

Шифр «ЕРІК»

**Моделювання намірів користувача
в усному діалозі між людиною та мобільним роботом**

2021 рік

ЗМІСТ

ВСТУП	2
РОЗДІЛ 1. ФУНКЦІОНАЛ РОБОТА ERIC’А ТА СТВОРЕННЯ АКУСТИЧНОГО КОРПУСУ	4
1.1. Особливості робота ERIC’а як системи усного діалогу	5
1.2. Типи смислів, типи речень, наміри користувача	7
1.3. Створення акустичного корпусу для мобільного робота ERIC’а	9
1.3.1. Передумови створення акустичного корпусу	9
1.3.2. Запис голосів дикторів та опрацювання звукових файлів	12
ВИСНОВКИ ДО РОЗДІЛУ 1	13
РОЗДІЛ 2. СТВОРЕННЯ ТА ТРЕНУВАННЯ КЛАСИФІКАТОРА НАМІРІВ КОРИСТУВАЧА	15
2.1. Постановка задачі для класифікатора та опис інструментарію	15
2.2. Підготовка даних для подальшого тренування	16
2.3. Тренування моделей	19
2.4. Метрики, що використовуються для оцінки моделей	21
2.5. Результати: оцінка моделей та висновки	22
ВИСНОВКИ ДО РОЗДІЛУ 2	25
ВИСНОВКИ	27
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ	29
ДОДАТКИ	31
Додаток 1	31
Додаток 2	32
Додаток 3	34
АНОТАЦІЯ	36

ВСТУП

XXI століття, а особливо початок його третьої декади, можна охарактеризувати тим, що роботи виконують значний відсоток роботи, яку раніше виконували люди. Змінились технології та змінилась психологія людей: тепер вони автоматизують навіть те, що, здавалося б, автоматизувати не можна, щоб економити дорогоцінний час та ресурси. Сервісна робототехніка – це клас роботів із різноманітними конструктивними рішеннями і принципами реалізації системи управління, що динамічно розвивається. Згідно з класифікацією Міжнародної федерації робототехніки [1], сервісні роботи призначені для виконання корисних для людини професійних і побутових задач. Ми оточені сервісною робототехнікою настільки, що домогосподарки тепер не миють підлогу, а вмикають робот-пилосос, а сучасні діти віком п'яти-шести років уже відвідують заняття на дитячих курсах з робототехніки.

Надзвичайно популярними є роботи, які виконують свою роботу автономно — без втручання людини. Такі роботи можуть працювати навіть у невизначених динамічних обставинах – у цьому полягає їх “інтелектуальність”. З'являється проблема розвитку технологій усного діалогу між людиною та машиною.

У світі уже представлена велика кількість систем, що демонструють інтелектуальне розуміння природної мови. Яскравими представниками діалогових систем є такі голосові помічники як Siri (Apple), Alexa (Amazon), Cortana (Microsoft), Google Assistant [2]. Науково-фантастичні фільми прогнозують таке майбутнє, де системи усного діалогу будуть настільки інтелектуально довершеними, що замінять спілкування з людиною. Але реальний стан технологій усного діалогу говорить нам про те, що до такого майбутнього ще далеко, а отже, є над чим працювати. Саме активним розвитком галузі мовленнєвих інформаційних технологій, робототехніки та штучного інтелекту (як у світі, так і в Україні) зумовлена **актуальність** цієї роботи.

Мета роботи – розробити мовленнєвий корпус та створити класифікатор намірів користувача для моделювання усного спілкування людини з роботом. Нашою задачею є створення моделі всіх можливих речень, що входять у діалог та виражають один і той самий зміст, моделювання параметрів слів у типах речень, генерація та пошук найбільш правдоподібних еталонних сигналів та їх параметрів. Досягнення мети передбачає виконання таких **завдань**:

1. Розглянути функціонал конкретного мобільного сервісного робота;
2. Розширити таблицю команд для мобільного сервісного робота, а саме збільшити варіативність типів речень;
3. Розробити акустичний корпус команд трьома мовами – українською, російською, англійською.
4. Створити та оцінити моделі-класифікатори намірів людини щодо робота.

Об’єктом дослідження є усний діалог людини з мобільним роботом. **Предметом** – наміри користувача як сторони діалогу. **Практичне значення** цієї роботи полягає в тому, що розроблений акустичний корпус та класифікатор будуть використовуватись для розширення мовного функціоналу робота та спрощення взаємодії між роботом і людиною. **Теоретичне значення** роботи – це розвиток методів і алгоритмів моделювання інтелектуальної (свідомої та підсвідомої) діяльності людини. **Матеріалом дослідження** є попередньо створена таблиця команд для сервісного робота та записи голосів дикторів. **Методи** дослідження: експериментально-фонетичний метод (використовувався для створення, сегментації та анотації акустичного корпусу), метод машинного навчання, метод статистичного аналізу та метод аналізу через синтез (були використані для створення моделі з’ясування намірів користувача мобільного робота).

РОЗДІЛ 1. ФУНКЦІОНАЛ РОБОТА ERIC'А ТА СТВОРЕННЯ АКУСТИЧНОГО КОРПУСУ

Унікальним є робот, створений у Міжнародному науково-навчальному центрі інформаційних технологій та систем Національної академії наук України та Міністерства освіти та науки України, — лабораторний двоколісний in-door мобільний робот ERIC (Evolution Robotics & International Center), зібраний з елементів робототехнічного конструкторського набору розробки Evolution Robotics (США) [3, 4].

Робота над ERIC'ом, що була проведена українськими науковцями, присвячена одній із задач візуальної навігації мобільного робота. Представлено практичний підхід до реалізації автономного виконання роботом завдання стеження за рухомими об'єктами за результатами його розпізнавання поточного зображення бортової камери. У робота є база знань, у якій закладені декларативна та процедурна частини. У системі управління до них приєднані вербальні мітки, тобто імена (назви об'єктів, дій тощо).

Для забезпечення функціонування робота використовуються два комп'ютери за аналогією до двох півкуль головного мозку. В одному знаходяться всі математичні моделі всіх його функціональностей (що таке рух, що таке відчуття сенсорів та ін.). Другий комп'ютер приймає рішення щодо виконання задач. Це – диспетчер, який діє в ситуації відповідно до типів ситуацій та типів невизначеності. На борту робота знаходиться перший комп'ютер (ноутбук). До нього підключені фізичні пристрої, а другий ноутбук працює з фізичними сигналами (фільтрація, інтерпретація).

Прикладні задачі такого робота досить великі. Він може виконувати функції нагляду за приміщеннями (як система відеоспостереження), мобільного електронного секретаря, чергового в нічні зміни, а також домашнього помічника тощо.

Наразі розробка використовується в освітніх цілях в Академії наук України.

На думку науковців, що створили робота, їм вдалося реалізувати ідею інтелектуального управління: подібно людині система в різних ситуаціях, у тому числі позаштатних, приймає "розумне" рішення [3].

1.1. Особливості робота ERIC'а як системи усного діалогу

Робот ERIC сприймає голосові команди. Завдання йому можна надавати як безпосередньо (через мікрофон), так і в дистанційному режимі (через мережу Інтернет). Акустичний сигнал розпізнається за допомогою розпізнавача мовлення Dragon Dictate, який випустила компанія Dragon Systems у 1992 році. Розпізнавач чудово навчається з прив'язкою до диктора. Головне — зв'язати акустичний образ, який робот отримує на вхід, з його відповіддю на ці запити чи команди.

У робота немає семантичного аналізатора, тож він не може "розуміти" смисли. Натомість він має семантичну модель світу.

Семантична модель світу з відповідною моделлю мови людино-машинного інтерфейсу для одного типу технічних систем часто не може бути застосована для інших систем. Для кожного автономного робота існує не реальний зовнішній світ, а його відображення у вигляді моделі світу, дійсною тільки для даного типу систем [3].

Роботи, оснащені різними апаратно-програмними засобами сприйняття зовнішнього світу і інтерпретації одержаної інформації про світ, сприймають світ по-різному. Понятійна множина конкретного автономного мобільного робота з відповідними семантичними складовими елементами словника людино-машинного інтерфейсу спирається на сукупність програмних структур верхнього рівня керуючої системи робота, що описують просторову модель сприйманого зовнішнього світу, модель власного пристрою і поточного стану робота, відповідний словник імен об'єктів і їх характеристик, які сприймаються

бортовою сенсорною підсистемою, бібліотеку формалізованих правил виконання певних цілеспрямованих дій робота з відповідними іменами можливих дій мобільного робота і їх параметрів [6].

Суб'єктивність світосприйняття мобільного робота пов'язана з конкретним набором органів сприйняття і їх чутливістю, методами аналізу отриманої інформації та побудови причинно-наслідкових зв'язків, способами організації пам'яті та видобування з неї необхідних даних, а також з репертуаром доступних для виконання дій.

Мобільні роботи з інтелектуальним керуванням здатні автономно виконувати завдання користувача, сформульовані у вигляді висловлювань із досить загальним змістом. У підсистемі людино-машинного інтерфейсу, що відповідає за ведення діалогу, істотну роль відіграє перевірка відповідності бажаного цільового стану робота, що міститься у висловленні користувача, можливостям його керуючої системи. Інакше кажучи, модуль розпізнавання і смислової інтерпретації мови користувача повинен постійно взаємодіяти з понятійною множиною керуючої системи у вигляді семантичної моделі світу робота. Послідовність уточнюючих питань, що генеруються підсистемою ведення діалогу для мінімізації розбіжностей в розумінні користувачем і керуючою системою робота сформульованого завдання, повинна приводити до повної визначеності всіх параметрів, що однозначно описують цільовий стан робота.

Робот може лише виокремлювати слоти з розпізнаних команд, так як у його базі знань містяться таблиці з фіксованою кількістю слотів. Зробимо висновок, що «мовні навички» робота дозволяють нам класифікувати його як просту діалогову систему на основі фреймів.

Зараз робот “володіє” трьома мовами – українською, російською та англійською.

В ERIC'а є фіксований набір фраз, які він розпізнає. Робот виконує велику кількість завдань. Наприклад, аналізує ситуацію в режимі реального

часу, відтворюючи при цьому просторову картину. Якщо виникає якась позаштатна ситуація, наприклад, з'являється звук, що відрізняється від акустичного фону, то робот прямує до джерела шуму та показує його через камеру.

Коли ERIC отримує команду, яка може з багатьох причин здаватись йому недостатньо детермінованою (некоректно поставлене завдання чи запитання, з'явилися перепони на його шляху до реалізації задачі тощо), він починає ставити уточнюючі питання для того, щоб заповнити всі необхідні, але на той час незаповнені слоти смислів, та розпочати виконання завдання. Також це відбувається, коли слова, які вимовив диктор, не входять в лексикон робота. У такому випадку він запитує щось на кшталт: «А що Ви мали на увазі?».

Його відповіді, навпаки, не є фіксованими, адже фізично важко завчасно передбачити і записати речення, у яких відображені ситуації, що в них потрапляє робот. Ситуацій є необмежена кількість, тому відповіді ERIC'а генеруються шляхом конкатенації слотів. Наприклад, *ім'я + дія + властивість + напрямок дії* і т.д. Згенеровані конкатенативно речення озвучуються синтезатором мовлення.

1.2. Типи смислів, типи речень, наміри користувача

Для розпізнавання інтентів (намірів) користувача щодо робота дуже важливою є проблема моделювання параметрів у реченнях. Це значить, що потрібно передбачити і врахувати всі можливі варіанти значень для певного речення.

У науковій статті, присвяченій параметризації типів речень, Яценко В. використовує таку термінологію: «Кожна ПО складається зі скінченної множини типів змістів (ТЗ). В кожен ТЗ входить множина еквівалентно змістових типів речень (ТР). Тип речення – це конструкція, що економно задає множину речень, отриманих з одного речення незалежними допустимою

заміною та допустимою перестановкою чи випадінням слів та словосполученнях” [7, с. 136].

База даних та знань є інформаційною моделлю предметної області [8].

Наприклад, коли користувач просить ERIC’а взяти певний предмет, то параметром є назва предмета, оскільки існує скінченна множина предметів, які він може схопити.

У справі із пошуком найбільш правдоподібних еталонних сигналів лінгвіст має врахувати всі обмеження на послідовність слів у реченні, хоча слов’янські мови не є мовами з прямим порядком слів (українська та російська для нашого дослідження).

Для специфікації обмежень на допустимі послідовності слів у фразах використовувалися LISP-структури, що відповідають типам речень [9], [10]. На основі цих структур генерується величезна кількість речень, що мають один і той самий зміст з точністю до параметрів.

Наприклад, така LISP-структура дає можливість вмістити в одній структурі аж 48 речень (Ілюстрація 1):

$$\left(\left(\begin{array}{c} \text{подоїди} \\ \text{переместись} \end{array} \right) \left(\begin{array}{c} \text{пожалуйста} \\ * \end{array} \right) \left(\left[\begin{array}{c} \text{объекту} \\ \text{предмету} \end{array} \right] (obj_n) \right) \right)$$

Ілюстрація 1. LISP-структура, що формує 48 речень [11]

Діалог між користувачем та мобільним роботом є почерговим заповненням типів речень або ж поступовим уточненням інтентів першої сторони у другій.

Для генерування фраз і уточнень як людини, так і робота пропонується використовувати єдину базу даних і знань. Характерною відмінністю усного діалогу від голосового управління пристроєм є те, що мобільний робот сам формує уточнення з урахуванням змін контрольованого ним середовища, наприклад, тривимірної сцени [11].

1.3. Створення акустичного корпусу для мобільного робота ERIC'a

1.3.1. Передумови створення акустичного корпусу

Для створення акустичного корпусу для моделювання діалогу було використано таблицю типів смислу, яку надав Відділ розпізнавання та синтезу звукових образів Міжнародного науково-навчального центру інформаційних технологій та систем НАН України та МОН України.

Надана таблиця інтентів (типів змістів, намірів) та типів речень містить:

- 30 інтентів;
- 47 LISP-структур російською мовою;
- 42 LISP-структури англійською мовою;
- 48 LISP-структур українською мовою.

Таблиця типів смислу для Еріка			
name	rus	eng	ukr
take_obj	возьми объект \$mobj_n [\$swobj_n] [\$direction]	take [the object] \$mobj_n [\$swobj_n] [\$direction]	взьми об'єкт \$mobj_n [\$swobj_n] [\$direction]
	возьми \$mobj_g [\$swobj_g] [\$direction]	take \$mobj_g [\$swobj_g] [\$direction]	взьми \$mobj_g [\$swobj_g] [\$direction]
come2obj	подойди к объекту \$obj_n [\$swobj_n] [\$direction]	come to the object \$obj_n [\$swobj_n] [\$direction]	підійти до об'єкта \$obj_n [\$swobj_n] [\$direction]
	подойди к \$obj_d [\$swobj_n] [\$direction]	come to \$obj_d [\$swobj_d] [\$direction]	підійти до \$obj_g [\$swobj_g] [\$direction]
analyse_obj	исследуй [объекты] [\$obj_n] [\$swobj_n] [\$direction]	analyse the object [\$obj_n] [\$swobj_n] [\$direction]	проаналізуй [об'єкти] [\$obj_n] [\$swobj_n] [\$direction]
exam_obj	изучи объект [\$obj_n] [\$swobj_n] [\$direction]	examine the object [\$obj_n] [\$swobj_n] [\$direction]	доследи об'єкти [\$obj_n] [\$swobj_n] [\$direction]
find_obj	найди объект [\$obj_n] [\$direction] [\$swobj_n]	find the object [\$obj_n] [\$direction] [\$swobj_n]	знайди об'єкти [\$obj_n] [\$direction] [\$swobj_n]
	найди [\$obj_g] [\$direction] [\$swobj_g]	find [\$obj_g] [\$direction] [\$swobj_g]	знайди об'єкти [\$obj_g] [\$direction] [\$swobj_g]
find_mark	найди маркеры \$marker	find landmarks \$marker	знайди маркери \$marker
bring_obj	принеси объект \$mobj_n [\$position] [\$mobj_n]	bring the object \$mobj_n [\$position] [\$mobj_n]	принести об'єкт \$mobj_n [\$position] [\$mobj_n]
	принеси [\$mobj_n] [\$position] [\$mobj_n]	bring [\$mobj_n] [\$position] [\$mobj_n]	принести [\$mobj_n] [\$position] [\$mobj_n]
cons_obj	рассмотри [объект] [\$obj_n]	consider [the object] [\$obj_n]	роздивися [об'єкт] [\$obj_n]
sort_obj	рассортируй [объекты] [\$obj_n]	sort [the objects] [\$obj_n]	посортуй [об'єкти] [\$obj_n]
follo_obj	преследуй объект \$mobj	follow the object \$mobj	переслідуй об'єкт \$mobj
		pursue the object \$mobj	стеж за об'єктом \$mobj

Ілюстрація 2. Фрагмент наданої таблиці інтентів та типів речень

Розширена автором роботи таблиця інтентів містить:

- 30 інтентів (кількість збережена);
- 96 LISP-структур російською мовою;
- 108 LISP-структур англійською мовою;
- 250 LISP-структур українською мовою.

Також для створення корпусу необхідно було розробити таблицю метаслів, тобто слотів, якими заповнювались параметри типів речень.

“Перейди вперед”. Мовленнєвий сигнал надходить у модуль розпізнавання та інтерпретації мовлення, у результаті чого отримуємо відповідь розпізнавання та інтерпретації в вигляді імені типу смислу *name* та розпізнаного тексту. У нашому випадку ця змінна приймає значення “navigate” (*name* = “navigate”). Система звертається до бази даних і знань, де в таблиці типів смислу знаходить відповідне представлення типу речення: “Перейди \$where [\$valid]” (Таблиця 1). Цей вираз містить метаслова, подані у вигляді змінних \$where та \$valid. Наявність метаслів вимагає уточнення (якщо фраза не була вимовлена або розпізнана повністю). Слід зауважити, що метаслова, подані у квадратних дужках, можуть бути відсутні, тому уточнення не вимагають.

MID	MT ua	MT eng	MT ru
take_obj	візьми [об’єкт] \$mobj [\$wobj] [\$direction]		
come2obj	підійди [до об’єкту] \$obj [\$wobj] [\$direction]		
	...		
navigate	перейди \$where [\$valid]		

Таблиця 1. Таблиця типів смислу для технічної системи [12, с. 3]

Система, знайшовши метаслово \$where, звертається до таблиці питань до метаслів та, в разі відсутності необхідних метаслів, ставить уточнююче питання відповідно до Таблиці 2. Уявімо, що користувач не вказав напрямок пересування. Тоді система усного діалогу повинна запитати “куди?” або “в якому напрямку?”. Користувач має дати коректну відповідь на це питання, наприклад, сказати: “вперед”. Тоді змінна набуде значення \$where = “вперед”. Імена та значення змінних визначаються Таблицею 3. Змінні, які передаються діалоговій системі, можуть мати визначення через інші змінні. У цьому випадку змінна \$where визначається через змінні \$position, \$direction, \$range.

ID	MID	Q:ua
\$where	*	куди?
	*	в якому напрямку?
...	*	
\$obj	\$which_obj	об'єкт ще не знайдено, уточніть
	*	який об'єкт?
	*	назвіть об'єкт
	*	що за об'єкт?
	*	де знаходиться об'єкт?
...		

Таблиця 2. Таблиця питань до метаслів для технічної системи [10]

1.3.2. Запис голосів дикторів та опрацювання звукових файлів

Для створення акустичного корпусу для моделювання усного діалогу було використано записи голосу таких мовців: трьох чоловіків віком 20-50 років та 13 жінок віком 13-21 років.

Читане мовлення у дикторському виконанні записане для трьох мов – української, російської, англійської. Для деяких мовців російська мова є рідною. Для решти рідною мовою є українська. Усі мовці, які читали тексти англійською мовою, володіють нею на рівні B2 (Upper-Intermediate).

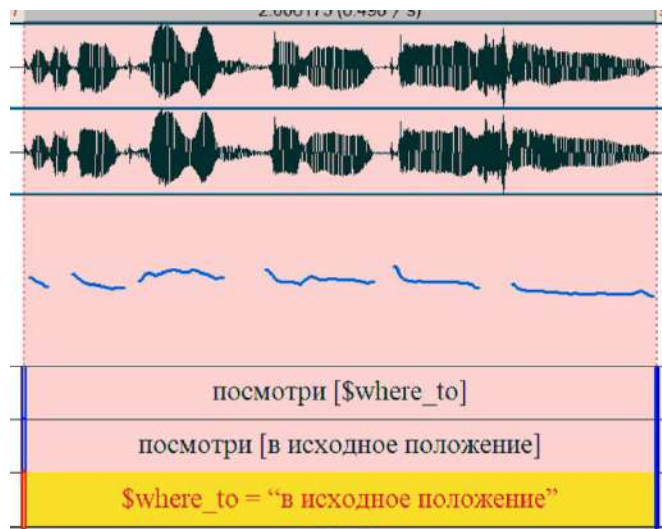
Записи читаного мовлення відбувалися не в студійних умовах: удома та на диктофон. Записи зроблені у форматі MP3. Відстань від мовця до мікрофону не була фіксованою, але знаходиться у межах 1-50 см.

Речення та метаслова з таблиць були поділені на уривки таким чином, щоб кожен мовець надиктував свій уривок таблиці трьома мовами. Також кожен тип речення був прочитаний трьома різними мовцями.

Корпус звукових записів був розмічений та проанотований у спеціалізованій комп'ютерній програмі Praat. Praat — це безкоштовна комп'ютерна програма для наукового фонетичного та акустичного аналізу усної мови. Він був розроблений П. Бурсмою та Д. Венінком з Амстердамського університету. Автори продовжують активно розвивати цей інструментарій. Він

може працювати на широкому діапазоні операційних систем, включаючи різні версії Unix, Linux, Mac і Microsoft Windows (2000, XP, Vista, 7, 8, 10). Програма підтримує синтез мови на основі формантного та артикуляційного підходів. Praat дозволяє аналізувати мовлення і керувати ним, обробляє аудіофайли у форматі WAV, AIFF, FLAC та іншими [13].

Звукові записи посегментовано та проанотовано в такий спосіб (Ілюстрація 5):



Ілюстрація 5. Зразок розмітки та анотації акустичного корпусу в Praat

Як бачимо, анотація є трирівневою, у ній:

- Перший рівень містить тип речення з незаповненим параметром-метасловом;
- Другий рівень містить тип речення із заповненим параметром;
- Третій рівень містить назву метаслова та слот, що відповідає метаслову.

ВИСНОВКИ ДО РОЗДІЛУ 1

У ході роботи було описано функціонал та алгоритм роботи мобільного лабораторного робота ERIC'а, створеного в Міжнародному науково-навчальному центрі інформаційних технологій та систем Національної академії наук України та Міністерства освіти та науки України.

Для моделювання діалогу з роботом ERIC'ом створено акустичний корпус, що містить:

- 30 інтентів;
- 96 LISP-структур російською мовою;
- 108 LISP-структур англійською мовою;
- 250 LISP-структур українською мовою.

Корпус складається із записів голосів 16 мовців.

Розмітка та анотація корпусу виконана на трьох рівнях:

- Перший рівень містить тип речення з незаповненим параметром-метасловом;
- Другий рівень містить тип речення із заповненим параметром;
- Третій рівень містить назву метаслова та слот, що відповідає метаслову.

РОЗДІЛ 2. СТВОРЕННЯ ТА ТРЕНУВАННЯ КЛАСИФІКАТОРА НАМІРІВ КОРИСТУВАЧА

2.1. Постановка задачі для класифікатора та опис інструментарію

На даному етапі задача ERIC’а – навчитися розуміти команди у вільній формі, а наша задача – створити модель-класифікатор, що буде розпізнавати так звані лейбли (мітки), тобто наміри користувача. Метою даного підходу є полегшення взаємодії людини та робота: користувачу не доведеться оголошувати команди для робота зі списку скінченних команд – тепер користувач зможе довільно висловлювати свої наміри та не хвилюватися, що робот їх не зрозуміє. Оскільки ERIC володіє скінченною кількістю «вмінь», список команд обмежений. Наразі арсенал робота складає 30 «вмінь», а отже користувач може оголосити не більше 30 команд.

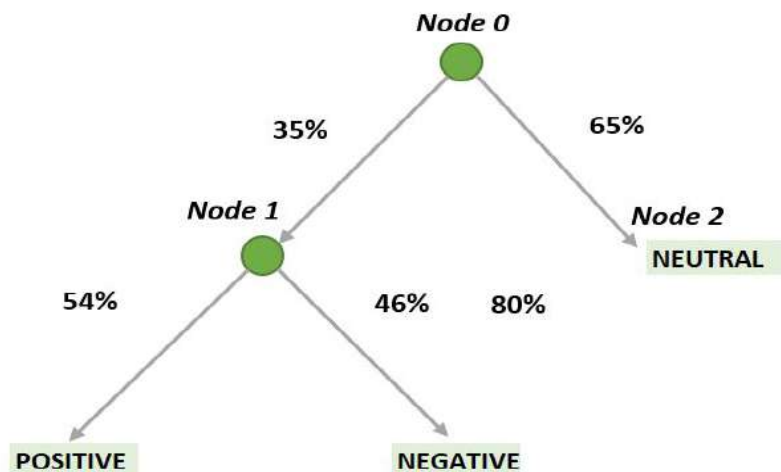
Задача класифікації належить до категорії машинного навчання з учителем, адже так званий учитель проставляє завчасно відомі мітки у даних. Ці мітки також називають лейблами (*label*), цільовими мітками, або ж значеннями, що очікуються [14]. Оскільки в команді користувача має міститися не більше одного наміру, одна фраза користувача може мати лише одну мітку.

Наша задача є небінарною в цілому, адже міток всього 30, а не дві, отже вона відноситься до категорії мультикласової класифікації (*multiclass classification*). Не слід плутати з мультилейбловою класифікацією (*multilabel classification*), адже у такому випадку кожна фраза користувача може мати більше однієї мітки [14]. Якщо розглянути кожну окрему фразу користувача, то можна застосувати бінарну логіку: наприклад, для фрази «*ходи до мене, Еріку*» модель буде вирішувати, чи підходить сюди мітка «*come2obj*». За високої точності моделі, рішенням буде «*так*».

У такому разі для створення і тренування моделі-класифікатора перед розробником постає величезний вибір інструментарію для реалізації даної задачі, а саме: *FastText*, *Scikit-learn*, *Natural Language Toolkit (NLTK)*, *SpaCy*,

TensorFlow, PyTorch, Keras. Ми надали перевагу інструменту *FastText*. ***FastText*** – бібліотека з відкритим кодом, яка дає змогу вивчати текстові представлення та створювати/використовувати текстові класифікатори. *FastText* підтримує 157 мов, містить натреновані моделі та ембединги. Ця бібліотека є продуктом Facebook AI і продовженням роботи творців *word2vec* та похідних технологій [15].

CBOW (Continuous Bag of Words) – це неглибока нейронна мережа, яка навчена передбачати слово, зважаючи на його контекст. *FastText*, по суті, міняє цільову мітку слова на цільові категорії. Ці одношарові моделі тренуються неймовірно швидко і можуть дуже добре масштабуватися. Ядро *FastText* спирається на модель *CBOW* для подання слів та на ієрархічний класифікатор для прискорення навчання. Крім того *FastText* замінює *softmax* над мітками на ієрархічний *softmax* (Ілюстрація 6). Кожен вузол представляє мітку. Обмежена кількість параметрів скорочує час тренування моделі [16].



Ілюстрація 6. Ієрархічна архітектура класифікатора *FastText* для сентимент-аналізу [17]

2.2. Підготовка даних для подальшого тренування

Початковим варіантом є таблиця з назвами намірів та власне LISP-структурами українською мовою (Ілюстрація 7).

#!TH	name(намір)	ukr
	take_obj	візьми об'єкт подарунок [найближчий] [ліворуч] підними kota Леопольда [найближчого] [ліворуч] підними об'єкт kota Леопольда [найближчого] [ліворуч] вхопися за об'єкт kota Леопольда [найближчого] [ліворуч] вхопися за kota Леопольда [найближчого] [ліворуч] візьми kota Леопольда [найближчого] [ліворуч]
	come2obj	підійти до об'єкта кіт Леопольд [найближчий] [ліворуч] підійти до Леопольда [найближчого] [ліворуч] іди до об'єкта Леопольда [найближчого] [ліворуч] іди до Леопольда [найближчого] [ліворуч] греби до об'єкта Леопольда [найближчого] [ліворуч] греби до Леопольда [найближчого] [ліворуч] сунь до об'єкта Леопольда [найближчого] [ліворуч] сунь до Леопольда [найближчого] [ліворуч] рухайся до об'єкта Леопольда [найближчого] [ліворуч]

Ілюстрація 7. Назви намірів та власне LISP-структури

FastText вимагає вихідний текстовий файл, де [16]:

- кожне речення починається з нового рядка;
- на початку рядка вказані:

__label__<мітка>, якщо міток декілька, вказати їх за такою ж схемою через пробіл, наприклад, __label__<мітка> __label__<мітка> (ця послідовність символів називається префіксом, за замовчуванням вона має значення «*label*», тож при бажанні її можна змінити на щось інше, тож матимемо, наприклад, __mitka__<мітка>);
- усі слова написані в нижньому регістрі.

Отже, проведено таку підготовку файлу:

- усі слова було зведено до нижнього регістру;
- було видалено такі допоміжні для LISP-структури символи: [/], \$, ;
- видалено зайві пробіли;
- на початку кожного рядка приписано одну мітку.

У результаті отримуємо готовий вихідний файл (Ілюстрація 8):

```

_label_come2obj котись до об'єкта леопольда найближчого ліворуч
_label_come2obj котись до леопольда найближчого ліворуч
_label_come2obj котися до об'єкта леопольда найближчого ліворуч
_label_come2obj котися до леопольда найближчого ліворуч
_label_come2obj під'їдь до об'єкта леопольда найближчого ліворуч
_label_come2obj під'їдь до леопольда найближчого ліворуч
_label_come2obj під'їжджай до об'єкта леопольда найближчого ліворуч
_label_come2obj під'їжджай до леопольда найближчого ліворуч
_label_analyse_obj проаналізуй об'єкти кіт леопольд найближчий ліворуч
_label_analyse_obj аналізуй об'єкти кіт леопольд найближчий ліворуч
_label_analyse_obj надай інформацію про об'єкти кіт леопольд найближчий ліворуч
_label_analyse_obj здійсни аналіз об'єкти кіт леопольд найближчий ліворуч
_label_analyse_obj виконай аналіз об'єкти кіт леопольд найближчий ліворуч
_label_analyse_obj виконуй аналіз об'єкти кіт леопольд найближчий ліворуч
_label_analyse_obj здійсни аналіз об'єкти кіт леопольд найближчий ліворуч
_label_analyse_obj отримай інформацію з об'єкти кіт леопольд найближчий ліворуч
_label_exam_obj дослідь об'єкт кіт леопольд найближчий ліворуч
_label_exam_obj вивчи об'єкт кіт леопольд найближчий ліворуч
_label_exam_obj повивчай об'єкт кіт леопольд найближчий ліворуч
_label_exam_obj досліджуй об'єкт кіт леопольд найближчий ліворуч
_label_exam_obj що цікавого розкажеш про об'єкт кіт леопольд найближчий ліворуч
_label_exam_obj що цікавого скажеш про об'єкт кіт леопольд найближчий ліворуч
_label_find_obj знайди об'єкт кіт леопольд ліворуч найближчий
_label_find_obj спробуй знайти об'єкт леопольд
_label_find_obj знайди об'єкт леопольд ліворуч найближчий
_label_find_obj здійсни пошук об'єкт леопольд ліворуч найближчий
_label_find_obj пошук об'єкт леопольд ліворуч найближчий

```

Ілюстрація 8. Готовий вихідний файл із намірами та мітками

Постає завдання розбити датасет, тобто вирази для кожного наміру, на тренувальну та тестову вибірки. Зазвичай це роблять, дотримуючись співвідношення 20-30% від усього датасету для тестового сету та відповідно 80-70% для тренувального. Оскільки наші датасети можна вважати мікродатасетами, на тренувальну частину доведеться віддати більше ніж 80%, а саме 85-95%. Оскільки деякі наміри мали подекуди дві LISP-структури, було вирішено в таких випадках ділити навпіл, тобто 50/50%.

У ході експериментальної роботи вирішено створити три датасети на основі одного вихідного файлу.

Датасет №1 генеруємо у такий спосіб: для кожної з 286 команд перемішуємо слова у власних межах, кількість ітерацій циклу – 50. Дублікати видаляються. Генерується 6 011 команд (див. Додаток 1). Початковий порядок згенерованих речень у файлі порушуємо – перемішуємо рядки за допомогою бібліотеки *random* (метод *shuffle*). Розділяємо датасет з 6 011 команд на 5% та 95% – відповідно перші 300 рядків для тестового сету та решта 5 711 рядків для тренувального сету. При такому підході, вірогідно, показники моделі

будуть найкращими, адже обидва сети будуть перетинатися однаковими наборами слів для рядка.

Спосіб генерації **датасету №2**. Кожен намір містить 2-24 варіанти команд. Для кожного наміру вручну відібрано більше ніж 90% (50% у випадку малої кількості варіантів команд) команд у тренувальний сет, решта – у тестовий сет. Таким чином команди не перетинаються, але й не мають великої кількості даних. Датасет містить 210 речень, із яких 178 – тренувальні, що складає 85%, а 32 – тестові (15%).

Датасет №3 схожий на датасет №2, але з тренувального сету згенеровано 3 028 речень за 50 ітерацій циклу. Дублікати видалено. Тестовий сет залишився таким же – 49 команд. Отже сети не перетинаються, але тренувальний сет, що спершу містив 178 речень, збільшився у 16 разів.

Для кращої візуалізації результати були вміщені в Таблицю 3 (детальніше див. Додаток 2).

	Д а т а с е т №1		Д а т а с е т №2		Д а т а с е т №3	
	т р е н у в а л ь н и й с е т - 95%	т е с т о в и й с е т - 5%	т р е н у в а л ь н и й с е т - 85%	т е с т о в и й с е т - 15%	т р е н у в а л ь н и й с е т - 98%	т е с т о в и й с е т - 2%
к і л ь к і с т ь р е ч е н ь (к о м а н д)	5711	300	178	32	2996	32
к і л ь к і с т ь л е м (х/289)	288/289	179/289	253/289	72/289	253/289	72/289

Таблиця 3. Інформація про датасети

2.3. Тренування моделей

Щоб уникнути можливих проблем із програмним забезпеченням та пришвидшити процес тренування, вся робота виконувалася в середовищі *Google Colab*. **Google Colab** – безкоштовний хмарний сервіс на основі *Jupyter Notebook*, створений для наукових експериментів [19]. Безсумнівна перевага

сервісу – це те, що він надає змогу користуватися дуже швидкими GPU и TPU, що є хорошим рішенням для машинного навчання. Можна використовувати вбудований термінал та редактор коду.

Тренування виконується за викликом `train_supervised` бібліотеки *FastText* (Додаток 3).

Для кожного датасету натреновано дві моделі. Перша модель (далі – «слабка модель») вчиться на параметрах за замовчуванням, а друга модель (далі – «сильна модель») має параметри, які розраховані на підвищення ефективності моделі. Ці параметрами є гіперпараметрами відповідно до термінології галузі машинного навчання, адже вони задаються вчителем до початку тренування [14].

Параметрами за замовчуванням є (числа у квадратних дужках є значеннями параметрів [20]):

- -lr: learning rate [0.1]
- -dim: size of word vectors [100]
- -ws: size of the context window [5]
- -epoch: number of epochs [5]
- -neg: number of negatives sampled [5]
- -loss: loss function {ns, hs, softmax} [softmax]
- -thread: number of threads [12]
- -pretrainedVectors pretrained word vectors for supervised learning []
- -saveOutput: whether output params should be saved [0]

Рекомендації щодо зміни значень кожного з гіперпараметрів задля налаштування тренування (Ілюстрація 9):

TABLE 1 | Hyperparameters chosen for tuning FastText model.

Parameters	Range	Stepsize	Optimal
Lr	0.05–0.25	0.05	0.1
Dim	50–500	25	100
Ws	1–10	1	5
Epoch	25–500	25	100
Loss	[ns, hs, softmax]	-	softmax

Lr, learning rate; dim, dimension; ws, size of context window; epoch, number of iterations; loss, loss function.

Ілюстрація 9. Рекомендації для налаштування значень параметрів [20]

Фрагмент-приклад програмного коду для «слабкої моделі», яка натренувалась за параметрами за замовчуванням:

```
weak_model_1 = fasttext.train_supervised(input="train_1.txt")
```

Фрагмент-приклад програмного коду для «сильної моделі»:

```
model_1 = fasttext.train_supervised(input="train_1.txt", lr=0.5, epoch=10, wordNgrams=4, bucket=200000, dim=100, loss='ova')
```

2.4. Метрики, що використовуються для оцінки моделей

Для оцінки якості моделей використаний метод `test` та `predict` бібліотеки *FastText*.

Метриками, обраними для оцінки якості моделі є точність/надійність (ассигасу) та якість класифікації на «живих прикладах», які не входять до датасетів. «Живі» приклади можуть бути в межах предметної області «Керування мобільним роботом», а можуть бути й сторонніми для неї.

Формула точності/надійності:

$$Accuracy = \frac{TP+TN}{TN+TP+FN+FP}$$

Фрагмент-приклад програмного коду, який викликає метод тестування:

```
weak_model_3.test("test_3.txt").
```

Фрагмент-приклад програмного коду, який викликає метод передбачення мітки:

```
model_3.predict("не бачу як ти поспішаєш із завданням", k=3.
```

«Живими» фразами обрано такі речення:

- «не бачу як ти поспішаєш із завданням»;
- «еріку гарна робота можеш взяти відпочинок»;
- «впорядкуй будь ласка об'єкти»;
- «за замовчуванням вона має значення»;
- «так званий учитель проставляє завчасно відомі мітки у даних»;
- «вбудований термінал та редактор коду».

2.5. Результати: оцінка моделей та висновки

У ході експериментів створено три моделі, кожна з яких тренувалась на власному тренувальному сеті і тестувалась на власному тестувальному сеті. Результати дослідження вміщено в таблицю (див. Таблиці 4-6).

		Слабка модель 1	Сильна модель 1
	Надійність АСС	0.96	0.9933333333333333
"Живі" приклади, що входять у предметну область "Керування мобільним роботом"	predict("не бачу як ти поспішаєш із завданням", k=3) - правильна мітка - '__label__run'	((('__label__run', '__label__relax', '__label__start'), array([0.50213528, 0.07249071, 0.05108482])))	((('__label__run', '__label__cancel_mis', '__label__check_sim'), array([0.86704576, 0.03623005, 0.01134175])))
	predict("еріку гарна робота можеш взяти відпочинок", k=3) - правильна мітка - '__label__relax'	((('__label__spk_uk', '__label__run', '__label__relax'), array([0.14345156, 0.14056815, 0.12063737])))	((('__label__relax', '__label__clarify', '__label__check_sim'), array([0.89331943, 0.01543456, 0.01407363])))
	predict("впорядкуй будь ласка об'єкти", k=3) - правильна мітка - '__label__sort_obj'	((('__label__sort_obj', '__label__analyse_obj', '__label__run'), array([0.83789629, 0.07419781, 0.05553239])))	((('__label__sort_obj', '__label__analyse_obj', '__label__relax'), array([0.9615438 , 0.12593275, 0.00942259])))
	за замовчуванням вона має значення	((('__label__monitor', '__label__foll_obj',	((('__label__foll_obj', '__label__stop',

предметну область "Керування мобільним роботом"		'__label__spk_uk'), array([0.49013999, 0.22470842, 0.12662126]))	'__label__start'), array([0.10375863, 0.05034063, 0.04469086]))
	так званий учитель проставляє завчасно відомі мітки у даних	((('__label__navigate', '__label__run', '__label__spk_uk'), array([0.25553706, 0.10236386, 0.09080145]))	((('__label__navigate', '__label__start', '__label__stop'), array([0.11280541, 0.10971579, 0.09269778]))
	вбудований термінал та редактор коду	((('__label__spk_uk', '__label__run', '__label__relax'), array([0.47451043, 0.17418723, 0.15648341]))	((('__label__start', '__label__relax', '__label__stop'), array([0.12941273, 0.10375863, 0.09010299]))

Таблиця 4. Результати для моделей «Слабка модель 1» та «Сильна модель 1» на датасеті №1

Як бачимо, «Сильна модель 1» показала себе краще ніж «Слабка модель 1» як на позитивних, так і на негативних реченнях. Зеленим кольором виділені позитивні результати. «Сильна модель 1» переважає в шести категоріях, а «Слабка модель 1» — лише у двох. «Слабка модель» дає занадто “позитивну” вірогідність для фраз, які не входять у предметну область. «Сильна модель» має дуже високу ефективність.

		Слабка модель 2	Сильна модель 2
	Надійність ACC	0.09375	0.03125
"Живі" приклади, що входять у предметну область "Керування мобільним роботом"	predict("не бачу як ти поспішаєш із завданням", k=3) - правильна мітка - '__label__run'	((('__label__run', '__label__relax', '__label__start'), array([0.03338204, 0.03338201, 0.03338123]))	((('__label__spk_en', '__label__open_grip', '__label__run'), array([0.14415886, 0.14034626, 0.13661839]))
	predict("епіку гарна робота можеш взяти відпочинок", k=3) - правильна мітка - '__label__relax'	((('__label__relax', '__label__run', '__label__start'), array([0.03341675, 0.0334158 , 0.03341533]))	((('__label__spk_en', '__label__start', '__label__relax'), array([0.37755069, 0.37023538, 0.37023538]))
	predict("впорядкуй будь ласка об'єкти", k=3) - правильна мітка - '__label__sort_obj'	((('__label__relax', '__label__run', '__label__start'), array([0.03339678, 0.03339662, 0.0333957]))	((('__label__spk_en', '__label__run', '__label__start'), array([0.22816648, 0.21207881, 0.20182322]))

"Живі" приклади, що НЕ входять у предметну область "Керування мобільним роботом"	за замовчуванням вона має значення	((('__label__relax', '__label__start', '__label__run'), array([0.03345264, 0.03345032, 0.03344947])))	((('__label__spk_en', '__label__run', '__label__relax'), array([0.23935935, 0.2281664, 0.21734752])))
	так званий учитель проставляє завчасно відомі мітки у даних	((('__label__relax', '__label__run', '__label__start'), array([0.0334501 , 0.03344881, 0.03344806])))	((('__label__spk_en', '__label__run', '__label__relax'), array([0.3629792 , 0.35578486, 0.3486551])))
	вбудований термінал та редактор коду	((('__label__relax', '__label__run', '__label__start'), array([0.03355746, 0.03355485, 0.03355334])))	((('__label__spk_en', '__label__run', '__label__relax'), array([0.27513972, 0.26285186, 0.25684199])))

Таблиця 5. Результати для моделей «Слабка модель 2» та «Сильна модель 2» на датасеті №2

Моделі «Слабка модель 2» та «Сильна модель 2» однаково спрацювали на фразах у межах та поза межами предметної області. У деяких місцях «Слабка модель» спрацювала краще ніж «Сильна модель». Жодна з моделей не має переваг.

		Слабка модель 3	Сильна модель 3
	Надійність ACC	0.3125	0.4375
"Живі" приклади, що входять у предметну область "Керування мобільним роботом"	predict("не бачу як ти поспішаєш із завданням", k=3) - правильна мітка - '__label__run'	((('__label__run', '__label__local_u', '__label__open_grip'), array([0.65874749, 0.17048055, 0.02762386])))	((('__label__run', '__label__local_u', '__label__find_obj'), array([0.8519628 , 0.01972913, 0.01451358])))
	predict("епіку гарна робота можеш взяти відпочинок", k=3) - правильна мітка - '__label__relax'	((('__label__relax', '__label__start', '__label__stop'), array([0.59267658, 0.56218654, 0.48439005])))	((('__label__relax', '__label__start', '__label__stop'), array([0.56218654, 0.51562995, 0.43783349])))
	predict("впорядкуй будь ласка об'єкти", k=3) - правильна мітка - '__label__sort_obj'	((('__label__sort_obj', '__label__analyse_obj', '__label__monitor'), array([0.67741007, 0.09141832, 0.04940153])))	((('__label__sort_obj', '__label__analyse_obj', '__label__cont_res'), array([0.66542059, 0.06372499, 0.01364684])))

"Живі" приклади, що НЕ входять у предметну область "Керування мобільним роботом"	за замовчуванням вона має значення	((('__label__monitor', '__label__foll_obj', '__label__navigate'), array([0.39795762, 0.23140442, 0.10027517])))	((('__label__foll_obj', '__label__monitor', '__label__relax'), array([0.12593275, 0.05835584, 0.03022459])))
	так званий учитель пропоставляє завчасно відомі мітки у даних	((('__label__navigate', '__label__spk_uk', '__label__monitor'), array([0.4696146, 0.21049747, 0.13195726])))	((('__label__navigate', '__label__relax', '__label__cont_res'), array([0.23935935, 0.15204224, 0.11280541])))
	вбудований термінал та редактор коду	((('__label__navigate', '__label__spk_uk', '__label__local_u'), array([0.38593933, 0.29627004, 0.21976794])))	((('__label__relax', '__label__stop', '__label__start'), array([0.12593275, 0.08757384, 0.08510906])))

Таблиця 6. Результати для моделей «Слабка модель 3» та «Сильна модель 3» на датасеті №3

«Сильна модель 3» працює краще за «Слабку модель 3», але має свої недоліки – вона лише розрізняє фрази потрібної предметної галузі. Надійність дуже низька, як і в попередньої пари моделей.

ВИСНОВКИ ДО РОЗДІЛУ 2

У ході роботи створено три датасети на основі одного вихідного файлу та натреновано шість моделей-класифікаторів намірів користувача щодо роботи ERIC’а.

Показовими метриками очевидно можна вважати надійність (ассурагу) та «живі приклади», які визначають три ($k=3$) найвірогідніші мітки. Ці значення відрізняють моделі одна від одної та відповідають дійсній ефективності моделей.

Що стосується гіперпараметрів за замовчуванням та адаптованих, адаптовані дають кращі результати в кожній із «сильних» моделей на противагу «слабким». Найгірші результати показала «Сильна модель 2», яка тренувалась

на датасеті без генерації. Цій моделі не допомогли навіть поліпшені гіперпараметри.

Дуже хороші результати показала генерація речень: моделі, треновані на більших датасетах, виявилися найефективнішими. Можна вважати, що «Сильна модель 1» є справді якісною моделлю на цьому етапі роботи. Зробимо висновок, що одношарова нейронна мережа не проводить настільки розумні операції, аби, у нашому випадку, «знати», що $[a, b, c] = [c, b, a]$.

ВИСНОВКИ

У роботі була зроблена спроба підготувати середовище для моделювання усного діалогу з лабораторним мобільним роботом ERIC'ом, створеним у Міжнародному науково-навчальному центрі інформаційних технологій та систем НАН України та МОН України.

На першому етапі підготовки було записано, посегментовано та проанотовано акустичний корпус на основі читаного мовлення 16 дикторів, що складається з:

- 30 інтенів (намірів);
- 250 LISP-структур (типів речень) українською мовою;
- 108 LISP-структур англійською мовою;
- 96 LISP-структур російською мовою.

Розмітка та анотація корпусу була виконана на трьох рівнях.

На другому етапі роботи нашою задачею було створення моделі всіх можливих речень, що входять у діалог та виражають один і той самий зміст, моделювання параметрів слів у типах речень, генерація та пошук найбільш правдоподібних еталонних сигналів та їх параметрів. У результаті було створено три датасети та шість моделей шляхом машинного навчання з учителем за допомогою бібліотеки *FastText* від *Facebook AI*. Найкраще показала себе “Сильна модель 1”, надійність якої складає 0.993. Ця модель найефективніше визначає мітки для наміру користувача та орієнтується у випадках, коли на вхід подається речення, що не належить до предметної галузі “Керування мобільним роботом”. Дана модель показала найкращі результати, оскільки при її тренуванні були задані адаптовані гіперпараметри, а її тренувальна вибірка містить 6 011 речень (що більше ніж в інших моделях) завдяки запропонованому та реалізованому генеруванню еталонних текстів.

В майбутніх дослідженнях, доцільно провести експеримент зі зміною такого гіперпараметра, як кількість *n-gram*. Оскільки *FastText* токенизує речення

самостійно, результат може змінитися. Варто також використати технологію векторного представлення речень, *Sentence2Vec*, щоб класифікувати наміри точніше та диференціювати предметні області якісніше.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. International Federation of Robotics: World Robotics Report, 2016 [Електронний ресурс]. Режим доступу:
https://ifr.org/img/office/Industrial_Robots_2016_Chapter_1_2.pdf
2. DALE R. The return of the chatbots // Natural Language Engineering. — 2016. — Vol. 22, no. 5. — P. 811–817.
3. Сухоручкина О.Н. Структуры функциональной организации интеллектуализированного управления мобильной системой // УСиМ. — 2007. — № 3. — С. 26–33, 63
4. О.Н. Сухоручкина, Н.В. Прогонный. Интеллектуальное управление мобильным роботом при слежении за подвижным объектом. // Международный научно-технический журнал «Проблемы управления и информатики» (6'2019), стор. 112-124
5. Рапопорт Г.Н., Герц А.Г. Искусственный и биологический интеллекты. Общность структуры, эволюция и процессы познания. — М.: КомКнига, 2005. — 312 с
6. Сухоручкина О.Н. Структуры функциональной организации интеллектуализированного управления мобильной системой // УСиМ. — 2007. — № 3. — С. 26–33, 63
7. В.В. Яценко. Параметризація типів речень предметної області для системи усного фразника-перекладача. // Науково-теоретичний журнал Штучний Інтелект (ШІ'2011), №4, стор. 134-142.
8. Валентина Яценко. Автоматизовані засоби формування лінгвістичної бази даних і знань для системи усного перекладу. // Науково-теоретичний журнал Штучний Інтелект (ШІ'2012), №4, стор. 211-220.].
9. Vintsiuk T.K., Analysis, Recognition and Understanding of Speech Signals / T.K Vintsiuk. – Kyiv : Naukova Dumka, 1987. – 264 p

10. Sazhok M. Spoken translation system based on speech understanding in subject area / M. Sazhok, V. Yatsenko. // All-Ukrainian Int. Conference on Signal : Image Processing and Pattern Recognition UkrObraz'2010. – Kyiv, 2010. – P. 103-106
11. Сухоручкина О.Н. Структуры функциональной организации интеллектуализированного управления мобильной системой // УСиМ. — 2007. — № 3. — С. 26–33, 63
12. Ніна Васильєва, Микола Сажок, Ольга Сухоручкіна, Валентина Яценко. Моделювання усного діалогу між людиною і технічною системою // Всеукраїнська конференція “УкрОбраз 2014”, Київ, 2014 р., с 13-16
13. Praat. [Електронний ресурс].
Режим доступу: <https://www.fon.hum.uva.nl/praat/>
14. Дейтел Пол, Дейтел Харви. Python. Искусственный интеллект, большие данные и облачные вычисления – Питер, 2020. – 864с.
15. What is fastText? [Електронний ресурс]: [Веб-сайт] – Електронні дані. – Режим доступу: <https://fasttext.cc/>
16. Bag of Tricks for Efficient Text Classification, Armand Joulin, Edouard Grave, Piotr Bojanowski, Tomas Mikolov, 2016
17. FastText sentiment analysis for tweets: A straightforward guide. [Електронний ресурс]: [Веб-сайт] – Електронні дані. – Режим доступу: <https://towardsdatascience.com/fasttext-sentiment-analysis-for-tweets-a-straightforward-guide-9a8c070449a2>
18. Getting and preparing the data. [Електронний ресурс]: [Веб-сайт] – Електронні дані. – Режим доступу: <https://fasttext.cc/docs/en/supervised-tutorial.html>
19. Google Colab/ [Електронний ресурс]: [Веб-сайт] – Електронні дані. – Режим доступу: <https://colab.research.google.com/notebooks/intro.ipynb#>.
20. List of options. [Електронний ресурс]: [Веб-сайт] – Електронні дані. – Режим доступу: <https://fasttext.cc/docs/en/options.html>

ДОДАТКИ

Додаток 1

Програмний код для генерації

```
import random
file = open("plain_commands.txt")
contents=file.read()
list_of_commands = contents.split('\n')
print("Length of the input file (commands in file):", len(list_of_commands))
generated_list = []
for command in list_of_commands:
    for i in range(50):
        splitted_command = command.split()
        var_for_shuffling = splitted_command[1:]
        random.shuffle(var_for_shuffling)
        shuffled_command = '___' + str(splitted_command[0]) + " " +
str(var_for_shuffling) [2:]
        generated_list.append(shuffled_command)
no_duplicates_list = list(set(generated_list))
print("Generated strings (commands in file):", len(no_duplicates_list))
with open("generated_sents_50_iterations.txt", "w") as file:
    for i in no_duplicates_list:
        seminormal_i = i[2:-2]
        seminormal_i = seminormal_i.replace("'", '"', ' ')
        seminormal_i = seminormal_i.replace("'''", "''''", ' ')
        seminormal_i = seminormal_i.replace('"', "'", ',')
        seminormal_i = seminormal_i.replace('"', ' ')
        seminormal_i = seminormal_i.replace(" '", ' ')
        seminormal_i = seminormal_i.replace(", ", ' ')
        seminormal_i = seminormal_i.replace('"', ' ')
        file.write(seminormal_i + "\n")
```

Програмний код для створення частотного словника

```

file_name = str(input("Enter name of input file: "))
import pymorphy2
import sqlite3
import tokenize_uk
import re

#CREATE TABLE
import sqlite3
db = sqlite3.connect(file_name + '.db')
cursor = db.cursor()
cursor.execute("""CREATE TABLE sentences
                (id INTEGER PRIMARY KEY,
                 sentence TEXT NOT NULL);
                """)
cursor.execute("""CREATE TABLE words
                (id INTEGER PRIMARY KEY,
                 sent_id INTEGER NOT NULL,
                 morph VARCHAR(255) NOT NULL,
                 word_forms VARCHAR(255),
                 lemmas VARCHAR(255))
                ;""")
cursor.execute("""CREATE TABLE frequency_dict
                (lemmas VARCHAR(255),
                 frequency INTEGER);
                """)
morph_analyzer = pymorphy2.MorphAnalyzer(lang='uk')
with open(file_name, 'r', encoding='utf-8') as file:
    strings = file.read()
    new_string = re.sub(r"__label__.*?\b", "", strings)
    text_list = re.findall(r"""\b[\w']+\b""", new_string)
    print(type(text_list))
    text = ' '.join(text_list)
    print(type(text))
sentences = tokenize_uk.tokenize_sents(text)
db = sqlite3.connect(file_name + '.db')
cursor = db.cursor()
for sentence in sentences:
    cursor.execute("""INSERT INTO sentences

```

```

        (sentence) VALUES(?);
        """ , (sentence,))

db.commit()

for n in range(len(sentences)):
    sent = sentences[n].split()
    for word in sent:
        clean_word = re.sub(r'[0-9%:;,«»-.?!"] ({}\[ \]\n]', r'', word)
        cursor.execute("""INSERT INTO words (sent_id, morph, word_forms, lemmas)
        VALUES(?, ?, ?, ?);
        """, (n+1, word, clean_word, morph_analyzer.parse(clean_word)[0].normal_form))
db.commit()

#frequency dict for lemmas
rows = cursor.execute("""SELECT lemmas, COUNT(*)
        FROM words
        GROUP BY 1;""")

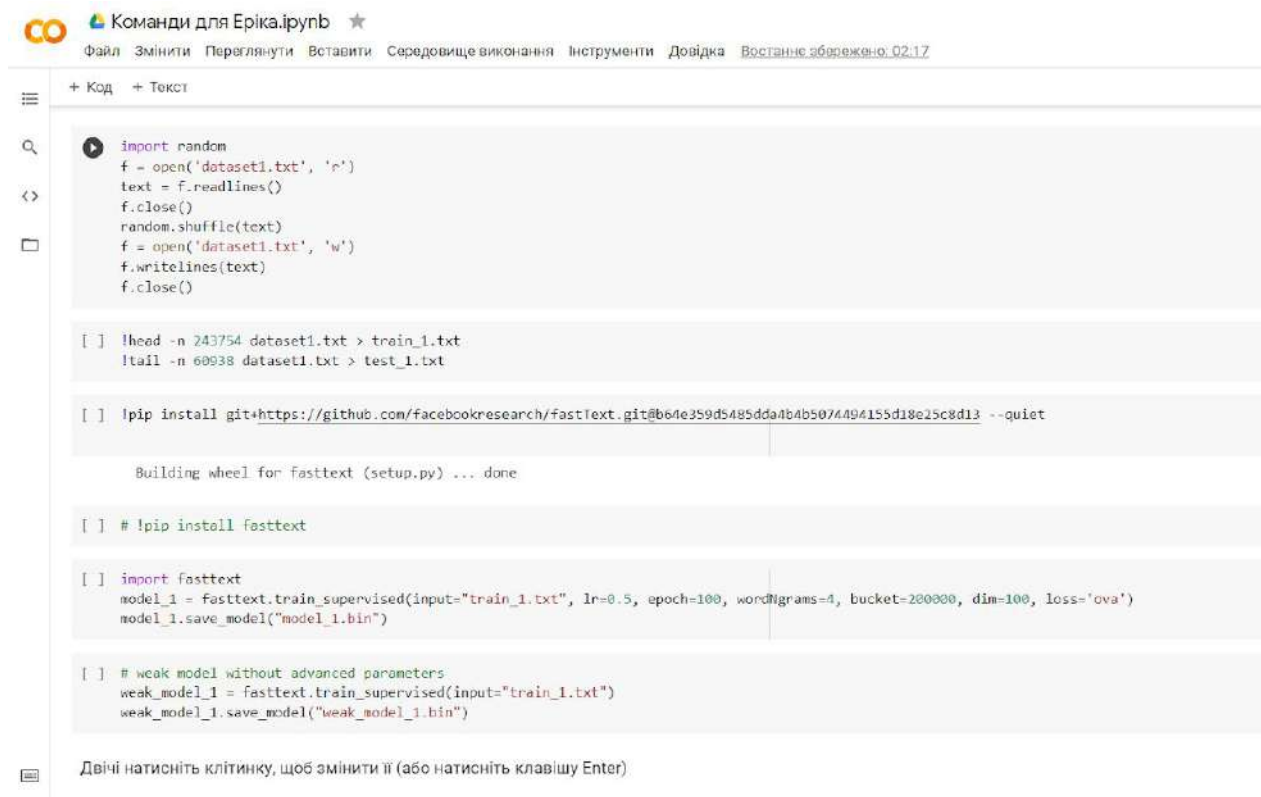
temp_cursor = db.cursor()
for row in rows:
    temp_cursor.execute("""INSERT INTO frequency_dict (lemmas, frequency) VALUES(?,
    ?) """,
        (row[0], row[1]))

db.commit()

```

Додаток 3

Фрагменти коду на Google Colab



Google Colab interface showing code execution. The top bar includes the Google Colab logo, the text "Команди для Еріка.ірунб", and a star icon. Below the bar are tabs: "Файл", "Змінити", "Переглянути", "Вставити", "Середовище виконання", "Інструменти", "Довідка", and "Востаннє збережено: 02:17". The main area has a tab for "+ Код" and "+ Текст". The code is executed in a Jupyter notebook style, with cells containing Python and shell commands. The output shows the installation of fastText and the training of a model.

```
import random
f = open('dataset1.txt', 'r')
text = f.readlines()
f.close()
random.shuffle(text)
f = open('dataset1.txt', 'w')
f.writelines(text)
f.close()

[ ] !head -n 243754 dataset1.txt > train_1.txt
    !tail -n 60938 dataset1.txt > test_1.txt

[ ] !pip install git+https://github.com/facebookresearch/fastText.git@b64e359d5485dda4b4b5074494155d18e25c8d13 --quiet

Building wheel for fasttext (setup.py) ... done

[ ] # !pip install fasttext

[ ] import fasttext
    model_1 = fasttext.train_supervised(input="train_1.txt", lr=0.5, epoch=100, wordNgrams=4, bucket=200000, dim=100, loss='ova')
    model_1.save_model("model_1.bin")

[ ] # weak model without advanced parameters
    weak_model_1 = fasttext.train_supervised(input="train_1.txt")
    weak_model_1.save_model("weak_model_1.bin")
```

Двічі натисніть клітинку, щоб змінити її (або натисніть клавішу Enter)



Google Colab interface showing code execution. The code is executed in a Jupyter notebook style, with cells containing Python and shell commands. The output shows the training of a model on a second dataset.

```
[ ] !wc -l train_2.txt
    !wc -l test_2.txt

wc: train_2.txt: No such file or directory
wc: test_2.txt: No such file or directory

[ ] import fasttext

    model_2 = fasttext.train_supervised(input="train_2.txt", lr=0.5, epoch=100, wordNgrams=4, bucket=200000, dim=100, loss='ova')
    model_2.save_model("model_2.bin")

[ ] # weak model without advanced parameters

    weak_model_2 = fasttext.train_supervised(input="train_2.txt")
    weak_model_2.save_model("weak_model_2.bin")
```

ТЕСТУВАННЯ

```

model_2.test("test_2.txt")
(49, 0.5714285714285714, 0.5714285714285714)

[ ] weak_model_3.test("test_3.txt")

[ ] model_2.test("test_1.txt", k=5)
(60938, 0.1932628040040697, 0.9663100200203485)

[ ] weak_model_3.test("test_3.txt", k=2)
(49, 0.29591836734693877, 0.5918367346938775)

[ ] model_3.predict("дивись на мовеск", k=3)
({'__label__cons_obj': '__label__walk_look', '__label__check_sim'},
 array([1.91340372e-02, 1.00000034e-05, 1.00000034e-05]))

[ ] weak_model_3.predict("дивись на мовеск", k=3)

[ ] model_3.test_label('test_3.txt')
{'__label__refresh': {'precision': nan, 'recall': 0.0, 'fiscore': 0.0}, '__label__clarify': {'precision': nan, 'recall': 0.0, 'fiscore': 0.0}, '__label__stop': {'precision': nan, 'recall': 0.0, 'fiscore': 0.0}}

```

АНОТАЦІЯ

Назва конкурсної роботи: “Моделювання намірів користувача в усному діалозі між людиною та мобільним роботом”

Актуальність дослідження: Сервісна робототехніка – це клас роботів із різноманітними конструктивними рішеннями та принципами реалізації системи управління, що динамічно розвивається. Надзвичайно популярними є роботи, які виконують свою роботу автономно — без втручання людини. Такі роботи можуть працювати навіть у невизначених динамічних обставинах — у цьому полягає їх “інтелектуальність”. З’являється проблема розвитку технологій усного діалогу між людиною та машиною. У світі вже представлена велика кількість систем, що демонструють інтелектуальне розуміння природної мови. Науково-фантастичні фільми прогнозують таке майбутнє, де системи усного діалогу будуть настільки інтелектуально довершеними, що замінять спілкування з людиною. Але реальний стан технологій усного діалогу говорить нам про те, що до такого майбутнього ще далеко, а отже, є над чим працювати. Саме активним розвитком галузі мовленнєвих інформаційних технологій, робототехніки та штучного інтелекту (як у світі, так і в Україні) зумовлена **актуальність** цієї роботи.

Мета дослідження — розробити мовленнєвий корпус та створити класифікатор намірів користувача для моделювання усного спілкування людини з роботом. Нашою задачею є створення моделі всіх можливих речень, що входять у діалог та виражають один і той самий зміст, моделювання параметрів слів у типах речень, генерація та пошук найбільш правдоподібних еталонних сигналів та їх параметрів. Досягнення мети передбачає виконання таких **завдань**:

1. Розглянути функціонал конкретного мобільного сервісного робота;
2. Розширити таблицю команд для мобільного сервісного робота, а саме збільшити варіативність типів речень;

3. Розробити акустичний корпус команд трьома мовами – українською, російською, англійською.

4. Створити та оцінити моделі-класифікатори намірів людини щодо робота.

Об’єктом дослідження є усний діалог людини з мобільним роботом.

Предметом дослідження є наміри користувача як сторони діалогу.

Практичне значення цієї роботи полягає в тому, що розроблений акустичний корпус та класифікатор будуть використовуватись для розширення мовного функціоналу робота та спрощення взаємодії між роботом і людиною.

Теоретичне значення роботи — це розвиток методів і алгоритмів моделювання інтелектуальної (свідомої та підсвідомої) діяльності людини.

Матеріалом дослідження є попередньо створена таблиця команд для сервісного робота та записи голосів дикторів.

Методи дослідження: експериментально-фонетичний метод (використовувався для створення, сегментації та анотації акустичного корпусу), метод машинного навчання, метод статистичного аналізу та метод аналізу через синтез (були використані для створення моделі з’ясування намірів користувача мобільного робота).

Інструменти дослідження: Анотація та сегментація звукових записів акустичного корпусу виконувалась за допомогою програми *Praat*. Задача мультикласової класифікації виконувалась за допомогою безкоштовної бібліотеки *FastText*, продукту Facebook AI. Програмний код написаний мовою *Python*. Для швидшої обробки файлів використано *Google Colab*.

Загальна характеристика роботи: Конкурсна робота складається зі вступу, 2 розділів, висновків, списку використаної літератури (20 позицій), додатків (3 фрагменти програмного коду), всього 35 сторінок (разом із титульною сторінкою та додатками).

У першому розділі "Функціонал робота ERIC’а та створення акустичного корпусу" проаналізовано особливості робота ERIC’а та частини його функціоналу (системи усного діалогу), описано суть таких понять: типи

смислів, типи речень, наміри користувача, висвітлено передумови, процес та результат створення, розмітки та анотації акустичного корпусу.

У другому розділі “Створення та тренування класифікатора намірів користувача” описана постановка задачі для класифікатора намірів, розглянуто підготовку даних, тренування моделей, визначено метрики для оцінки роботи моделей та описано результати експериментального дослідження.

У висновках проаналізовано результати створення акустичного корпусу та тренування моделей-класифікаторів, а також окреслено плани роботи на майбутнє, зазначено ті задачі, вирішення яких підвищить ефективність роботи класифікатора.