Міністерство освіти і науки України

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «КИЄВО-МОГИЛЯНСЬКА АКАДЕМІЯ»
Кафедра мультимедійних систем факультету інформатики

РОЗРОБКА ЧАТ-БОТУ ДЛЯ ПОШУКУ РЕЛЕВАНТНИХ ВІДПОВІДЕЙ ЗІ STACK OVERFLOW

Текстова частина до курсової роботи
за спеціальністю „Інженерія програмного забезпечення” 121

Керівник курсової роботи
с.в. _Борозенний С.О.__
(прізвище та ініціали)
_____
(підпис)
“____” _____ 2020 р.

Виконав студент _____
___Посвистак А.І._____
(прізвище та ініціали)
“____” _____ 2020 р.

Київ 2020

Міністерство освіти і науки України
НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «КИЄВО-МОГИЛЯНСЬКА АКАДЕМІЯ»

Міністерство освіти і науки України

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «КИЄВО-МОГИЛЯНСЬКА АКАДЕМІЯ»

Кафедра мультимедійних систем факультету інформатики

ЗАТВЕРДЖУЮ
Зав.кафедри мультимедійних систем,
доцент, к.ф-м.н.
_____О. П. Жежерун (підпис)
„____”_____2020 р.

ІНДИВІДУАЛЬНЕ ЗАВДАННЯ

на курсову роботу

студенту Посвистак Анастасії Ігорівні факультету інформатики 3-го курсу

ТЕМА Розробка чат-боту для пошуку релевантних відповідей зі Stack Overflow

Зміст ТЧ до курсової роботи:

Індивідуальне завдання
Календарний план виконання роботи
Анотація
Вступ
1 Аналіз задачі
2 Представлення слів
3 Метод
Результати
Висновки
Список використаних джерел

Дата видачі „___” _____ 2020 р. Борозенний О.С. _____
(підпис)

Завдання отримав _____ (підпис)

Тема: Розробка чат-боту для пошуку релевантних відповідей зі Stack Overflow

Календарний план виконання роботи:

№ п/п	Назва етапу дипломного проект (роботи)	Термін виконання етапу	Примітка
1.	Отримання завдання на курсову роботу	01.11.2019	
2.	Аналіз матеріалів за темою	14.01.2020	
3.	Розробка та програмування алгоритму	14.02.2020	
4.	Написання текстової частини до курсової роботи	01.04.2020	
5.	Коригування виконаної роботи	15.04.2020	
6.	Створення слайдів для доповіді та написання доповіді.	01.05.2020	
7.	Остаточне оформлення роботи та слайдів	07.05.2020	
8.	Захист курсової роботи	20.05.2020	

Посвистак А. І. _____

Борозенний О. С. _____

“_____”_____

Зміст

Анотація

Робота присвячена розробці діалогової системи, орієнтованої на вирішення питання. У даному випадку — це пошук релевантних відповідей на запит користувача зі Stack Overflow. Було розглянуто різні приклади використання обробки природної мови, проте більша частина роботи присвячена саме діалоговим системам: архітектурі, типам, характеристикам.

Окрім цього були розглянуті різні моделі для векторних представлень слів, їх переваги та недоліки.

Ключові слова: діалогові системи, обробка природної мови.

Вступ

Обробка природньої мови — це цікава та популярна проблема, особливо зараз, у 21 столітті, коли інформаційні технології розвиваються з шаленою швидкістю, а алгоритми та підходи до роботи з текстом стають потужнішими. Обробка застосовується в багатьох сферах. Зокрема, автоматичний переклад.

Один з найяскравіших прикладів — Google Translate[14], який використовує прямий переклад, тобто в режимі онлайн перекладає текст з будь-якої природної мови на іншу безпосередньо. А також Facebook[15], що використовує цю техніку для того, щоб перекладати текст в постах та коментарях, аби не виникало мовленнєвого бар'єру та користувачам було зручно слідкувати за іншими людьми з різних куточків світу.

Паралельно із цим прикладом стрімко розвивається ще один з напрямків використання обробки природньої мови — діалогові системи(ДС). ДС поділяються на автоматизовані голосові помічники та чат-боти. Тим самим їх можна охарактеризувати за наступними ознаками: цільові — ті, що мають певну мету й вирішують проблему користувачів, та нецільові, які створені з розважальною метою та імітують розмову з реальним співрозмовником.

Перші орієнтовані на коротку розмову, щоб після отримання інформації від користувача, система видала необхідну відповідь на його запит, тобто вони Task oriented. Деякі з таких «помічників» відомі на весь світ: Apple Siri[16], Amazon Alexa[17], Yandex Alisa[18]. Вони допомагають користувачам дістатися до певного пункту призначення, ввімкнути музику, керувати домашніми приладами тощо. І ще необхідно додати, що такі голосові помічники є основою для інтерфейсу роботів.

Другий вид зосереджений на розширеній розмові, іншими словами чат. Найперша подібна система — ELIZA[1] — віртуальний співрозмовник. Цей приклад демонструє використання чат-ботів у практичних цілях: ELIZA — це програма, яка імітує розмову з психотерапевтом, одночасно тестуючи техніку уважного слухання та надання порад. Вона належить до закритого типу діалогових систем, тому що вміє говорити тільки на тему психоаналізу, проте в той самий момент ELIZA не має певної задачі для розмови й завершить її тільки тоді, коли цього захоче користувач, тому з нею можна поспілкуватися.

Постановка завдання

Виходячи з актуальності проблеми на даний час, за мету роботи була поставлена розробка та реалізація чат-боту, який стане в нагоді тим, хто хоче зекономити свій час у пошуку необхідних питань на Stack Overflow[9].

Основна частина роботи складається з 3 розділів.

Перший розділ присвячено обробці природної мови та одному з її напрямків в області застосування нейромереж, глибинного навчання, який зараз стрімко розвивається, — діалогові системи. Наразі дослідження в цій області отримують назву розмовного інтелекту — чат-боти. Ця тема є актуальною через те, що все більше людей завантажують на свої смартфони різні месенджери, що стає вигідно різним компаніям задля просування своїх послуг.

Другий розділ містить інформацію про компоненти, які використовуються при обробці природної мови, зроблено їх аналіз.

У третьому розділі розглядаються методи, інструменти, представлені в курсовій роботі.

Основна частина

Розділ 1 : Обробка природної мови та напрямки її використання

1. 1. Обробка природної мови

Обробка природної мови (NLP — Natural Language Processing) — це технологія в області штучного інтелекту, яка описує взаємодію між людською мовою та комп'ютером. Її мета — прочитати та «зрозуміти» природну мову, щоб правильно обробити текст.

Обробка шалено застосовується в нашому житті кожного дня в різних проявах: ми надсилаємо в месенджерах голосові повідомлення, даємо завдання голосовим помічникам, застосовуємо Т9 (предиктивну систему набору тексту) при написанні текстових повідомлень та багато іншого.

З кожним днем задач стає все більше, а через те, що NLP настільки багатозадачна технологія й є можливість працювати з великими об'ємами даних, вона стає все більш популярної та розвивається шаленими темпами, а застосування їй знаходять в багатьох сферах.

Слід виокремити значущі приклади:

- Amazon Comprehend Medical використовують обробку природної мови[2], щоб отримувати дані про хвороби та лікування пацієнта з його амбулаторної картки, клінічних звітів та інших записів.
- Amazon Alexa, Apple Siri, Microsoft Cortana[19], Аліса Яндекс. Вони є голосовими помічниками та використовують NLP, щоб користувачеві було легко та швидко подзвонити, надіслати повідомлення, знайти дорогу до їхнього місця призначення, знайти

рецепт, ресторан або ж увімкнути прилади автоматизації будинку, іншими словами, Розумний дім.

- Google використовує NLP для їхнього сервісу Gmail[20]. Він вміє визначати яке повідомлення є спамом, щоб воно не потрапляло у вхідні поштової скриньки.

Один з найперспективніших напрямків в області застосування нейронних мереж та глибинного навчання, є діалогові системи. Через те що людська мова складна та заплутана, ця тема ще не є повністю дослідженою.

1. 2. Діалогові системи

Діалогові системи — це програми, створені для комунікації з користувачем через текст, мову, або те й інше одночасно. Вони мають певні характеристики та поділяються на два класи:

— діалогові системи, запрограмовані для вирішення певної проблеми, вони забезпечують користувача короткою розмовою

— діалогові системи, що імітують розмову з людиною та забезпечують користувача більш розширеною бесідою без певного сценарію

Якщо брати за мету генерацію відповідей, то діалогові системи можна поділити на такі типи:

— системи, які основані на правилах (rule-based). Відповідь на повідомлення користувача будується по шаблону чи правилу

— системи, які обирають відповіді на запитання користувача зі списку, підготованого раніше (retrieval-based)

— системи, які не мають заздалегідь підготовленого списку та основані на породжуючих моделях (generative)

Це можуть бути голосові помічники (task-oriented dialog agents) або чат-боти (chatbots). Головний компонент кожної діалогової системи — це діалоговий менеджер, який відповідає за 2 завдання:

1) Відслідковування стану, тобто він стежить за діалогом між користувачем і системою й змінює цей стан після кожного висловлювання користувача.

2) Контролювання вибору наступної дії. Ця частина приймає оцінку з першої частини й обирає найкращу дію діалогової системи від агента.

Наприклад, якщо користувач запитує систему: «Як краще мені дійти до кінотеатру, що знаходиться біля метро Льва Толстого?», діалогова система, проаналізувавши запит, має зробити запит до внутрішньої бази даних, результатом якого буде список усіх кінотеатрів. І по цьому списку вона має сформувати новий список з назвами фільмів, що йдуть у найближчих кінотеатрах за заданою адресою, та видати це користувачеві.

Ще дві не менш важливі частини — це NLU (Natural Language Understanding) та NLG (Natural Language Generation).

Перше — це основа для будь-якого чат-боту, модуль «розуміння» природної мови. NLU — це підмножина процесів NLP, це вміння комп'ютеру розуміти значення того, що подає на вхід користувач. Також можна сказати, що це вміння комп'ютеру правильно обробити неструктуровані дані й зробити з них таку структуру, що буде зрозуміла машині.

Друге — це процес перетворення структурованих даних в текст, що буде зрозумілий людині при її бесіді з чат-ботом. Тобто NLG — це те, що відбувається коли комп'ютер «пише» на природній мові.

Чат-боти замінюють людей. Яскравий приклад — чат-бот Зоряна від компанії «Київстар». Це віртуальний асистент, який спілкується з абонентами Київстар задля вирішення їх завдань або проблем. Майже за 2 роки вона провела

близько 2 мільйонів діалогів. Зоряна так добре навчилася, що 85% запитів[3] вирішуються без втручання операторів, тому що стандартні відповіді, що зберігаються в базі, задовольняють користувачів.

Ще один приклад системи, орієнтованої на вирішення певних питань, — голосовий помічник Аліса від Яндекс. Особливість цієї системи — це те, що Яндекс використовують не тільки заготовлені сценарії, а й deep learning, що робить Алісу гнучкішою на відповідь.

Як працює Аліса? Перш за все, те, що сказала людина, обробляється та перетворюється в текст, який сприймає система. Після цього він потрапляє в класифікатор інтентів(Intent classifier). Його задача — зрозуміти, що хотів сказати користувач.

Наступний крок — виділення основних частин з повідомлення користувача. Для кожного інтенту є свою модель, яка називається семантичний тегер. У даному випадку, цей тегер для погоди виділив слово «завтра» й зрозумів би, що користувачеві потрібно дізнатися погоду на завтра.

Йдучи по тексту, яке ми сформували з голосового повідомлення, ми структуруємо його й надалі будемо називати це фреймом. У ньому зберігається вся зібрана системою інформація: що інтент (get_weather) – погода, що нам потрібна ця інформація на завтра. Проте нам невідома локація, про яке саме місце потрібна погода.

Усе це потрапляє до Dialog Manager(DM), якому відомий стан діалогу та його контекст, тобто що відбувалося до й те, що відбувається зараз. За його основу прийнята концепція form-filling: своїми запитами користувач заповнює уявну віртуальну форму. Коли всі поля заповнені, користувач буде задоволений тим, що його запит виконаний. DM на вхід приймає фрейм та приймає рішення, що надсилати користувачеві. По геолокації користувача він може дізнатися, де

він знаходиться, та відправити йому інформацію про погоду в його місті на завтра, хоч і користувач це не вказував.

Після цього NLG генерує текст, який був сформований та містить опис погоди. Цей текст відправляється на модуль синтезу мови, що невдовзі промовить Аліса на запит людини.

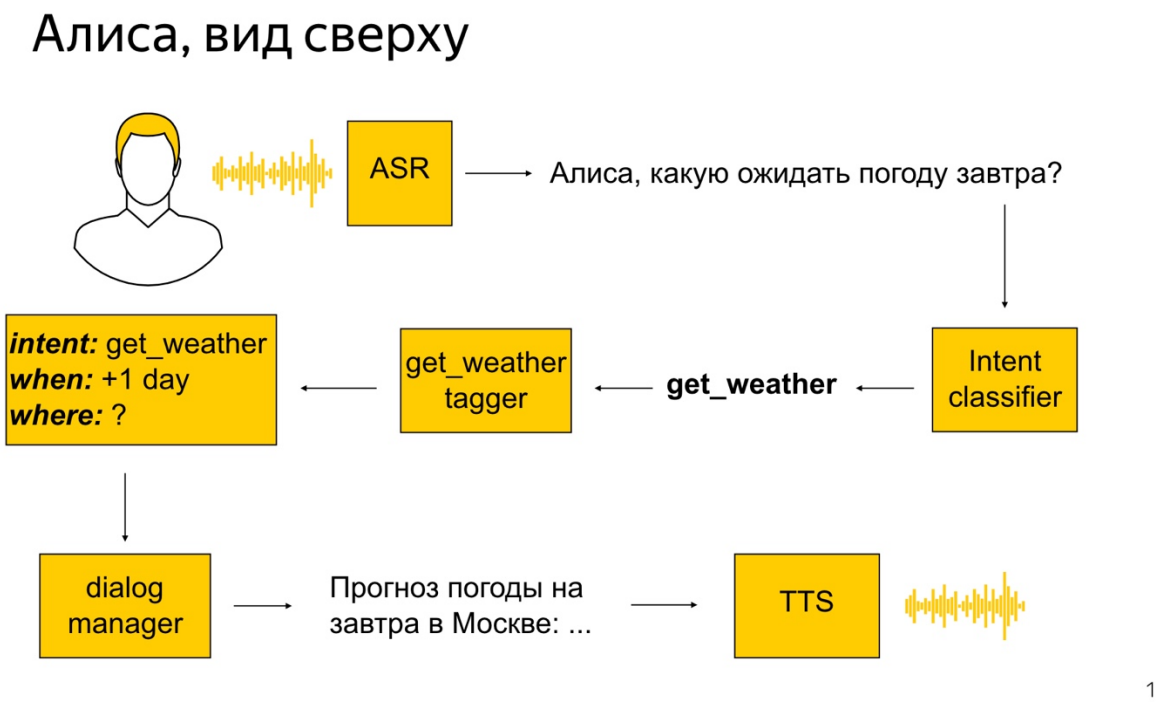

рис.1 Яндекс.Аліса [4]

Розділ 2 : Представлення слів

2. 1. Токенізація та векторизація

Існує такий метод обробки тексту як токенізація. Він використовується майже на самому початку, щоб розбити речення, абзац або текст на шматки, що називаються токенами. Слова, цифри, розділові знаки та інше можна вважати за токени. Цей процес також називається лексичним аналізом або сегментацією тексту.

Символи, по яким будується розбиття, залежать від того, який саме текст та що необхідно виділити, які символи потрібно залишити, тому що без них втратиться сенс речення, а які– видалити.

Наприклад, маємо речення «My dog was talking to me when my mum came.» Розбиваємо на токени й отримуємо: [‘My’, ‘dog’, ‘was’, ‘talking’, ‘to’, ‘me’, ‘when’, ‘my’, ‘mum’, ‘came’, ‘.’].

Наступний крок — нормалізація токенів. Одне слово має багато різних форм та закінчень, однокореневих слів, проте мета нормалізації — зведення токенів до початкової словникової форми.

Наприклад,

- leaves, leaf –> leaf
- talked, talks, talk –> talk
- dog, dog’s, dogs –> dog

Процес нормалізації слів(токенів) називається стемінг або лематизація. Це два випадки нормалізації, котрі суттєво відрізняються між собою.

Стемінг — евристичний процес видалення або змінення місцями суфіксів задля того, щоб дістатися до кореня слова, що називається стемом. Тобто цей

метод відрізає «зайве» від кореня слова, що може призвести до втрати важливих суфіксів.

Лематизація — більш точний та тонкий процес, що використовує словник та морфологічний аналіз задля приведення слова до його канонічної форми – – леми.

Приклади:

1. Слово bad є лемою для слова worse. У цьому випадку процес лематизації справиться, коли стемінг провалить завдання, тому що під час обробки слова не побачить зв’язку, тому що необхідно звірятися зі словником.
2. Зі словами із закінченням -ing справляться обидва процеси. Наприклад, talking стане talk, тому що це є базовою формою.

Проте слова із закінченням -ing можуть бути як іменником, так і дієсловом (формою to talk), залежно від контексту. У цьому випадку, лематизація буде опиратися на контекст, обираючи правильну лему, чого не можна сказати про стемінг.

Чому використовується стемінг, якщо він багато в чому поступається лематизації? По-перше, його можна застосовувати для того, щоб виправити орфографічні помилки в токенах. Стемінг простий у використанні та працює дуже швидко. Якщо швидкість та продуктивність ставляться за мету NLP моделі, то слід обирати стемінг як метод нормалізації слів.

Лематизація є більш ресурсномістким процесом. Також через те, що вона потребує більше знань природної мови, вимагається більша обчислювальна потужність, ніж під час стемінгу.

Після цього йде процес видалення стоп-слів. Суть стоп-слів полягає в тому, що це слова, які зустрічаються в нашому тексті з великою частотністю, а також не дають додаткової корисної інформації, але впливають на завдання NLP, оскільки вони трапляються дуже часто.

Якщо проаналізувати текст та порахувати кількість входжень кожного слова, the, is, are, not, and будуть згори списку. Дуже часто такі слова є найчастішими словами в будь-якому фрагменті тексту, і це так, тому що вони утворюють функціональний скелет будь-якого речення, що повідомляє граматичні зв'язки, які мають змістові матеріали.

Видалення стоп-слів — це один з етапів попередньої обробки слів під час якого з тексту видаляються слова, що не мають значення для завдань з природної мови.

Наступний необхідний крок — векторизація слів. Векторне представлення — це метод представлення рядків у вигляді векторів зі значеннями. Для цього процесу застосовуються різні спеціальні моделі, такі як: мішок слів(bag of words), n-грами, Word2Vec, GloVe.

1) **Bag of words** дозволяє підрахувати кількість входжень кожного слова в тексті. Метод створює певну матрицю для речення або документу, де на перетині рядка зі стовпцем буде число, яке відповідає за входження слова в документі або реченні.

Наприклад:

	it	was	an	awful	weather	they	will	…
It was an awful weather yesterday	1	1	1	1	1	0	0	…
If the weather is not awful, they will meet tomorrow	0	0	0	1	1	1	1	…

Недоліком є те, що втрачається порядок слів.

2) **n-грами** — комбінації з n послідовних термінів. 1-грама — це токен, 2-грама — це пара токенів і тд. У цьому випадку ми зберігаємо деякий порядок слів.

Наприклад:

	awful weather	weather	not	will meet	…
awful weather	1	1	0	0	…
not awful	0	0	1	0	…
will meet tomorrow	0	0	0	1	…

Проте існує певний недолік цього методу. Він в тому, що може існувати дуже багато комбінацій. Маючи велику кількість слів у документі, після розбиття, матриця стане великих розмірів через кількість n-грам.

TF-IDF — це покращений варіант мішка слів. Він враховує важливість слів, виходячи з його рідкості в документі, що в свою чергу є елементом колекції або корпусу.

1) TF(term frequency): $tf(t,d)$ — частота терміну (токену, n-грами) t в документі d.

Варіанти обрахунку:

Схема присвоєння вагових коефіцієнтів	Вага терміну
бінарний	0(не належить документу), 1(належить документу)
кількість рядків	$f_{t,d}$

частота входжень терміну	$f_{t,d} / \sum_{t` \in d} f_{t`,d}$
логарифмічна нормалізація	$log(f_{t,d})$

2) Також існує IDF(inverse document frequency) — обернена частота документа. Вага IDF вимірює значення слова в корпусі, надаючи словам, які часто трапляються, низьку вагу, і навпаки для тих, що трапляються рідко.

Нехай $N = |D|$ — загальна кількість документів у корпусі.

$|\{d \in D : t \in d\}|$ — кількість документів, де зустрічається термін t

$$idf(t, D) = log \frac{N}{|\{d \in D : t \in d\}|}$$

3) TF-IDF — це статистична міра для оцінки важливості слова в документі, який є частиною колекції або корпусу.

$$tfidf(t, d, D) = tf(t, d) * idf(t, D)$$

Висока вага в TF-IDF досягається тоді, коли маємо високу частоту терміну в документі та низьке значення IDF у всій колекції документів.

Інші методи будуть розглянуті в наступній частині розділу.

2. 2. Огляд моделі Word2Vec

Word2Vec — це набір алгоритмів, основаних на моделях Continuous Bag of Words (CBOW) та Skip-gram, що був створений групою дослідників Google на чолі з Томасом Міколову 2013 році[5,10]. Наразі алгоритми є досить популярними та використовуються в багатьох роботах NLP. Наприклад, цю технологію застосував сервіс Airbnb[6] у своїй рекомендаційній системі.

Сам Міколов вважає, що модель CBOW краще підходить для великих колекцій(корпусів) текстів, де містяться мільйони або мільярди слів та більше, тому що вона ліпше працює зі словами, які мають високу частотність. Коли Skip-gram краще застосовувати до невеликих корпусів текстів з менше ніж 100 мільйонів слів, тому що ця модель краще враховує рідкі слова та працює повільніше.

Як працюють ці два алгоритми:

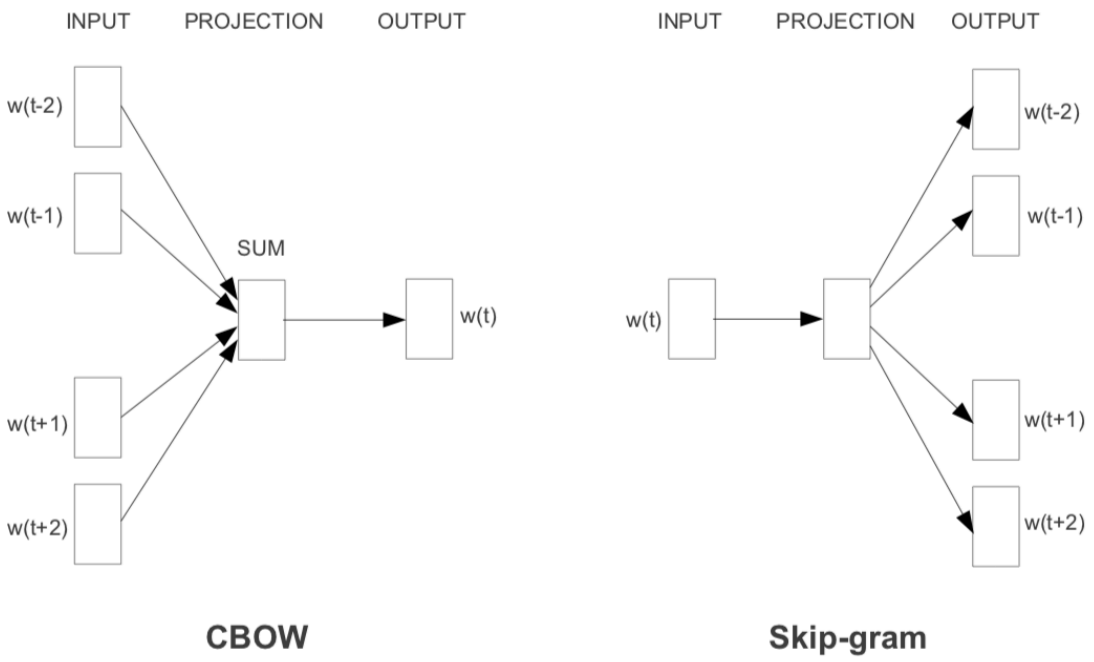

рис. 2 Алгоритми CBOW та Skip-gram[10]

В алгоритмі CBOW моделі на вхід (input) подаються слова $(w(t-2), w(t-1), w(t+1), w(t+2))$, що стоять поруч із цільовим словом $(w(t))$ і задача — видати це слово.

У Skip-gram — навпаки: на вхід приходить поточне слово ($w(t)$), а отримуємо те, що стоїть поруч з ним, тобто його контекст ($w(t-2), w(t-1), w(t+1), w(t+2)$).

Як було написано раніше, слова-вектори — це числові представлення слів, де зберігається семантичний зв’язок між ними.

Word2Vec намагається мінімізувати відстань між семантично близькими словами. Тобто він розміщує слова-вектори на площі ознак таким чином, що розміщення визначається їхніми значеннями. Слова, які мають схожі значення розташовані ближче один до одного, згруповані разом.

Розглядаючи граф чоловік-жінка, зображений на рис. 3, можна помітити, що відстань між чоловіком і жінкою така ж сама, як відстань між королем (чоловіком) і королевою (жінкою). Відстань між королевою і жінкою та відстань між королем і чоловіком однакові (король і чоловік, королева і жінка являють собою однакове порівняння статі, отже, вони повинні мати однакові відстані між собою)[7].

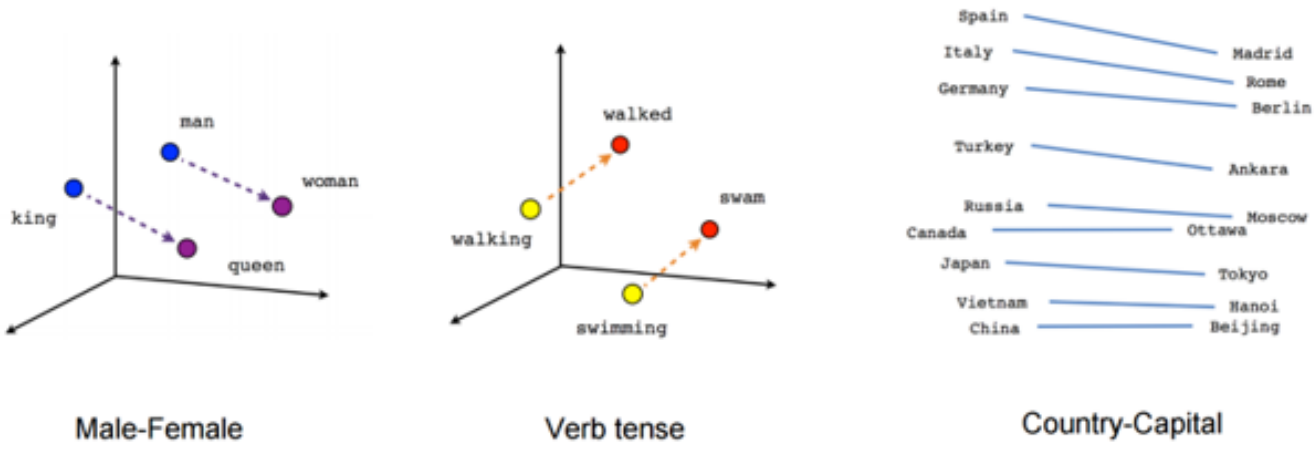

рис. 3 Слова-вектори[7]

StarSpace — ще одна модель для генерації представлень слів у вигляді векторів, розроблена компанією Facebook[8, 12], яка була використана в цій роботі.

Це бібліотека загального використання, яку можна використовувати як для векторизації слів, так і для речень, документів. Часто використовують для колаборативної фільтрації.

Вона має різні стани (trainMode) задля вирішення різних задач. У цій роботі trainMode = 3, тобто метою є вивчення подібності між реченнями (у нашому випадку питаннями). Тренування полягає d поданні на вхід триплетів — два схожих між собою речення і одне випадкове. Цей підхід дозволяє вивчити більш якісні векторні репрезентації завдяки навчанню з учителем на противагу Word2Vec.

Розділ 3 : Метод

3. 1. Інструменти та бібліотеки, використані в роботі

Мова написання роботи — Python, так як він найкраще підходить для роботи з текстом, а саме NLP. Більшість моделей та алгоритмів підтримуються цією мовою, що є гарною перевагою у використанні.

Етапи в роботі(рис. 3.1):

1) Користувач надсилає своє питання боту

2) Попередня обробка тексту:

- Зведення слів до нижнього регістру.
- Видалення пунктуації ([/(){}\[\]\|@,;), замінюючи їх на пробіл.
- Видалення «поганих» символів. Символи, що не входять у список (0-9a-z #+_), будуть замінені на пробіл.
- Видалення стоп-слів, що взяті з бібліотеки NLTK[13]

3) Intent classifier: визначення типу запиту користувача.

Для цього були використані відкриті дані[21] — два датасети: питання зі Stack Overflow та загальні фрази для підтримання діалогу. На їх базі розроблено класифікатор, який визначає мету запиту користувача: загальне повідомлення чи технічне питання на Stack Overflow.

Через великий дисбаланс класів була взята метрика ROC – AUC(receiver operating characteristic – area under the curve). На тестовому наборі даних досягнуто 0.99 за цільовою метриковою, що каже про гарне розбиття між класами.

4) Programming Language classifier: визначення мови програмування в запиті.

Через те, що датасет з питаннями великих масштабів, було додано класифікатор, що розпізнає тег питання, щоб оптимізувати систему.

Якщо запит користувача класифіковано як технічний, то він передається до класифікатора мови програмування(Programming Language classifier). Цей етап був доданий задля поліпшення швидкості роботи системи.

А також на цьому етапі було проведено аналіз і на тестовому наборі даних досягнуто 0.81 за F1-метрикою.

5) Stack Overflow Assistant

Формується повідомлення користувачеві, тобто відшукується правильна відповідь на його запит — посилання з питанням на Stack Overflow. Отримавши мову програмування до запиту, можна дістати найбільш релевантне питання, знайшовши питання з найменшою відстанню до векторизованого запиту. Була використана косинусна метрика(рис.5).

6) Result

Надсилається повідомлення у вигляді посилання на питання Stack Overflow.

Якщо ж Intent classifier визначив, що повідомлення є загального типу, то користувачеві в результаті буде надіслана заготована відповідь.

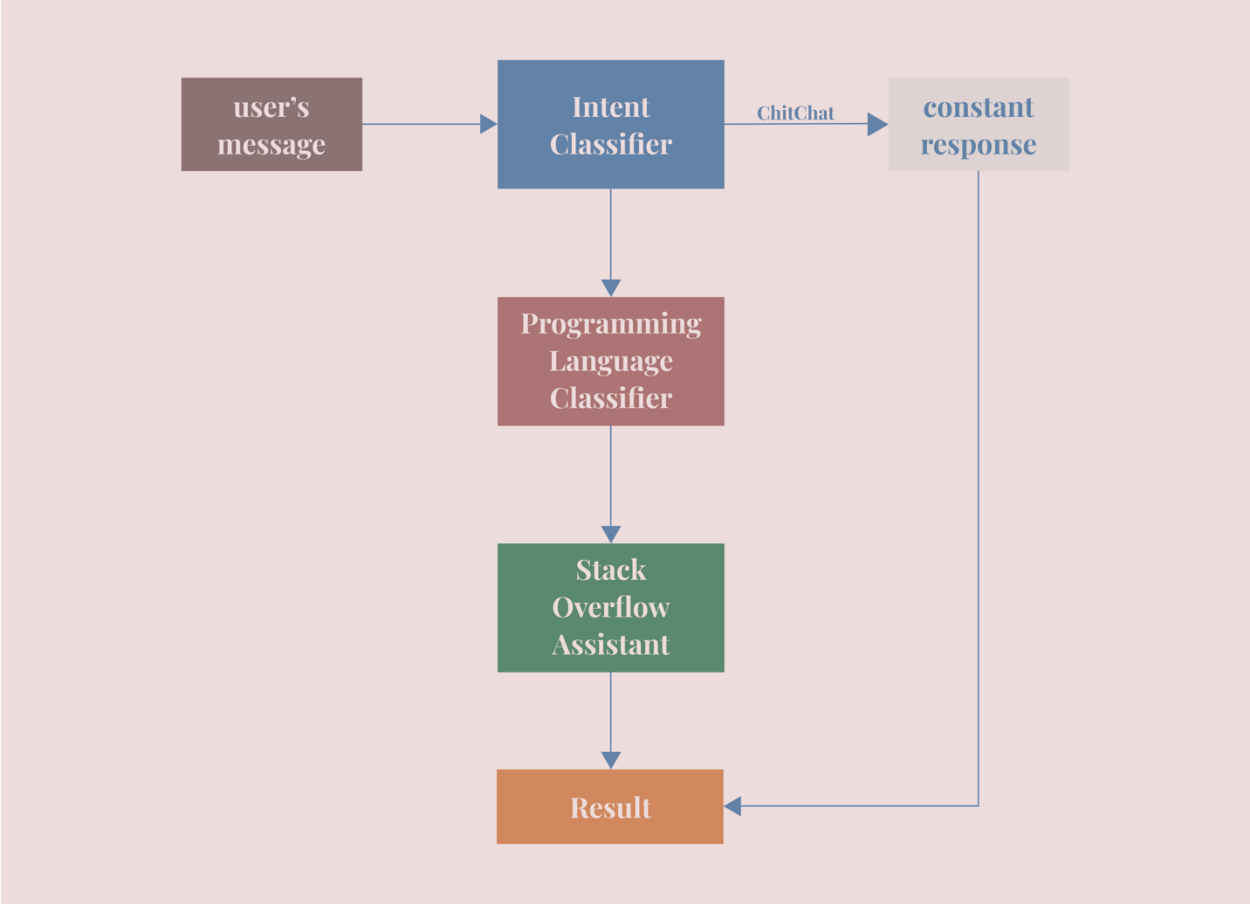

рис. 4 Схема роботи програми: як обробляється запит користувача

3. 2. Огляд класів та методів

Усього робота налічує 4 класи: IntentClassifier, TagClassifier, TechnicalClassifier, DialogueManager.

1) IntentClassifier

На вхід приймає запит користувача та розпізнає до якого інтенту він належить: загального чи технічного. Була використана модель TF-IDF, описана в другому розділі, для векторизації. Після цього векторизована матриця була далі використані в якості ознак для логістичної регресії.

2) TagClassifier

Визначає номер тегу мови програмування по запиту.

3) TechnicalClassifier

Визначає найбільш релеванте питання за StarSpace представленням по класифікованній мові програмування та видає це питання.

Був створений словник з усіма представленнями. Ключ — унікальний тег, що відповідає за певну мову програмування, а значення —питання по ньому, все це векторизовано за допомогою StarSpace.

```
def get_nearest(self, embedding, embeddings):
    index_min = pairwise_distances_argmin(embedding, embeddings, metric='cosine')
    return index_min[0]
```

рис. 5 Пошук найближчих слів-векторів

4) DialogueManager

Усе перелічене, використовується в цьому класі. Уся логіка роботи прописана в DialogueManager.

```python
class DialogueManager():

    def __init__(self):
        self.clf_intent = IntentClassifier().fit()
        self.clf_tag = TagClassifier().fit()
        self.clf_query = TechnicalClassifier().fit()
        self.chitchat_message = 'I am not built to chitchat with you. Better ask me some technical stuff.'

    def preprocess(self, query):
        query = text_prepare(query)
        intent = self.get_intent(query)
        intent_name = 'chitchat' if intent == 0 else 'tech'
        if intent == 0:
            response = {'intent':intent_name,
                        'text':self.chitchat_message}
            return response

        tag = self.get_tag(query)

        response = self.get_response(query, tag)

        response = {
            'intent':intent_name,
            'text':response[0],
            'post_id':response[1],
            'language':tag
        }
        return response

    def get_intent(self, query):
        intent = self.clf_intent.predict(query)
        return intent

    def get_tag(self, query):
        tag = self.clf_tag.predict(query)
        return tag

    def get_response(self, query, tag):
        response = self.clf_query.predict(query, tag)
        return response
```

рис. 6 Клас Dialogue Manager

Результати

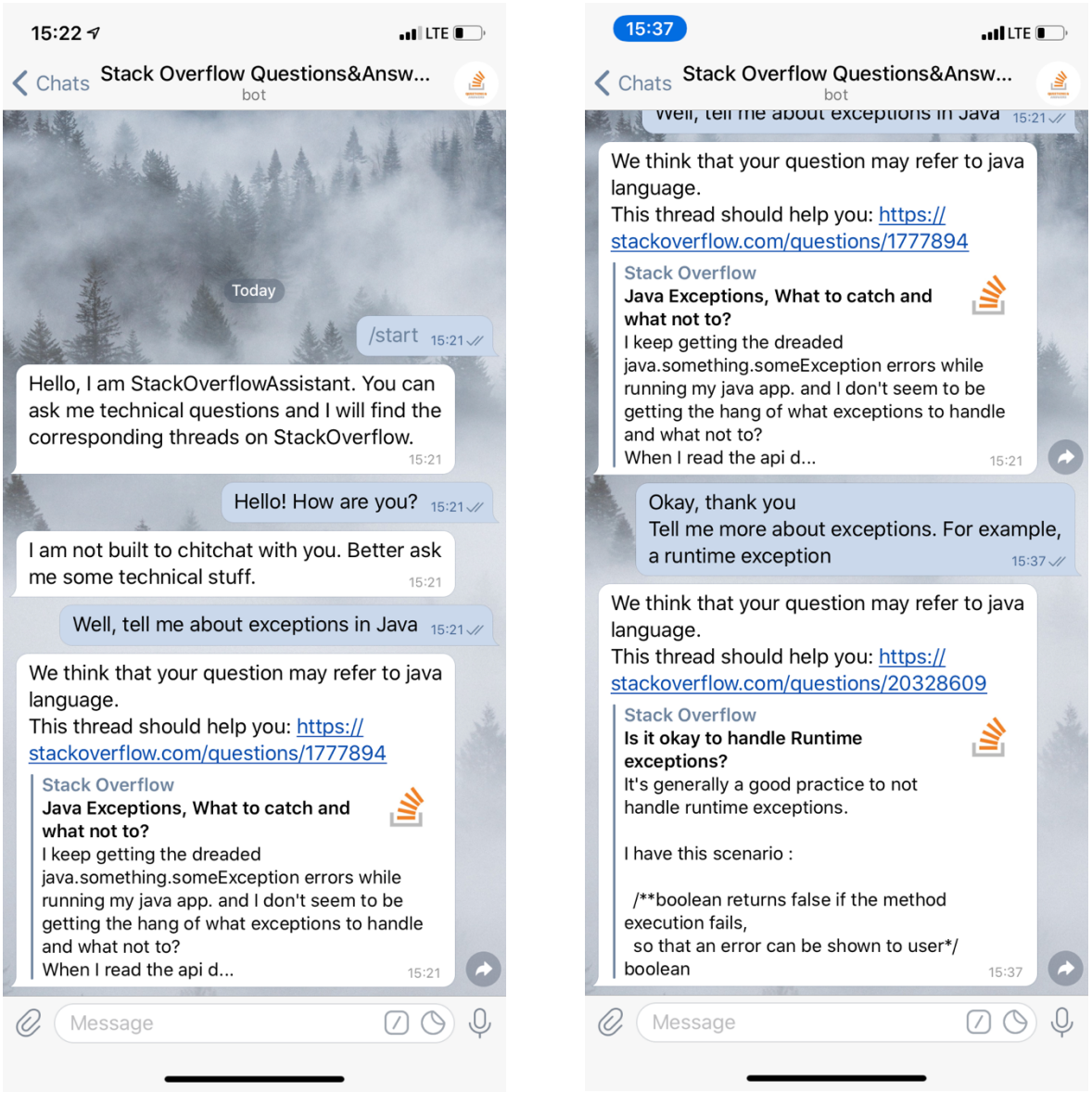

рис. 7, 8 Приклади роботи бота

Результатом роботи є бот, який розпізнає питання та відповідає найбільш релеватним постом зі Stack Overflow. Якщо користувач напише йому повідомлення загального типу, то отримає у відповідь "I am not built to chitchat

with you. Better ask me some technical stuff." Проте далі можна вести з ним розмову та продовжувати ставити технічні питання.

Як можна побачити на рис.7, 8, є можливість у вигляді звичайного розмовного повідомлення надіслати свій запит боту, а він правильно обробить його та відповість релеватним питанням.

Список прийнятих скорочень

ДС — Діалогові системи

NLP — Natural Language Processing — Обробка природної мови

NLU — Natural Language Understanding

NLG — Natural Language Generation

TF — Term frequency — Частоту терміну

IDF — Inverse document frequency — Обернена частота документа

CBOW — Continuous Bag of Words

NLTK — Natural Language Toolkit

Висновок

У цій роботі було розглянуто основні поняття та методи для роботи з обробкою природної мови. Результатом є створення чат-боту, який допомагає користувачу знаходити відповідь на його запит на Stack Overflow. Він розпізнає питання користувача, шукає найрелевантніше серед даних та видає відповідь.

Було взято два датасети: з діалогами загального типу та питаннями зі Stack Overflow. На тестовому наборі даних досягнуто 0.99 за цільовою метриковою, що каже про гарне розбиття між класами. Саме завдяки цьому бот досконало класифікує повідомлення та визначає мету запиту користувача: загальне повідомлення чи технічне питання на Stack Overflow.

Список використаних джерел

[1] Weizenbaum, J. (1966). ELIZA – A computer program for the study of natural language communication between man and machine. Communications of the ACM, 9(1), 36–45.

[2]https://www.thenewsminute.com/article/tech-giants-india-join-ai-bandwagon-focus-healthcare-93833

https://habr.com/ru/company/Voximplant/blog/446738/

[3]https://kyivstar.ua/uk/mm/news-and-promotions/chat-bot-zoryana-vid-kyyivstar-2-mln-dialogiv-12-tysyach-vidpovidey-v-bazi-i

[4] https://habr.com/ru/company/yandex/blog/349372/

[5]https://papers.nips.cc/paper/5021-distributed-representations-of-words-and-phrases-and-their-compositionality.pdf

[6]https://www.kdd.org/kdd2018/accepted-papers/view/real-time-personalization-using-embeddings-for-search-ranking-at-airbnb

[7]https://medium.com/@paritosh_30025/natural-language-processing-text-data-vectorization-af2520529cf7

[8] Wu, Ledell Yu, et al. “Starspace: Embed all the things!.” Thirty-Second AAAI Conference on Artificial Intelligence. 2018.

[9] https://stackoverflow.com

[10] https://arxiv.org/pdf/1301.3781.pdf

[11] https://nlp.stanford.edu/pubs/glove.pdf

[12] https://github.com/facebookresearch/StarSpace

[13] https://www.nltk.org

[14] https://www.translate.google.com

[15] https://www.facebook.com

[16] https://www.apple.com/siri/

[17] https://www.amazon.com/b?ie=UTF8&node=17934671011

[18] https://yandex.ru/alice

[19] https://www.microsoft.com/en-us/cortana

[20] https://gmail.com

[21] https://www.kaggle.com/stackoverflow/stackoverflow