("Привіт"));

 // Другий рядок клавіатури
 KeyboardRow keyboardSecondRow = new KeyboardRow();
 // Додаємо кнопки в другу строчку клавіатури
 keyboardSecondRow.add(new KeyboardButton("Допомога");

 // Додаємо все рядки клавіатури в список
 keyboard.add(keyboardFirstRow);
 keyboard.add(keyboardSecondRow);
 // і устанвалівать цей список нашої клавіатурі
 replyKeyboardMarkup.setKeyboard(keyboard);
 }
```

Приклад 3.3 – Створення клавіатури

При натисканні на ці кнопки боту буде відправлено повідомлення, текст якого є те, що написано на кнопці. Тобто якщо клієнт натисне «Допомога», то боту прийде повідомлення з текстом «Допомога». Для нього це як нібито клієнт сам написав текст «Допомога» і відправив би йому. Ну а потім ви обробляєте такі повідомлення. InlineKeyboardMarkup Це теж масив масивів, він схожий на попередній Markup, але логіка роботи тут трохи інша. Така клавіатура прив’язується до певного повідомленням і існує тільки для нього. Ось метод для установки Inline-клавіатури. У прикладі створюється List. Додаємо в перший рядок Inline-кнопку. Така кнопка може містити URL, посилання на канал або жCallbackQuery, Про яку я напишу трохи пізніше. Тут ми встановлюємо текст для нашої кнопки, який буде бачити користувач, а потім встановлюємо дані, які будуть відправлені боту. У нашому прикладі

користувач бачить «Привіт», а боту при натисканні відправиться число 17, це і є нашCallbackQuery. Пару слів проCallbackQuery.

```
      private void setInline() {
       List <List<InlineKeyboardButton>> Buttons = new ArrayList <>();
       List<InlineKeyboardButton> buttons1 = new ArrayList <>();
       buttons1.add(new
InlineKeyboardButton().setText("Кнопка").setCallbackData(17));
       buttons.add(buttons1);

       InlineKeyboardMarkup markupKeyboard = new InlineKeyboardMarkup();
       markupKeyboard.setKeyboard(buttons);
       }
```

Приклад 2.4 – Установка Inline-клавіатури

Для отримання таких даних з об’єктаUpdate потрібно виконати update.getCallbackQuery (), Цей метод повертає CallbackQuery, 3 якого вже можна отримати дані, передані боту. Не потрібно намагатися отримати ці дані через методupdate.getMessage (). getText (), отримаєте NullPointerException.

```
      @Override
       public void onUpdateReceived(Update update) {
       if(update.hasMessage()) {
       ThreadClass thread = new ThreadClass(update.getMessage());
       } else if(update.hasCallbackQuery()) {
       AnswerCallbackThread           answerThread           =           new
AnswerCallbackThread(update.getCallbackQuery());
       }
       }
```

Приклад 3.5 – Перезавантаження методу

Якщо є повідомлення, відправляємо на обробку в новий потік повідомлення, якщо є CallbackQuery, Відправляємо його на обробку в відповідний потік. На CallbackQuery можна відправляти відповідь. У кожного об’єкта в Telegram є свій id. Для відправлення відповіді на певний CallbackQueryпотрібно знати лише його id, який ми отримаємо з відповідного об’єкта. Для відправлення відповіді викличемо такий метод:

```
      public synchronized void answerCallbackQuery(String callbackId, String
message) {
       AnswerCallbackQuery answer = new AnswerCallbackQuery();
       answer.setCallbackQueryId(callbackId);
       answer.setText(message);
       answer.setShowAlert(true);
       try {
       answerCallbackQuery(answer);
       } catch (TelegramApiException e) {
       e.printStackTrace();
       }
       }
```

Приклад 3.6 – Відправлення повідомлення

Текст у відповіді на CallbackQuery повинен бути не довше 200 символів. При відправці такої відповіді клієнт отримає спливаюче вікно, в якому буде написано повідомлення. Таке вікно може зникнути через кілька секунд саме після появи, а може висіти до тих пір, поки користувач не натисне ок. Для перемикання цих режимів ми викликаємо метод answer.setShowAlert (true). При true віконце висить до натискання ок, при false – зникає через 5 секунд. В принципі, це всі базові особливості бібліотеки Telegram bot. Такі речі як відправка мультимедіа, геолокації тощо, при бажанні можно освоїти з документації. Перейдемо до загрузки розробленого бота на хостингу. Для свого проекту я вибрав Heroku, так как, на мою думку, це досить зручний хостинг, який має свій CLI. Він безкоштовний, але на такому тарифі ваш бот при відсутності запитів буде йти в сплячку через 30 хвилин. Коли ж до нього буде відправлений запит, він прокидається. Це відбувається досить швидко, ви навіть не помітите (якщо звичайно коннект до БД не піднімається заново). Обмеження на безкоштовний тариф - 5МБ база даних, 100МБ дисковий простір, 2ТБ трафіку в місяць, 1 дино. Діно - це ваше запущене застосування. Саме стадія деплоя може викликати труднощі. Heroku при Деплой вимагає наявності файлу з ім’ям Procfile (без розширення). Створюємо його в корені проекту, пишемо туди worker: sh target / bin / workerBot workerBot - ім’я, яке ми вказуємо в pom.xml. Запускатиметься sh скрипт, що генерується за допомогою Maven плагіна appassembler-maven-plugin. В скрипті описаний

запуск скомпільованої jar. Ім’я запускається класу вказується між <mainClass> </ mainClass>, ім’я скрипта між <name> </ name> pom.xml:

```
<build>
 <plugins>
 ...
 <plugin>
 <groupId>org.codehaus.mojo </ groupId>
 <artifactId>appassembler-maven-plugin </ artifactId>
 <version>1.1.1</ Version>
 <configuration>
 <assembleDirectory>target </ assembleDirectory>
 <programs>
 <program>
 <mainClass>com.home.server.TelegramBot </ mainClass>
 <name>workerBot </ name>
 </ Program>
 </ Programs>
 </ Configuration>
 <executions>
 <execution>
 <phase>package</ Phase>
 <goals>
 <goal>assemble </ goal>
 </ Goals>
 </ Execution>
 </ Executions>
 </ Plugin>
 </ Plugins>
</ Build>
```

Приклад 3.7 – Скрипт запуску додатка

Перед початком цього процесу вам слід зареєструватися на Heroku, встановити Git і Heroku CLI. Якщо вашому додатку необхідна БД, то при оформленні нового додатка не забудьте додати потрібну вам БД. Далі вам необхідно дізнатися host, username, password і port вашої БД, а після вказати в своєму додатку. Далі перед деплоем виконайте збірку вашого проекту за допомогою Maven:

```
mvn clean install
```

Для початку ми переходимо в каталог нашого проекту, ініціалізуемо репозиторій командою git init Потім додаємо в цей репозиторій свій проект:

```
git add.
```

Після коммітимо зміни:

```
git commit -m "First commit in project"
```

Далі потрібно залогінитися на heroku, пишемо в командному рядку:

```
heroku login
```

Вводимо свої дані, вказані при реєстрації. Після цього потрібно дізнатися URL вашого сховища на heroku, робиться це в налаштуваннях. потім пишемо:

```
git remote add heroku[url]
```

Для сховища додасться віддалений репозиторій heroku, тому далі пишемо:

```
git push heroku master
```

Далі чекаємо . При успішному Деплої додатку, виконуємо команду:

```
heroku ps: scale worker =1
```

Додаток запущено. Якщо ж цього не сталося, перегляньте уважно логи, швидше за все виникає помилка в вашому додатку, через яку воно впало.

3.2 Особливості розробки чат-бота Telegram з використанням мови програмування Python

Для початку нам необхідно зареєструвати в Telegram майбутнього бота. Це робиться в такий спосіб:

- необхідно встановити додаток Telegram на телефон або комп’ютер.

- додаємо до себе в контакт-лист бота з ім’ям BotFather.
- запускаємо процедуру «спілкування» з ботом натисканням кнопки Start. Далі перед нами постане список команд.
- для того, щоб створити нового бота необхідно виконати команду / newbot і слідувати інструкціям. Зверніть увагу, що username для бота повинен завжди містити в кінці слово bot. Наприклад, DjangoBot або Django_bot.
- для нашого бота треба вибрати ім’я, наприклад PythonPlanetBot, так як його основна функція полягає в парсінгу RSS feed сайту Python Planet і видача інформації про останні постах користувачеві.

Після створення бота, треба звернути увагу на рядок з текстом: «Use this token to access the HTTP API».

За якою слідкує назва tokeny, за яким ми будемо маніпулювати нашим ботом. Крім функції створення telegram бота, BotFather також має ряд інших можливостей:

- присвоїти боту опис,
- встановити аватар,
- поміняти token.

І так далі. Повний опис доступних команд можна побачити на першому скріншоті.

Наступним етапом треба розробити код основного додатку.

Як раніше вже згадувалося, ми будемо писати веб-додаток на Django. Але варто зазначити, що це робити необов’язково. Можна обійтися і звичайним Python скриптом, правда в цьому випадку необхідно буде періодично опитувати Telegram на предмет нових запитів від користувачів бота (використовуючи метод getUpdates) і збільшуючи offset для отримання найостанніших даних без повторень. У Telegram існує два взаємовиключних методу отримання команд / повідомлень для вашого бота.

- використання виклику API методу getUpdates.
- установка Webhook.

Установка Webhook полягає в передачі боту спеціального URL адреси на який буде надходити POST запит кожен раз, коли хтось почне посилати повідомлення боту. Саме цей варіант ми і будемо використовувати для взаємодії між ботом і його користувачем. Для того, щоб задати URL, необхідно використовувати API метод set Webhook. Зазначу, що URL повинен починатися з https, тобто мати захищене SSL з'єднання з дійсним сертифікатом. Telegramдозволяє використовувати само подпісанний сертифікат, правда для цього необхідно в методі setWebhook передавати також публічний ключ в PEM форматі (ASCII base64). Або ж можна отриматиДійсний безкоштовний SSL сертифікат від Let's Encrypt.

Детальніше про getUpdates і set Webhook можна почитати відповідно [10,12].

Отже, повернемося до Python бібліотеці для роботи з Telegram - telepot. На поточний момент користуємося версією 6.7. Встановлюємо її в віртуальне оточення python virtualenv:

```
pip install telepot
```

Змінної token присвоюємо значення токена, отриманого при створенні бота через BotFather. У підсумку після виконання цих команд ми отримаємо:

```
{u'username ': u'PythonPlanetBot ', u'first_name ': u'Python Planet Bot ', u'id ': 199266571}
```

Найпростіший варіант взаємодії з Telegram ботом на Python виглядає наступним чином:

```
import telepot
token ='123456'
TelegramBot = telepot.Bot (token)
print TelegramBot.getMe ()
```

Приклад 3.8 – Взаємодія з Telegram ботом

Таким чином, ми викликали найпростіший API запит getMe, який повертає інформацію про бота: username, id, first_name.

Додамо нашого бота до себе в контакт-лист і пошлемо йому першу стандартну команду / start.

Виконуємо код:

```
TelegramBot.getUpdates ()
[{u’message ’: {u’date ’: 1459927254, u’text ’: u ’/ start’, u’from ’: {u’username ’: u’adilkhash ’, u’first_name ’: u’Adil ’, u’id ’: 31337}, u’message_id ’: 1, u’chat ’: {u’username ’: u’adilkhash ’, u’first_name ’: u’Adil ’, u’type ’: u’private ’, u’id ’: 7350}}, u’update_id ’: 649179764}]
```

Приклад 3.9 – Встановлення початкових даних

Процес спілкування з ботом відбувається по HTTPS; для передачі даних використовується JSON. Метод getUpdates повертає список / масив з об’єктів типу . Усередині Update знаходиться об’єкт . Для стандартної взаємодії з ботом нас фактично цікавить саме об’єкт Message, у якого ми зчитуємо атрибут text, який зберігає в собі текст, переданий боту і об’єкт chat, в якому лежить інформація про користувача, який ініціював спілкування з нашим Telegram ботом. Також є параметр update_id, який служить в якості offset параметра при виклику методу getUpdates. Тобто update_id + 1 поверне всі повідомлення, що надійшли після останнього update_id, при цьому всі попередні повідомлення будуть видалені.

На етапі написання найпростішого Telegram бота нам цих викликів досить. Приступимо до написання Django додатки для обслуговування наших користувачів.

Проста функція парсингу RSS виду Planet Python виглядає ось так:

```
from xml.etree import cElementTree
import requests
def parse_planetpy_rss():
 "" "Parses first 10 items from http://planetpython.org/rss20.xml
response = requests.get (’http://planetpython.org/rss20.xml’)
 parsed_xml = cElementTree.fromstring (response.content)
 items = []
 for node in parsed_xml.iter ():
```

```
if node.tag == 'Item':
item = {}
for item_node in list (node):
if item_node.tag == 'Title':
item ['Title'] = Item_node.text
if item_node.tag == 'Link':
item ['Link'] = Item_node.text
items.append (item)
return items [:10]
```

Приклад 3.10 – Функція парсингу

CommandReceiveView чекає POST запит на себе, парсит його і відповідає виходячи із заданої команди. Повноцінне Django додаток можна знайти за цимзасланні. Варто відзначити в коді використання ще одного API виклику – sendMessage. Цей метод відправляє повідомлення заданому користувачеві, використовуючи при цьому chat_id і сам текст повідомлення. Chat_id - це унікальний ідентифікатор чату між користувачем і ботом (його ідентифікатор є у відповіді на запит getUpdates). У Telegram ботів є одне обмеження, вони не можуть посилати повідомлення користувачам, які попередньо не ініціювали спілкування з ним. Мабуть це зроблено щоб уникнути масового створення спам-ботів.

Тут можливо використання python бібліотеки requests для роботи з HTTP у самому простому варіанті без обробки помилок. Django "в'юшки" виглядає наступним чином:

```
TOKEN = '<Our_Bot_Token>'
TelegramBot = telepot.Bot (TOKEN)
def _display_help():
 return render_to_string ('Help.md')
def _display_planetpy_feed():
 return render_to_string ('Feed.md', {'Items': Parse_planetpy_rss ()})
class CommandReceiveView (View):
 def post(Self, request, bot_token):
 if bot_token! = TOKEN:
 return HttpResponseForbidden ('Invalid token')
 commands = {
 '/ Start': _Display_help,
 'Help': _Display_help,
 'Feed': _Display_planetpy_feed,
 }
 try:
 payload = json.loads (request.body.decode ('Utf-8'))
 except ValueError:
```

```
        return HttpResponseBadRequest (’Invalid request body’)
        else:
        chat_id = payload [’Message’] [’Chat’] [’Id’]
        cmd = payload [’Message’] .Get (’Text’) # command
        func = commands.get (cmd.split () [0] .Lower ())
        if func:
        TelegramBot.sendMessage (chat_id, func (), parse_mode =’Markdown’)
        else:
        TelegramBot.sendMessage (chat_id, ’I do not understand you, Sir!’)
        return JsonResponse ({}, status =200)
        @method_decorator (csrf_exempt)
        def dispatch(Self, request, * args, ** kwargs):
        return super (CommandReceiveView, self) .dispatch (request, * args, **
kwargs)
```

Приклад 3.11 – Візуальний інтерфейс додатку

Необхідно налаштували оточення і встановили всі необхідні залежності: Django, requests, telepot. Отже, веб-додаток на Django запущено. Наступний етап – тестування бота. Необхідно симулювати дії Telegram сервісу. Для цього нам знадобиться HTTP клієнт і тіло запиту. Як HTTP клієнта часто використовується Chrome плагін під назвою Postman. Тіло запиту ми візьмемо безпосередньо з даних, отриманих за допомогою API виклику getUpdates.

Після запуску runserver, URL на який необхідно надсилати запит виглядає наступним чином:

```
http: //127.0.0.1: 8000 / planet / b ...BOT_TOKEN/
```

де BOT_TOKEN - це токен нашого бота.

Наступним кроком є деплой нашого Django додатки на віддалений хост і наступний виклик методу setWebhook для передачі URL на який буде надсилатися POST запит від сервісу Telegram кожен раз при надходженні команд боту від користувачів.

ВИСНОВКИ

Під час виконання магістерської атестаційної роботи розроблена модель і метод адаптації багаторівневого чат-бота, які дозволять автоматизувати процес створення бота з урахуванням індивідуальних особливостей користувача.

Для реалізації мети було вирішено ряд задач:

Проведено аналіз сучасного стану програмних засобів створення та імплементації чат-ботов, а також інформаційних технологій, які можливо використовувати для їх впровадження; створення моделі багаторівневого чат-бота.

Розроблена модель багаторівневого чат-бота.

Розроблено метод адоптації багаторівневого чат-бота к користувачу.

Розроблені програмні засоби, які реалізують взаємодію розподілених компонентів чат-бота.

Практична значення отриманих результатів підтверджується наявним програмним забезпеченням, яке демонструє можливість реалізації запропонованих теоретичних досліджень при розробці чат-ботів. Як приклад, у роботі наведені компоненти, які дозволяють використовувати дві різні платформи для створення чат-ботів для відомого сервісу обміну повідомленнями Telegram.

До використаних методів дослідження можна віднести: критичний аналіз теоретичних і прикладних розробок вітчизняних і зарубіжних вчених у галузі розробки моделей та методів розподіленого програмного забезпечення хмарних систем. Для вирішення поставлених завдань використані методи системного аналізу, теорії множин (для розробки моделі багаторівневого чат-бота), загальної теорії систем (метода адоптації багаторівневого чат-бота к користувачу), теорії імітаційного моделювання (для моделювання обчислювальних процесів).

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1.Боты как альтернатива мобильным приложениям. – 2016. – Режим доступа: http://webzavod.ru/blog/boty-kak-alternativa-mobilnym-prilozheniyam.

2. Telegram API. – 2016. – Режим доступа: https://core.telegram.org/.

3. Автоматизация при помощи бота или как из истерии по роботам сделать полезное решение. – 2016. – Режим доступа: https://habrahabr.ru/company/flowwow/blog/301996/

4. Технологическая эволюция: что боты дают людям и бизнесу прямо сейчас. – 2016. – Режим доступа: https://vc.ru/p/bots-evolution/

5. Czajkowski, K., Fitzgerald, S., Foster, I. and Kesselman, C. GRID Information Services for Distributed Resource Sharing, 2001.

6. Realizing the Information Future: The Internet and Beyond. National Academy Press, 1994. http://www.nap.edu/readingroom/books/rtif/.

7. Tierney, B., Johnston, W., Lee, J. and Hoo, G. Performance Analysis in High-Speed Wide Area IP over ATM Networks: Top-to-Bottom End-to-End Monitoring. IEEE Networking, 1996.

8. Beynon, M., Ferreira, R., Kurc, T., Sussman, A. and Saltz, J., DataCutter: Middleware for Filtering Very Large Scientific Datasets on Archival Storage Systems. In Proc. 8th Goddard Conference on Mass Storage Systems and Technologies/17th IEEE Symposium on Mass Storage Systems, 2000, 119-133.

9. Butler, R., Engert, D., Foster, I., Kesselman, C., Tuecke, S., Volmer, J. and Welch, V. Design and Deployment of a National-Scale Authentication Infrastructure. IEEE Computer, 33(12):60-66. 2000.

10. Foster, I., Kesselman, C., Tsudik, G. and Tuecke, S. A Security Architecture for Computational GRIDs. In ACM Conference on Computers and Security, 1998, 83-91.

11. Gasser, M. and McDermott, E., An Architecture for Practical Delegation in a Distributed System. In Proc. 1990 IEEE Symposium on Research in Security

and Privacy, 1990, IEEE Press, 20-30.

12. Howell, J. and Kotz, D., End-to-End Authorization. In Proc. 2000 Symposium on Operating Systems Design and Implementation, 2000, USENIX Association. The Anatomy of the GRID 24

13. Berman, F. High-Performance Schedulers. In Foster, I. and Kesselman, C. eds. The GRID: Blueprint for a New Computing Infrastructure, Morgan Kaufmann, 1999, 279-309. The Anatomy of the GRID 22

14. Berman, F., Wolski, R., Figueira, S., Schopf, J. and Shao, G. Application-Level Scheduling on Distributed Heterogeneous Networks. In Proc. Supercomputing '96, 1996.

15. Frey, J., Foster, I., Livny, M., Tannenbaum, T. and Tuecke, S. Condor-G: A Computation Management Agent for Multi-Institutional GRIDs, University of Wisconsin Madison, 2001.

16. Abramson, D., Sosic, R., Giddy, J. and Hall, B. Nimrod: A Tool for Performing Parameterized Simulations Using Distributed Workstations. In Proc. 4th IEEE Symp. on High Performance Distributed Computing, 1995.

17. Beiriger, J., Johnson, W., Bivens, H., Humphreys, S. and Rhea, R., Constructing the ASCI GRID. In Proc. 9th IEEE Symposium on High Performance Distributed Computing, 2000, IEEE Press.

18. Allcock, B., Bester, J., Bresnahan, J., Chervenak, A.L., Foster, I., Kesselman, C., Meder, S., Nefedova, V., Quesnel, D. and Tuecke, S., Secure, Efficient Data Transport and Replica Management for High-Performance Data-Intensive Computing. In Mass Storage Conference, 2001

19. Hoschek, W., Jaen-Martinez, J., Samar, A., Stockinger, H. and Stockinger, K., Data Management in an International Data GRID Project. In Proc. 1st IEEE/ACM International Workshop on GRID Computing, 2000, Springer Verlag Press.

20. Casanova, H. and Dongarra, J. NetSolve: A Network Server for Solving Computational Science Problems. International Journal of Supercomputer Applications and High Performance Computing, 11(3):212-223. 1997.

21. Nakada, H., Sato, M. and Sekiguchi, S. Design and Implementations of Ninf: towards a Global Computing Infrastructure. Future Generation Computing Systems, 1999.

22. DeFanti, T. and Stevens, R. Teleimmersion. In Foster, I. and Kesselman, C. eds. The GRID: Blueprint for a New Computing Infrastructure, Morgan Kaufmann, 1999, 131-155. The Anatomy of the GRID 23

23. Leigh, J., Johnson, A. and DeFanti, T.A. CAVERN: A Distributed Architecture for Supporting Scalable Persistence and Interoperability in Collaborative Virtual Environments. Virtual Reality: Research, Development and Applications, 2(2):217-237. 1997.

24. Foster, I. and Karonis, N. A GRID-Enabled MPI: Message Passing in Heterogeneous Distributed Computing Systems. In Proc. SC'98, 1998.

25. Gabriel, E., Resch, M., Beisel, T. and Keller, R. Distributed Computing in a Heterogeneous Computing Environment. In Proc. EuroPVM/MPI'98, 1998.

26. Casanova, H., Obertelli, G., Berman, F. and Wolski, R., The AppLeS Parameter Sweep Template: User-Level Middleware for the GRID. In Proc. SC'2000, 2000.

27. Goux, J.-P., Kulkarni, S., Linderoth, J. and Yoder, M., An Enabling Framework for Master-Worker Applications on the Computational GRID. In Proc. 9th IEEE Symp. on High Performance Distributed Computing, 2000, IEEE Press.

28. I Foster,C Kesselman, S Tuecke, The Anatomy of the GRID; Enabling Scalable Virtual Organisations; 2001

29. Kohl, J. and C. Neuman, "The Kerberos Network Authentication Service (V5)," RFC 1510, September 1993.

30. Dierks, T. and C. Allen, "The TLS Protocol, Version 1.0," RFC 2246, January 1999.

31. Arsenault, A. and S. Turner, "Internet X.509 Public Key Infrastructure, PKIX Roadmap," Internet Draft, March 10 2000.

32. Housley, R., "Cryptographic Message Syntax," RFC 2630, June 1999.

33. Linn, J., "Generic Security Service Application Program Interface,

Version 2, Update 1," RFC 2743, January 2000.

34. Wray, J., "Generic Security Service API Version 2, C-bindings," RFC 2744, January 2000.

35. Adams, C., "Independent Data Unit Protection Generic Security Service Application Program Interface (IDUP-GSS_API," RFC 2479, December 1998.

36. Владимир Галатенко, Информационная безопасность – обзор основных положений Открытые Системы # 3 1996

37. Владимир Галатенко, Информационная безопасность, Открытые Системы # 4 1996

38. Илья Трифаленков, Система информационной безопасности на основе ОС Solaris, Открытые Системы # 3 1996

39. Н. Вьюкова, Несанкционированный доступ – примеры вторжения, Открытые Системы # 1 1996

40. Беляев В.И. Безопасность в распределенных системах, Открытые Системы # 3 1995

41. Зегжда Д.П., А.М. Ивашко Основы информационной безопасности, Москва Горячая линия-Телеком 2000

42. J.Andreeva, S.Belov, I.Sidorova, E.Tikhonenko et at al., Dashboard for the LHC experimentsJ.Phys.Conf.Ser., 119:062008, 2008.

.