НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ імені ІГОРЯ СІКОРСЬКОГО»
ФАКУЛЬТЕТ ІНФОРМАТИКИ ТА ОБЧИСЛЮВАЛЬНОЇ ТЕХНІКИ
Кафедра автоматизованих систем обробки інформації і управління

До захисту допущено:
В.о. завідувача кафедри
_____ *Олександр ПАВЛОВ*
(підпис) (вл.ім'я, прізвище)
"___"_____2020 р.

Дипломний проєкт
на здобуття ступеня бакалавра

за освітньо-професійною програмою «Інформаційні управляючі системи та технології»
спеціальності 122 «Комп'ютерні науки та інформаційні технології»

на тему: «Інформаційно-аналітична система визначення лінгвістичних параметрів перекладача»

Виконав: студент *IV* курсу, групи *ІС-61*

Тодоров Дмитро Віталійович _____
(прізвище, ім'я, по батькові) (підпис)

Керівник *доц., к.т.н., Фіногенов Олексій Дмитрович* _____
(посада, науковий ступінь, вчене звання, прізвище, ім'я, по батькові) (підпис)

Консультант з графічної документації *доц., к.т.н., доц. Тєлишева Тамара Олексіївна* _____
(посада, науковий ступінь, вчене звання, прізвище, ім'я, по батькові) (підпис)

Рецензент *доц. каф. ПСОН, ПБФ, к.т.н. Сергій МУРАХОВСЬКИЙ* _____
(посада, науковий ступінь, вчене звання, прізвище, ім'я, по батькові) (підпис)

Засвідчую, що у цьому дипломному проєкті немає запозичень з праць інших авторів без відповідних посилань.
Студент _____
(підпис)

Київ – 2020 року

Національний технічний університет України
“Київський політехнічний інститут імені Ігоря Сікорського”

Факультет (інститут) *інформатики та обчислювальної техніки*
(повна назва)

Кафедра *автоматизованих систем обробки інформації і управління*
(повна назва)

Рівень вищої освіти – перший (бакалаврський)

Спеціальність – 122 «Комп'ютерні науки та інформаційні технології»

Освітньо-професійна програма «Інформаційні управляючі системи та технології»

ЗАТВЕРДЖУЮ
В.о. завідувача кафедри
_____ *Олександр ПАВЛОВ*
(підпис) (вл.ім’я, прізвище)
“___”_____2020 р.

З А В Д А Н Н Я
на дипломний проєкт студенту

Тодорову Дмитру Віталійовичу
(прізвище, ім’я, по батькові)

1. Тема проєкту «*Інформаційно-аналітична система визначення лінгвістичних параметрів перекладача* »

керівник проєкту *Фіногенов Олексій Дмитрович, к.т.н., доцент*
(прізвище, ім’я, по батькові, науковий ступінь, вчене звання)
затверджені наказом по університету від “*7*”*травня* 2020 р. №1081-с
2. Термін подання студентом проєкту “*01*”*червня* 2020 року
3. Вихідні дані до проєкту
Технічне завдання

4. Зміст пояснювальної записки
1. Загальні положення: основні визначення та терміни, опис предметного середовища, огляд ринку програмних продуктів, постановка задачі
2. Інформаційне забезпечення: вхідні дані, вихідні дані, опис структури бази даних
3. Математичне забезпечення: змістовна та математична постановки задачі, обґрунтування та опис методу розв’язання
4. Програмне та технічне забезпечення: засоби розробки, вимоги до

технічного забезпечення, архітектура програмного забезпечення, побудова звітів

5. Технологічний розділ: керівництво користувача, методика випробувань програмного продукту

5. Перелік графічного матеріалу

1. Схема структурна процесу діяльності

2. Схема структурна послідовності

3. Схема бази даних

4. Схема структурна класів програмного забезпечення

5. Схема структурна компонентів

6. Рішення математичного забезпечення

7. Креслення вигляду екранних форм

6. Консультанти розділів проєкту

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв

7. Дата видачі завдання *«13» квітня 2020 року*

Календарний план

№ з/п	Назва етапів виконання дипломного проєкту	Термін виконання етапів проєкту	Примітка
1.	*Вивчення рекомендованої літератури*	*16.04.2020*	
2.	*Аналіз існуючих методів розв'язання задачі*	*18.04.2020*	
3.	*Постановка та формалізація задачі*	*20.04.2020*	
4.	*Розробка інформаційного забезпечення*	*22.04.2020*	
5.	*Алгоритмізація задачі*	*24.04.2020*	
6.	*Обґрунтування використовуваних технічних засобів*	*25.04.2020*	
7.	*Розробка програмного забезпечення*	*30.04.2020*	
8.	*Налагодження програми*	*02.05.2020*	
9.	*Виконання графічних документів*	*07.05.2020*	
10.	*Оформлення пояснювальної записки*	*14.05.2020*	
11.	*Подання ДП на попередній захист*	*15.05.2020*	
12.	*Подання ДП на основний захист*	*01.06.2020*	
13.	*Подання ДП рецензенту*	*02.06.2020*	

Студент Дмитро Тодоров

Керівник Олексій Фіногенов

№ з/п	Формат	Позначення	Найменування	Кількість листів	Примітка
1	А4		Завдання на дипломний проєкт	2	
2	А4	ДП 6126.00.000 ПЗ	Пояснювальна записка	79	
3	А4	ДП 6126.01.000 ТЗ	Технічне завдання	10	
4	А3	ДП 6126.02.000 ССД	Схема структурна процесу діяльності	1	
5	А3	ДП 6126.03.000 ССП	Схема структурна послідовності	1	
6	А3	ДП 6126.04.000 СБД	Схема бази даних	1	
7	А3	ДП 6126.05.000 ССК	Схема структурна класів програмного	1	
			забезпечення		
8		ДП 6126.06.000 ССК	Схема структурна компонентів		
9	А3		Рішення з математичного забезпечення	1	
10	А3	ДП 6126.07.000 КЕ	Креслення вигляду екранних форм	2	

Зм.	Арк.	ПІБ	Підп.	Дата	ДП 6126 00.000.00		
Розробн.		Тодоров Д.В.			Відомість дипломного проєкту	Літ.	Лист 1 / Листів 1
Керівн.		Фіногенов О.Д.					
Консульт.						КПІ ім. Ігоря Сікорського Каф. АСОІУ Гр. ІС-61	
Н/контр.		Тєлишева Т.О.					
В.о.зав.каф.		Павлов О.А.					

АНОТАЦІЯ

Структура та обсяг роботи. Пояснювальна записка дипломного проекту складається з семи розділів, містить 18 рисунків, 26 таблиць, 1 додаток та 10 джерел.

Дипломний проект присвячений вирішенню задачі визначення лінгвістичних особливостей перекладача. Метою створення системи є визначення лінгвістичних особливостей перекладача, за допомогою яких можна було б ідентифікувати перекладача невідомого тексту.

У розділі загальних положень описано процес діяльності системи визначення лінгвістичних особливостей та ідентифікації перекладача, побудовано функціональну модель системи що проектується, наведено аналогічні системи ідентифікації автора. Визначено мету розробки та встановлено задачі, які необхідно виконати для досягнення цих цілей.

У розділі інформаційного забезпечення детально описано вхідні та вихідні дані, наведено схему та опис спроектованої бази даних.

В розділі математичного забезпечення обґрунтовано вибір методів розв’язання задачі та описано їх зміст.

Розділ програмного забезпечення описує засоби розробки програмного продукту описано його архітектуру, специфікацію функцій та звіти, які генеруються в ході виконання програми.

У технологічному розділі наведено інструкцію користувача та результати випробувань програмного продукту.

В розділі проведення досліджень описано експерименти та їх результати, зроблено висновки щодо актуальності методів розв’язання задачі.

ІДЕНТИФІКАЦІЯ ПЕРЕКЛАДАЧА, ЛІНГВІСТИЧНІ ОСОБЛИВОСТІ ПЕРЕКЛАДАЧА, ЧАСТОТНІСТЬ ГРАМІВ

ABSTRACT

Structure and scope of work. The explanatory note of the diploma project consists of seven sections, contains 18 figures, 26 tables, 1 application and 10 sources.

The diploma project is devoted to the decision of a problem of definition of linguistic features of the translator. The purpose of the system is to determine the linguistic features of the translator, which could be used to identify the translator of an unknown text.

In the section of general provisions the process of activity of the system of definition of linguistic features and identification of the translator is described, the functional model of the projected system is constructed, similar systems of identification of the author are given. There are defined the purposes of work and the tasks which need to be done for achievement of these purposes.

The section of information support describes in detail the input and output data, provides a diagram and description of the designed database.

The section of mathematical support substantiates the choice of methods for solving the problem and describes their sense.

The software section describes the software development tools, program architecture, feature specifications, and reports that are generated during program execution.

The technological section provides user instructions and test results of the software product.

The research section describes the experiments and their results, and draws conclusions about the use of methods for solving the problem.

TRANSLATOR IDENTIFICATION, LINGUISTIC FEATURES OF THE TRANSLATOR, GRAM FREQUENCY

ЗМІСТ

ВСТУП

Створення ЕОМ в середині 20-го століття і швидкий розвиток кібернетичної науки стимулювали появу нових наук, які раніше просто неможливо було уявити. Як правило, вони виникали на стику наук, часто не пов'язаних одна з одною. Так, на стику обчислювальної техніки та лінгвістики народилася наука, яка отримала назву «Комп'ютерна лінгвістика».

У появи комп'ютерної лінгвістики було дві причини. По-перше, кібернетична наука керується прагненням автоматизувати всі сфери людської діяльності. Однією з таких сфер є лінгвістика. Використання комп'ютерів дозволило лінгвістам автоматизувати трудомісткі процеси, наприклад, ведення лексичних та словникових картотек, статистичну обробку текстів, автоматичний переклад. По-друге, із введенням комп'ютерів у поширене користування з'явилася проблема організації зрозумілого інтерфейсу взаємодії користувача та програми. Звісно, найзрозумілішою мовою взаємодії для людини є природна мова. Але для того, щоб інтегрувати природну мову в інтерфейс програми, необхідно зрозуміти закони побудови та використання природної мови, такі як побудова речень, пов'язання слів, вибір закінчень і т.д.

Таким чином, комп'ютерна лінгвістика займається задачами лінгвістичного забезпечення процесів збору, накопичення, обробки та пошуку інформації. Найважливішими задачами є: автоматичний переклад тексту з однієї природної мови на іншу, задача пошуку нечітких дублікатів, задача класифікації текстів, задача пошуку інформації в базах даних, автоматичне індексування документів та інформаційних запитів.

Також в сьогоденній комп’ютерній лінгвістиці існують дослідження щодо задачі визначення авторства тексту. Вирішення даної задачі базується на виділенні особливостей (числових характеристик), притаманних конкретному автору. Така числова характеристика має зберігати «постійне значення» для усіх творів автора, тобто, мати невелике відхилення від середнього значення на протязі усіх його книг. Такими характеристиками можуть слугувати

довжина речень, довжина слів, частота використання іменників, прикметників, дієслів, розподіл частот буквосполучень (грамів).

Однак, при перекладі твору на іншу мову ми стикаємось з деякими проблемами як в задачі визначення автора, так і в інших задачах комп'ютерної лінгвістики (наприклад, визначення міжмовного плагіату). Ці проблеми виникають через те, що процес текстового перекладу з однієї мови на іншу містить не тільки трансформування слів від їх значення в оригіналі до їх значення в мові перекладу, але і трансформування структури побудови тексту, оскільки різні мови мають різні правила і «звички» побудови речень. Окрім того, тепер на текст, його форму та зміст має вплив не тільки особа автора, але і особа перекладача разом із його особистим сприйняттям мови, словниковим запасом та стилем мовлення. Те, що особа перекладача впливає на текст, можна помітити неозброєним оком, порівнявши різні переклади одного й того самого твору на одну й ту саму мову.

Отже, постає питання, чи можна методами комп'ютерної лінгвістики визначити та систематизувати, який вплив перекладач має на текст, за якими числовими характеристиками це можна визначити, та чи можна визначити перекладача за його перекладом.

Під час пошуку інформації не знайдено досліджень на предмет визначення лінгвістичних параметрів перекладача. Тому ця робота має за мету визначити, якими числовими характеристиками, за якими можна класифікувати перекладача, володіє перекладений текст.

Практичне значення одержаних результатів

Розроблено систему ідентифікації перекладача невідомих текстів методом аналізу частотності грамів.

Публікації

Наразі публікацій результатів роботи немає.

1 ЗАГАЛЬНІ ПОЛОЖЕННЯ

1.1 Опис предметного середовища

За історію свого розвитку комп'ютерна лінгвістика займалася дослідженням автоматизації багатьох прикладних лінгвістичних завдань, таких як класифікація і кластеризація тексту, машинний переклад, автоматичне розпізнавання тексту і т.д. У тому, числі, добре досліджувалася і завдання ідентифікації автора невідомого тексту. Вітчизняними та зарубіжними дослідниками було запропоновано безліч методів вирішення даного завдання [10], починаючи від простого підрахунку кількості певних слів в порівнюваних текстах і закінчуючи розробками в області штучного інтелекту

Однак, питання ідентифікації перекладача тексту на даний момент ніким не було зачеплено. Під час пошуку інформації не було знайдено публікацій, в яких були б наявні дослідження особливостей стилю перекладача і були б наведені програмні методи автоматичної ідентифікації перекладача тексту. Тому дослідження даного завдання є дуже актуальним для розвитку комп'ютерної лінгвістики і може знайти нові практичні застосування.

У сфері ідентифікації автора невідомого тексту популярністю користуються методи, засновані на припущенні про те, що кожен автор має набір специфічних стилістичних прийомів, характерних мовних особливостей (лексичних, граматичних, фразеологічних), який простежуються у всіх творах, завдяки яким його можна впізнати. Такими особливостями можуть бути середня довжина слів, частота використання різних частин мови, «улюблені слова» автора, частота використання певних буквосполучень і т.д.

Так як завдання ідентифікації перекладача ще не вивчалася, і, відповідно, ніякі відомі методи комп'ютерної лінгвістики, не пройшло перевірку в застосуванні до цього завдання, первинно необхідним та актуальним в цій області є дослідження того, чи володіє перекладач набором

стилістичних, мовних, морфологічних та інших особливостей, які будуть збереженість протягом усіх його перекладів різних текстів різних авторів. Якщо такі особливості існують, стоїть завдання визначити методи, за якими ці особливості можна інтерпретувати в числові характеристики.

1.1.1 Опис процесу діяльності

Розглянемо процес діяльності користувача в системі. Було вирішено розробити програмне забезпечення у вигляді веб-застосування.

Для початку роботи з програмою, продукт необхідно буде завантажити на сервер.

В даній системі буде присутній лише один актор – користувач. Також в системі може бути й актор Адміністратор, в якого буде доступ до адмін-панелі керування базою даних, проте цей актор нас не приймає важливу участь безпосередньо в процесі діяльності системи.

На веб-сторінці веб-застосування буде представлено три пункти меню: «Тренування», «Аналіз» і «Про програму»

В меню «Тренування» розташовано панель управління базою перекладачів. У цьому меню можна додавати та видаляти перекладачів. До кожного перекладача можна завантажити датасет з текстових файлів (за допомогою кнопки «Завантажити файли»), в яких містяться перекладені твори перекладача. Завантажені файли теж можна поодиноко видаляти та завантажувати нові. Це є датасет для тренування моделі. Після вибору датасету користувач може тренувати модель, натиснувши на кнопку «Тренувати».

В меню «Аналіз» реалізовано проведення аналізу сукупності текстів. Для цього користувач має завантажити тестові тексти за допомогою кнопки «Вибрати файли». Крім того, надається чек-ліст перекладачів, в якому можна визначити, порівняння з якими перекладачами необхідно здійснити. При натисненні кнопки «Провести аналіз» програма проводить аналіз текстів на

основі частотності грамів та частотності цілих слів. При великій кількості даних аналіз займе певний час. На вихід програма видає результати аналізу у вигляді файлу формату Excel.

1.1.2 Опис функціональної моделі

Всі дії або варіанти використання акторів, наведені в таблиці 1.1. В даній таблиці описаний актор, варіанти використання та опис його дій.

Таблиця 1.1– Типи залежностей між варіантами використання

Актор	*Варіант використа-ння*	*Опис дії варіанта використання*
Користувач	Додавання перекладача	Користувач вводить в текстове поле ім’я перекладача та додає його до моделі за допомогою кнопки «Додати перекладача».
	Додавання перекладача	Користувач може видалити перекладача з усіма його текстами і аналізами, натиснувши кнопку «Видалити» навпроти перекладача у меню «Тренування».
	Видалення текстів перекладача	Користувач видаляє файл, натиснувши на хрестик поруч із назвою файлу.
	Тренування моделі	Натисненням кнопки «Тренувати» здійснення тренування моделі та визначаються лінгвістичні параметри кожного перекладача.

Продовження таблиці 1.1

Актор	*Варіант використання*	*Опис дії варіанта використання*
Користувач	Завантаження файлів з лінгвістичними параметрами	При натисненні на посилання на файл з лінгвістичними параметрами перекладача, цей файл завантажується.
	Завантаження тестових файлів	При натисненні кнопки «Завантажити файли» користувач завантажує тестові файли.
	Проведення аналізу	Вибравши порівнюваних перекладачів та натиснувши кнопку «Провести аналіз» користувач проводить аналіз близькості текстів до обраних перекладачів. На вихід програма видає Excel-файл з результатами у вигляді коефіцієнтів близькості числових характеристик текстів до числових характеристик кожного перекладача.

1.2 Огляд наявних аналогів

Як вже зазначалося раніше, досліджень та програмних продуктів, що вирішують задачу визначення лінгвістичних параметрів саме перекладача, не знайдено. Додатковою перешкодою при дослідженні є відсутність якісних датасетів українською мовою та програмного забезпечення, що її підтримує. Найближчими аналогами можна вважати існуючи програми з визначення

авторства тексту російською мовою, характерними представниками яких з точки зору покладених у функціонування алгоритмів є наступні.

Система «Лінгвоаналізатор» [1,2]

Д.В. Хмельовим було проведено ряд дослідів, за результатами яких було зроблено висновки щодо ефективності використання алгоритмів, який використовує ланцюги Маркова першого порядку та методів стиснення[XX]. З'ясовано, що вони дають гарні результати на довгих текстах і погані у порівнянні з іншими алгоритмами на текстах розмірами у 2 – 5 тисяч символів. Такий метод було реалізовано у системі «Лінгвоаналізатор».

Метод, запропонований Д.В. Хмельовим заснований на застосуванні відносної ентропії. Існує декілька способів розрахунку цієї величини, зокрема, шельфовий алгоритм, метод передбачення за частковим співпадінням та застосування індексу повторюваності. Серед алгоритмів стиснення даних без втрат найбільш популярними є кодування Хафмана, арифметичне кодування, метод Бароуза-Вілера, та варіації методу Лемпеля-Зіва.

Необхідно зазначити, що серед алгоритмів, які спрямовані саме на стиснення текстів, найкраще себе показали метод передбачення за частковим співпадінням, який засновано на Марківському ланцюгу та метод динамічного Марківського стиснення, який також засновано на ланцюгу Маркова.

Загалом, для експерименту, проведеному на системі «Лінгвоаналізатор» було відібрано 285 текстів 82 письменників. Кожен текст був попередньо оброблений, викинуто роздільники, та слова, що починаються з великої літери. Результати показали, що методи, засновані на ланцюгах Маркова, визначають авторство тексту з ймовірністю 84% – 89%, що є прийнятним показником. Серед цікавих побічних результатів було те, що порівняння письменників різних історичних періодів давало погані результати, з чого можна зробити припущення, що історична епоха, коли було написано твір, суттєво впливає на точність результатів.

Система «Атрибутор» [3]

Як продовження підходу, який використовує у якості стилістичної характеристики бінарні буквосполучення, О.М. Тімашев запропонував використовувати тріади – трибуквенні буквосполучення [XX]. За такого підходу аналізу піддаються однобуквенні та двобуквенні службові слова, а це значна частина таких частин мови, як прийменник, частка, сполучник, які вважаються значущими стилеметричними характеристиками.

Саме через це двобуквенні сполучення, та сполучення з 4 та більше букв менш показові, що було виявлено в результаті досліду. На основі цього дослідження було побудовано програмний комплекс «Атрибутор», для автоматичного порівняння і класифікації текстів на основі триграм (трибуквенних сполучень).

База цієї системи містить твори 103 авторів та використовує експертну обробку текстів. У еталонну вибірку для системи було обрано романи, які отримані з електронних бібліотек. Вибірку було підібрано таким чином, щоб тексти різних письменників максимально відрізнялись одне від одного, а тексти одного письменника були максимально подібними.

Для обробки тексту цим програмним комплексом необхідно, щоб його об’єм складав не менше 6 сторінок. Це обмеження введене для того, щоб статистична вибірка була репрезентативною. В основі алгоритму аналізу системи «Атрибутор» також лежать Марківські ланцюги. Точність цього аналізатора оцінюється у 80 – 85%.

Система «Авторовед» [4,5]

Продовження досліджень щодо застосування нейронних мереж в поєднанні з методом опорних векторів при встановленні авторства текстів знайшло відображення в роботі «Авторовед». Якщо задачу визначення авторства сформулювати як задачу класифікації, то одним з широко розповсюджених рішень є побудова бінарного класифікатора. Всі тексти, включно з навчальною частиною вибірки, розгортаються в дуже великий вектор, що індексується словами. Після цього є дві множини точок з

навчальної вибірки в багатовимірному просторі: що належать даному автору і не належать автору. Для того щоб розділити ці множини, потрібно поділити простір на дві частини. Найпростіший спосіб зробити це – побудувати гіперплощину. Таку гіперплощину можна побудувати за допомогою методу опорних векторів (SVM – Support Vector Machines). Після цього для класифікації тексту з невідомим автором досить перевірити, в яку частину простору він потрапив. Прикладами застосування методу опорних векторів при встановленні авторства є роботи.

Методи класифікації за допомогою SVM значно перевершують численних конкурентів: кластерну класифікацію (коли документ співвідноситься до найближчої множини; в залежності від визначення відстані до множини – є багато варіантів реалізації цього методу – середньозважений, k найближчих сусідів і ін.), а також наївний Байєсовий класифікатор (який передбачає, що частоти слів у тексті є незалежними випадковими величинами).

В якості характерних ознак тексту для опису авторського стилю пропонується брати найбільш часті триграми символів і найбільш уживані(з найбільшою частотою) слова російської мови. Як інструменти для атрибуції текстів в даній роботі були обрані штучні нейронні мережі архітектури багатошаровий перцептрон (MLP), мережі каскадної кореляції (CCN) і апарат машини опорних векторів (SVM). CCN дозволяють знизити часові витрати на навчання в порівнянні з перцептроном за рахунок алгоритму автоматичної побудови топології мережі. SVM є найбільш точним з існуючих методів класифікації і одночасно найшвидшим. Підсумкове рішення про автора тексту приймається ансамблем класифікаторів за принципом мажоритарного голосування. Основні результати проведених досліджень були отримані на корпусі, що складається з 215 прозових текстів 50 російських письменників. Тексти взяті з електронної бібліотеки М. Мошкова [XX]. Розмір кожного тексту становив понад 100 000 символів. Використовувалися вибірки об'ємом

10 00 – 100 000 символів (200 – 20 000 слів). Кількість навчальних прикладів кожного учасника бралося рівним 3, для тестування використовувалося по 1 вибірці автора. Експерименти для випадку 2, 5 і 10 авторів показали, що найбільш інформативними авторським ознаками є обмеження в 300 – 700 найбільш частотних триграм і 500 найбільш частих слів. Автора можна визначити з точністю в середньому 95–98% при обсязі текстової вибірки 20 000 – 25 000 символів. При цьому починаючи з 10 000 символів машина опорних векторів показує кращі з трьох досліджуваних класифікаторів результати. Встановлено, що використання при ідентифікації автора комбінації частот букв російської мови, знаків пунктуації, найбільш частих триграм символів і найбільш частих слів збільшує точність ідентифікації в середньому на 6 –12% на обсягах тексту до 10 000 символів.

1.3 Постановка задачі

1.3.1 Призначення розробки

Метою створення системи є визначення лінгвістичних особливостей перекладача, за допомогою яких можна було б ідентифікувати перекладача невідомого тексту.

Так як на даний момент не знайдено досліджень щодо визначення лінгвістичних параметрів перекладача, даний дипломний проект може стати початком руху у цьому напрямку.

Цей проект може бути використаний науковцями та спеціалістами комп’ютерної лінгвістики у проведенні подальших досліджень у сфері визначення лінгвістичних параметрів перекладача.

Крім того, в разі успішного виявлення лінгвістичних особливостей перекладача, що спостерігаються впродовж усіх його робіт, дану розробку можна буде використати для залучення перших спроб в ідентифікації українських перекладачів для творів, особи перекладачів яких достовірно невідомі.

1.3.2 Цілі та задачі розробки

Цілями розробки системи є:

- визначення лінгвістичних особливостей, якими володіє перекладач;
- визначення можливості ідентифікації перекладача невідомого тексту.

Для досягнення поставлених цілей необхідно реалізувати наступні задачі:

- знайти перекладачів, якими було перекладано велику кількість творів, написаних різними авторами, при чому на кожний твір є переклади усіх перекладачів;
- створити датасет перекладів цих творів;
- реалізувати алгоритм визначення частотності грамів;
- реалізувати алгоритм визначення «мішку слів»;
- проаналізувати створений датасет за розробленими алгоритмами, за проведеним аналізом визначити лінгвістичні параметри досліджуваних перекладачів;
- провести множину експериментів за таким принципом: взяти текст одного з перекладачів, що не увійшов у датасет та провести ідентифікацію перекладача цього тексту;
- за результатами ідентифікації зробити висновки щодо доцільності використання даних методів для досліджень у визначенні лінгвістичних особливостей перекладача.

Висновок до розділу

В даному розділі наведено опис предметного середовища дослідження лінгвістичних параметрів перекладача, розглянуто існуючі програмні продукти з ідентифікації автора, так як програм для дослідження перекладачів немає, наведено опис функціональної моделі майбутньої системи, визначено призначення, ціль та задачі розробки.

2 ІНФОРМАЦІЙНЕ ЗАБЕЗПЕЧЕННЯ

2.1 Вхідні дані

Під час проектування даного продукту, було зроблено висновки, що найкращим рішенням для користувача і розробника є подача результатів досліджень у *.xlsx форматі, так як це зручний метод для отримання звіту і подальшого аналізу результатів. На вхід програми подаються текстові файли, що зберігаються в фізичній пам’яті комп’ютера користувача.

Дані, що надходять від користувача:

- список перекладачів;
- тренувальна вибірка текстів для кожного перекладача;
- тестова вибірка текстів.

2.2 Вихідні дані

Після проведення аналізу в браузер завантажується файл «Analysis.xlsx» з результатами аналізу. Цей файл містить дві таблиці: «Близькості» та «Результати аналізу».

В листі «Близькості» наводяться коефіцієнти близькості між тестовими текстами та стилями перекладачів за лінгвістичними параметрами.

В листі «Результати аналізу» наводяться дві таблиці. В першій таблиці наводяться результати ідентифікації кожного перекладача за кожним критерієм. В другій таблиці листа «Результати аналізу» наводяться загальні результати ідентифікації за перекладачами.

В загальному вихідними даними системи є:

- масив коефіцієнтів близькості кожного тексту до кожного автора за кожним параметром;
- масив з результатами ідентифікації кожного тексту за кожним параметром;

- масив із зальними кількостями ідентифікацій кожного перекладача за кожним параметром.

2.3 Опис структури бази даних

Система міститиме базу даних із такими таблицями: «Translator», «TrainText», «TestText», «AnalysisParam», «TrainDictionary».

В таблиці 2.1 наведено короткий опис кожної таблиці.

Таблиця 2.1 – Опис структур даних

Структура	Опис
Translator	Містить об’єкти перекладачів в моделі.
TrainText	Містить файли з текстами перекладачів для тренування моделі.
TestText	Містить тестові файли для проведення аналізу.
AnalysisParam	Містить параметри аналізу.
TrainDictionary	Містить результати тренування моделі – лінгвістичні властивості перекладачів за кожним параметром. Дані кожного перекладача за кожним параметром зберігаються в csv-файлах.

На рисунку 2.1 наведено схему бази даних.

Рисунок 2.1 – Схема бази даних

В таблицях 2.2-2.6 містяться детальні описи кожної таблиці.

Таблиця 2.2 – Детальний опис таблиці «Translator»

Поле	Тип	Опис
id	Integer <pk>	Унікальний номер запису в таблиці.
name	Varchar(200)	Ім’я перекладача

Таблиця 2.3 – Детальний опис таблиці «TrainText»

Поле	Тип	Опис
id	Integer <pk>	Унікальний номер запису в таблиці.
translator_id	Integer <fk>	Номер запису із таблиці з перекладачами.
file	Varchar(100)	Назва файлу у внутрішньому сховищі.

Таблиця 2.4 – Детальний опис таблиці «TestText»

Поле	Тип	Опис
id	Integer <pk>	Унікальний номер запису в таблиці.
file	Varchar(100)	Назва файлу у внутрішньому сховищі.

Таблиця 2.5 – Детальний опис таблиці «AnalysisParam»

Поле	Тип	Опис
id	Integer <pk>	Унікальний номер запису в таблиці.
name	Varchar(200)	Назва параметра
code	Integer	Код, що ідентифікує параметр в програмі.

Таблиця 2.6 – Детальний опис таблиці «TrainDictionary»

Поле	Тип	Опис
id	Integer <pk>	Унікальний номер запису в таблиці.
translator_id	Integer <fk>	Номер запису із таблиці з перекладачами.
analysisParam_id	Integer <fk>	Номер запису із таблиці з параметрами.
file	Varchar(100)	Назва файлу у внутрішньому сховищі.

Висновок до розділу

У даному розділі було розглянуто вхідні та вихідні дані програмного забезпечення. Описано структуру бази даних та її компоненти.

3 МАТЕМАТИЧНЕ ЗАБЕЗПЕЧЕННЯ

3.1 Змістовна постановка задачі

Нехай маємо набір перекладів, що були створені певними перекладачами. Візуально це можна побачити в таблиці 3.1.

Таблиця 3.1 – Представлення перекладів та перекладачів

Переклади	Перекладачі
Переклад 1	Перекладач 1
…	…
Переклад k-1	Перекладач 1
Переклад k	Перекладач 2
…	…
Переклад *l*	Перекладач 2
…	…
Переклад t	Перекладач m
…	…
Переклад n	Перекладач m

Задача полягає у тому, аби однозначно визначити, ким був здійснений переклад, що не належить вже наявному набору перекладів.

3.2 Математична постановка задачі

Нехай маємо множину перекладених текстів *D*, що складається із довільних текстів:

$$D = \{d_1, d_2, ...d_n\}, \quad (3.1)$$

де d_i – конкретний переклад, $i = \overline{1,n}$, n – кількість цих перекладів.

Нехай маємо множину перекладачів C, які здійснювали дані переклади:

$$C = \{c_1, c_2, ...c_m\}, \quad (3.2)$$

де $c_i, i = \overline{1,m}$ конкретний перекладач, m – їх кількість.

Множину перекладів D, кожен елемент якої ідентифікований до деякого перекладача із множини C, назвемо матрицею ідентифікацій позначатимемо як Ds.

$$Ds = \begin{pmatrix} d_1 & c_1 \\ \vdots & \vdots \\ d_{k-1} & c_1 \\ d_k & c_2 \\ \vdots & \vdots \\ d_l & c_2 \\ \vdots & \vdots \\ d_t & c_m \\ \vdots & \vdots \\ d_n & c_m \end{pmatrix} \quad (3.3)$$

Матриця ідентифікацій побудована за допомогою невідомої цільової функції F:

$$F : C \times D \to \{0,1\} \quad (3.4)$$

Задача ідентифікації перекладача полягає у тому, аби побудувати ідентифікатор F`, максимально наближений до F.

Ідентифікатором F` зветься визначена на множині перекладів D функція, яка однозначно зіставляє кожен переклад із множини D конкретному перекладачу із множини C.

3.3 Обґрунтування методу розв'язання

Метод аналізу n-грамів широко використовується в сфері комп'ютерної лінгвістики і є базовим. За допомогою цього методу проводиться ідентифікація автора тексту. Так як характеристики n-грамів перекладача залежать насамперед від словникового наповнення тексту, перевіривши ідентифікацію перекладача методом n-грамів можна визначити, наскільки словниковий запас перекладача впливає на словникове наповнення перекладеного твору.

Для наочності представимо графічне зображення відмінності між перекладачами при порівнянні статистичних даних з біграм.

На рисунку 3.1 показано відмінності між лінгвістичними особливостями О. Гижі та І. Хоменко, рис. 3.2 – О. Гижі та його перекладом книги «Буття». Статистичні дані отримано за допомогою створеного програмного забезпечення дипломного проекту. Вісь абсцис на графіках представляє собою сумарні значення частотностей кожних 10 біграм.

Рисунок 3.1 – Відмінності між біграмами О. Гижі та І. Хоменко

Рисунок 3.2 – Відмінності між біграмами О. Гижі та його перекладу книги «Буття»

Аналіз графіків показує, що функція щільності перекладача співпадає з функцією щільності його тексту. В той самий час функції щільності перекладачів відрізняються значно більше.

3.4 Опис методів розв'язання

Формалізована задача розпізнавання перекладача тексту може бути представлена у наступний спосіб.

Нехай є бібліотека текстів, яка представлена у вигляді щільності функції розподілу або статистиці n-грамів (грамів, біграмів, триграмів і т.д.) для А відомих авторів. Нехай K_a – кількість текстів а–го автора, $N_{i,a}$ – кількість символів в і-му тексті цього автора, де $i = 1,2,...,K_a$. Будемо вважати, що довжина кожного тексту достатня для проведення статистичного аналізу (тобто довжина тексту дає достатньо інформації про стиль автора в розрізі частоти зустрічі n-грамів).

Нехай $f_{i,a}(j)$ – функція щільності n-грамів відповідного тексту. де $j = 1,...,a(n) = 1,...,\Omega^n$ (Ω – розмір алфавіту або кількість символів, що досліджуються). Для кожного автора визначимо середньозважене значення щільності функції розподілу, нехтуючи відмінністю (одиницею) між грамами, біграмами і далі, внаслідок $N_a >> n$:

$$F_a(j) = \frac{1}{N_a}\sum_{i=1}^{K_a} f_{i,a}(j)N_{i,a}\,,\; N_a = \sum_{i=1}^{K_a} N_{i,a} \qquad (3.5)$$

Значення $F_a(j)$ – будемо вважати авторським еталоном.

Введемо «бібліотечну норму» ρ_{ik} як відстань між щільністю функцій розподілу текстів та в нормі підсумованих функцій

$$\rho_{ik} = \|f_i - f_k\| = \sum_{j=1}^{\alpha(n)} |f_i(j) - f_k(j)| \qquad (3.6)$$

Нехай в наявності є текст «0» невідомого автора, який необхідно ідентифікувати всередині даної бібліотеки. Автором тексту «0» вважається той з авторів «a», для якого норма різниці між щільністю функції розподілу тексту «0» та середньою авторською щільністю функції розподілу мінімальна:

$$\rho_a^0 = \|f_0 - f_a\|, \quad a^0 = \arg\min_a \rho_a^0 \tag{3.7}$$

Надалі в роботі норма різниці між щільністю функції розподілу тексту «0» та середньою авторською щільністю функції розподілу буде називатися «коефіцієнт близькості» або «близькість».

Висновок до розділу

В даному розділі було описано постановку задачі та детально описано запропонований метод її вирішення – аналіз n-грам.

Використання методу обґрунтовано на прикладі порівняння біграмів двох перекладачів та перекладу одного з них.

4 ПРОГРАМНЕ ТА ТЕХНІЧНЕ ЗАБЕЗПЕЧЕННЯ

4.1 Засоби розробки

Після проведення аналізу вимог до програмного продукту було вирішено реалізувати клієнт-серверне веб-застосування. Для цього було обрано наступні засоби розробки:

- середовище розробки: **PyCharm –** середовище із серії продуктів фірми JetBrains, в якому можна створювати застосування на основі мови Python.
- мова написання коду програми: **Python** – гнучка і легка в написанні коду мова. Містить багато готових бібліотек, що пришвидшує та спрощує написання коду;
- Python-фреймворк для створення веб-застосунків **Django** – використовує шаблон проектування MVC а також власний ORM, в якому модель даних описується мовою Python.
- СУБД SQLite – вбудована в Django система управління базами даних. Зберігає всі таблиці даних в одному файлі.
- мова гіпертекстової розмітки документів **HTML**;
- мова для опису зовнішнього вигляду веб-сторінок **CSS**;
- фреймворк для верстування сайтів **Bootstrap** – зручний фреймворк з готовими стилями для верстування сайтів;

У виборі мови програмування були альтернативи Java і C#. Проте було вибрано мову Python з наступних причин:

- наявність багатьох інструментів для роботи з текстовими даними, таких як зрізи, словники і т.і.;
- наявність додаткових бібліотек із готовими рішеннями різних задач, що спрощує реалізацію роботи з текстами;
- компактність коду.

4.2 Вимоги до технічного забезпечення

4.2.1 Загальні вимоги

Програмний продукт є веб-застосуванням, що складається з клієнта та сервера.

Для коректної роботи серверної частини веб застосування необхідно використовувати веб-сервер, що може бути представлений локальною обчислювальною машиною. Сервер мінімально повинен мати такі характеристики:

- процесор з тактовою частотою не нижче 2 ГГц;
- об’єм оперативної пам’яті (не менше 2 ГБ);
- жорсткий диск (не менше 40 ГБ);
- операційна система Windows 7 і вище;
- Python 3.5.
- Django 3.0

Для коректної роботи клієнтської частини веб застосування необхідно:

- підключення до мережі Інтернет.

4.2.2 Опис локальної обчислювальної мережі

Планується першу версію програмного забезпечення розробити для випадку, коли серверною частиною виступає локальна обчислювальна машина, на якій запускається програмний продукт. У цьому випадку для коректної роботи клієнтської частини застосування, локальна обчислювальна машина повинна мати такі самі параметри, що наведені вище в пункті 4.2.1.

4.3 Архітектура програмного забезпечення

4.3.1 Діаграма класів

Для реалізації логіки алгоритму ідентифікації перекладача розроблено клас TranslatorsModel, що використовує класи Translator, TrainText, TestText,

AnalysisParam, TrainDictionary, через які реалізується доступ до бази даних. Опис бази даних наведено в розділі 2.3 «Опис структури даних». Тому нижче описано лише клас TranslatorsModel.

Клас TranslatorsModel має такі атрибути:

- список перекладачів (translators), що є масивом об’єктів класу Translator;
- список тестових текстів (testTexts), що є масивом об’єктів класу TestText;
- список параметрів (analysisParams), що є масивом об’єктів класу AnalysisParam;

Клас TranslatorsModel має такі методи:

- канонізувати текст canonize();
- підрахувати частотності count_freq();
- підсумувати словник частотностей перекладача total_det_gram();
- створити словник кількостей зустрічань грамів в текстах createDictionary();
- записати словники частостей в csv-файли writeDictionary();
- очистити текст від пунктуації clean();
- підрахувати близькість тексту до перекладача difference();
- створити таблицю результатів аналізу analysis();
- записати результати в Excel-файл writeExcel();
- дістати тренувальний датасет перекладачів з файлів takeTrain();
- тренувати модель train();
- провести аналіз тестових файлів test();

На рисунку 4.1 наведено діаграму класів програмного забезпечення.

Рисунок 4.1 – Діаграма класів програмного забезпечення

4.3.2 Діаграма послідовності

Для відображення взаємодії між об’єктами в додатку була побудована структура послідовності, що наведена на рисунку 4.2.

В діаграмі послідовності описана взаємодія між користувачем, моделлю та базою даних.

Спочатку користувач вводить в модель дані про перекладача, через що модель вносить зміни в базу даних. Так само користувач завантажує тексти перекладачів в модель, і вони автоматично завантажуються і в базу даних.

Коли користувач дає запит на тренування моделі, запускається функція train(), що в кінці своєї діяльності завантажує до бази даних csv-файли із статистичними даними перекладачів.

Потім користувач завантажує модель тестові тексти, що автоматично завантажуються в базу даних.

Коли користувач дає запит на проведення аналізу, модель виконує функцію test(), модель повертає користувачу Excel-документ з результатами аналізу.

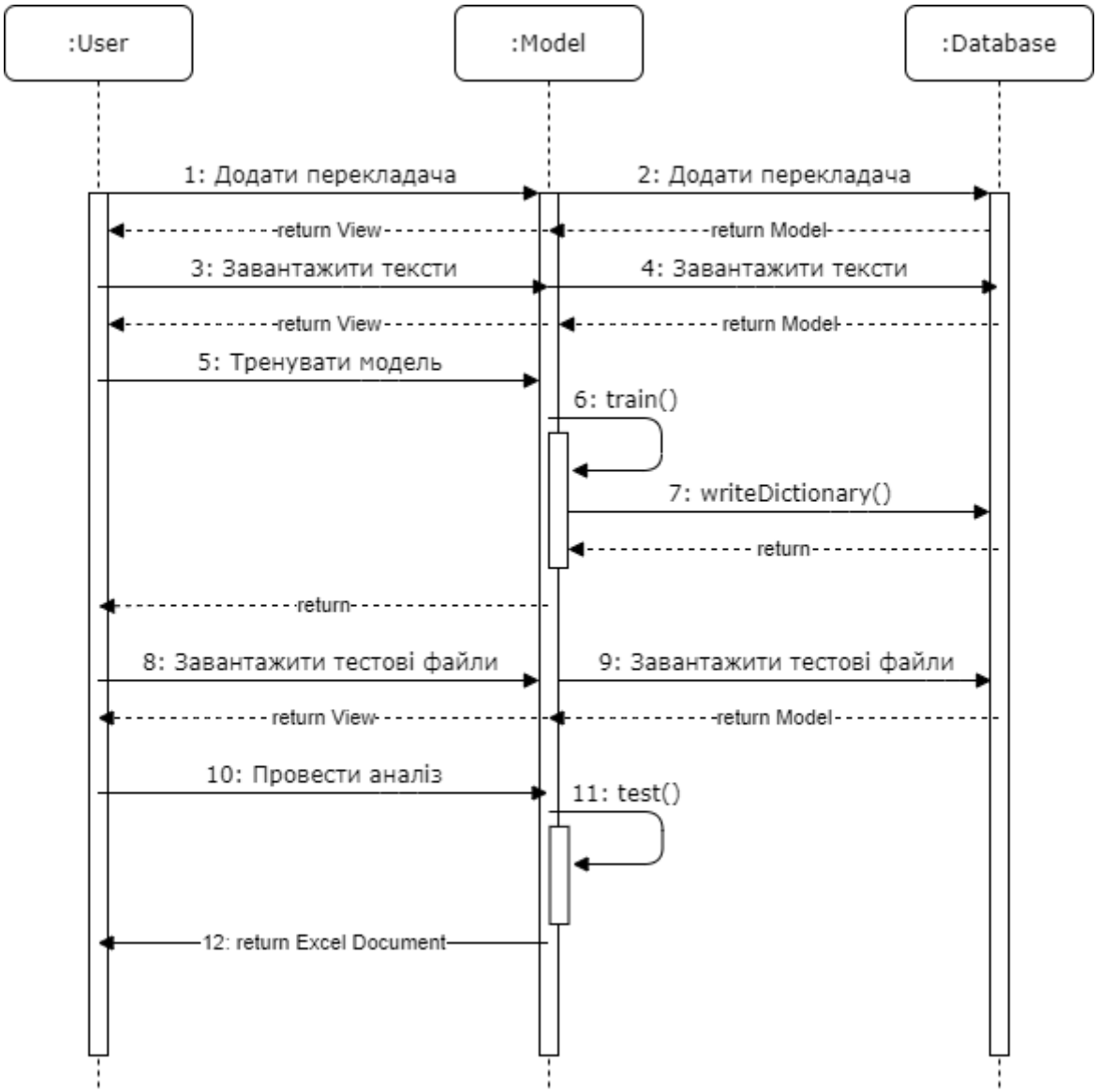


Рисунок 4.2 – Діаграма послідовності програмного забезпечення

4.3.3 Діаграма компонентів

Для відображення взаємодії між компонентами системи побудовано діаграму компонентів. З діаграми видно, що система містить такі компоненти:

Логіка (Logic), Модель (Model), Користувач (User), Веб-застосування (Web Application) та База Даних (Database).

Компонент Логіка (Logic) надає компоненту Модель (Model) необхідну інформацію про своє функціонування.

Компонент Модель (Model) надає компоненту Веб-астосування (Web Application) інформацію про створену модель.

Компонент Користувач (User) надає компоненту Веб-застосування (Web Application) вхідні дані.

Компонент База Даних (Database) збірігає інформацію про дані моделі та надає її компоненту Модель (Model).

Рисунок 4.3 – Діаграма послідовності програмного забезпечення

4.3.4 Специфікація функцій

Детальний опис всіх функцій для здійснення логіки програми представлено в таблиці 4.1. Так як єдиним класом реалізації логіки є клас TranslatorsModel, всі представлені в таблиці функції належать цьому класу

Таблиця 4.1 – Специфікація функцій програмного забезпечення

Назва функції	Опис
canonize(text)	Приймає на вхід текст та видаляє закінчення в словах. На вихід повертає текст з видаленими закінченнями.
count(text, length)	На вхід приймає текст та довжину граму. Повертає словник кількостей зустрічання грамів в тексті.
count_freq(res, mode)	Приймає на вхід масив словників кількостей зустрічання грамів в тексті та режим (може мати значення «test» фбо «train»). В режимі «train» функція розуміє, що це масив словників перекладача, а в «test» - масив словників тестових файлів і повертає масив словників підрахованих частостей.
total_det_gram(res)	Приймає на вхід масив словників грамів перекладача. Повертає сумуючі частотності грамів перекладача
createDictionary(texts, mode)	Приймає на вхід масив текстів та режим (може мати значення «test» фбо «train»). В режимі «train» функція розуміє, що це тексти перекладача, а в «test» - тестові файли і повертає масив словників підрахованих частостей по кожному параметру.
writeDictionary(translator, translator_data)	На вхід приймає перекладача та параметр аналізу. Записує узагальнений словник конкретного параметра конкретного перекладача у csv-файл.
clean(paths)	На вхід приймає масив шляхів до текстових файлів. Очищує вміст від пунктуації, переводить все в нижній регістр. Повертає масив текстів типу (string).

Продовження таблиці 4.1

Назва функції	Опис
difference(translator_dictionary, test_data)	На вхід приймає словник тренування перекладача та словник тестового файлу. На вихід подає коефіцієнт близькості.
analysis(test_data, train_data)	На вхід приймає тестові словники частотностей та словники частотностей перекладачів. Повертає таблицю результатів аналізу у формі масиву.
writeExcel(differences)	На вхід приймає масив результатів аналізу. Здійснює запис результатів аналізу в Excel-файл.
takeTrain()	Витягує з csv-файлів дані з лінгвістичними параметрами перекладачів.
train()	Запускає та скеровує тренування моделі.
test()	Запускає та скеровує процес проведення аналізу.

4.4 Опис звітів

Після проведення аналізу в браузер завантажується файл «Analysis.xlsx» з результатами аналізу. Цей файл містить дві таблиці: «Близькості» та «Результати аналізу».

В листі «Близькості» наводяться коефіцієнти близькості між тестовими текстами та стилями перекладачів за лінгвістичними параметрами. На рисунку 4.2 показано приклад листа «Близькості» при аналізі двох перекладачів та 4 тестових файлів.

Таблиця 4.2 – Приклад таблиці «Близькості»

Іван Огієнко				
	Євангелія_від_Матвія	Євангелія_від_Марка	Євангелія_від_Луки	Євангелія_від_Івана
Слова	0,758669	0,790999	0,798764	0,801774
3 грам	0,683736	0,693723	0,705587	0,725819
4 грам	0,806943	0,81112	0,832529	0,855306
5 грам	0,878836	0,883759	0,90258	0,921624
6 грам	0,918832	0,919548	0,940181	0,954694

Продовження таблиці 4.2

Пантелеймон Куліш				
	Євангелія_від_Матвія	Євангелія_від_Марка	Євангелія_від_Луки	Євангелія_від_Івана
Слова	0,776684	0,799832	0,812226	0,8126
3 грам	0,695951	0,701588	0,714919	0,731176
4 грам	0,815235	0,816595	0,83789	0,86118
5 грам	0,885394	0,885673	0,907379	0,927941
6 грам	0,925303	0,924244	0,944692	0,959746

Чим менший коефіцієнт близькості, тим більше вірогідність, що даний текст належить даному перекладачу.

В листі «Результати аналізу» наводяться дві таблиці. В першій таблиці наводяться результати ідентифікації кожного перекладача за кожним критерієм.

В таблиці 4.3 показано приклад таблиці результатів ідентифікації перекладачів текстів у листі «Результати аналізу».

Таблиця 4.3 – Приклад таблиці результатів ідентифікації перекладачів текстів у листі «Результати аналізу»

	Євангелія_від_Матвія	Євангелія_від_Марка	Євангелія_від_Луки	Євангелія_від_Івана
Слова	Іван Огієнко	Іван Огієнко	Іван Огієнко	Іван Огієнко
3 грам	Іван Огієнко	Іван Огієнко	Іван Огієнко	Іван Огієнко
4 грам	Іван Огієнко	Іван Огієнко	Іван Огієнко	Іван Огієнко
5 грам	Іван Огієнко	Іван Огієнко	Іван Огієнко	Іван Огієнко
6 грам	Іван Огієнко	Іван Огієнко	Іван Огієнко	Іван Огієнко

В другій таблиці листа «Результати аналізу» наводяться загальні результати ідентифікації за перекладачами.

В таблиці 4.4 показано приклад таблиці загальних результатів ідентифікації за перекладачами у листі «Результати аналізу».

Таблиця 4.4 – Приклад таблиці загальних результатів ідентифікації за перекладачами у листі «Результати аналізу»

Загальні результати ідентифікації:					
Загальна кількість текстів: 4					
	Слова	3 грам	4 грам	5 грам	6 грам
Іван Огієнко	4	4	4	4	4
Пантелеймон Куліш	0	0	0	0	0

Висновок до розділу

В даному розділі було розглянуто програмні засоби та мови програмування, за допомогою яких було реалізовано програмний продукт. Розглянуто загальні вимоги для технічних засобів.

Для кращого уявлення взаємодії об’єктів на фізичному рівні було побудовано діаграму компонентів.

Розроблено структурну схему класів, в якій представлено кожен клас, методи та поля в класі та взаємодію між класами.

Розроблено структурну схему діаграми послідовностей, на якій було показано послідовність взаємодій між користувачем, моделлю та базою даних на проміжку часу від запуску програми до отримання кінцевого результату.

5 ТЕХНОЛОГІЧНИЙ РОЗДІЛ

5.1 Керівництво користувача

Розроблене веб-застосування може бути розгорнуте на локальному або віддаленому сервері з доступом за допомогою браузеру.

В керівництві наведено порядок та можливі варіанти використання.

На рисунках 5.1 – 5.12 показаний інтерфейс програмного забезпечення та покроково розібрано процес введення даних, тренування моделі та аналізу.

5.1.1 Головна сторінка

Після переходу користувача на веб-адресу за доменом (у випадку встановлення на глобальний сервер) або за портом локального серверу, користувач попадає на головну сторінку застосування, яку показано на рисунку 5.1.

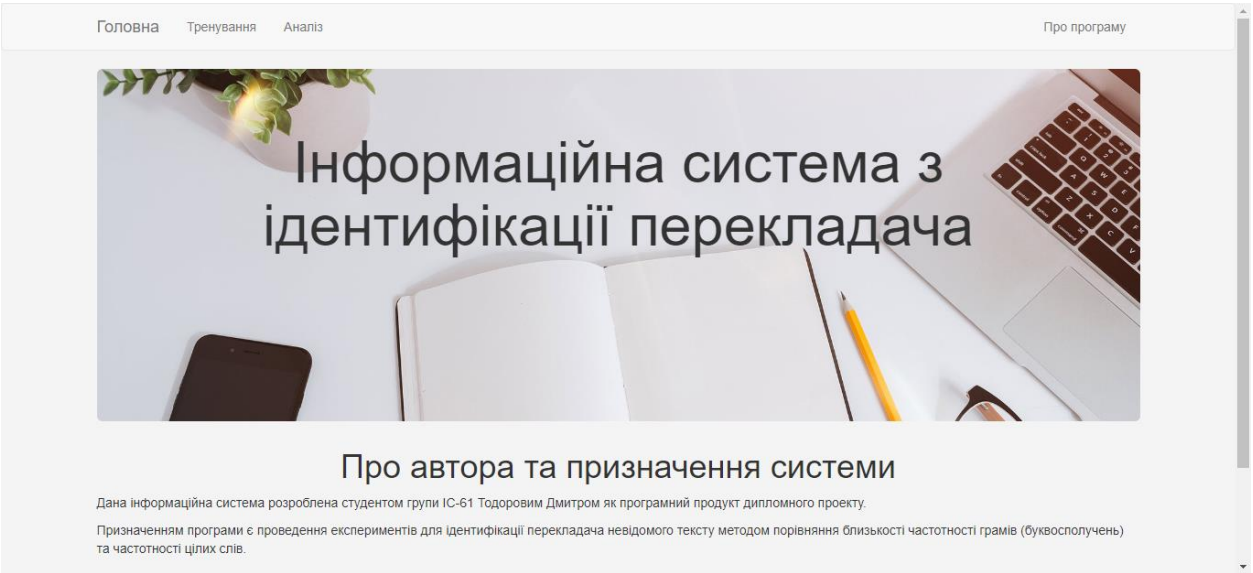


Рисунок 5.1 – Головна сторінка додатку

На головній сторінці розташовано меню із пунктами «Головна» для переходу на головну сторінку, «Тренування» для переходу на сторінку з

тренування моделі та «Аналіз» для проведення аналізу. Також в правому верхньому куті є посилання «Про програму».

5.1.2 Огляд покрокової інструкції

Для того, щоб більш детально ознайомитися із програмним забезпеченням, користувачеві пропонується вивчити покрокову інструкцію.

Щоб оглянути інструкцію, користувач повинен натиснути пункт меню «Про програму», після чого здійснюється перехід на сторінку, де буде детальна інформація по взаємодії із даним програмним забезпеченням.

На рисунку 5.2 зображено початкову інструкцію програмного забезпечення:

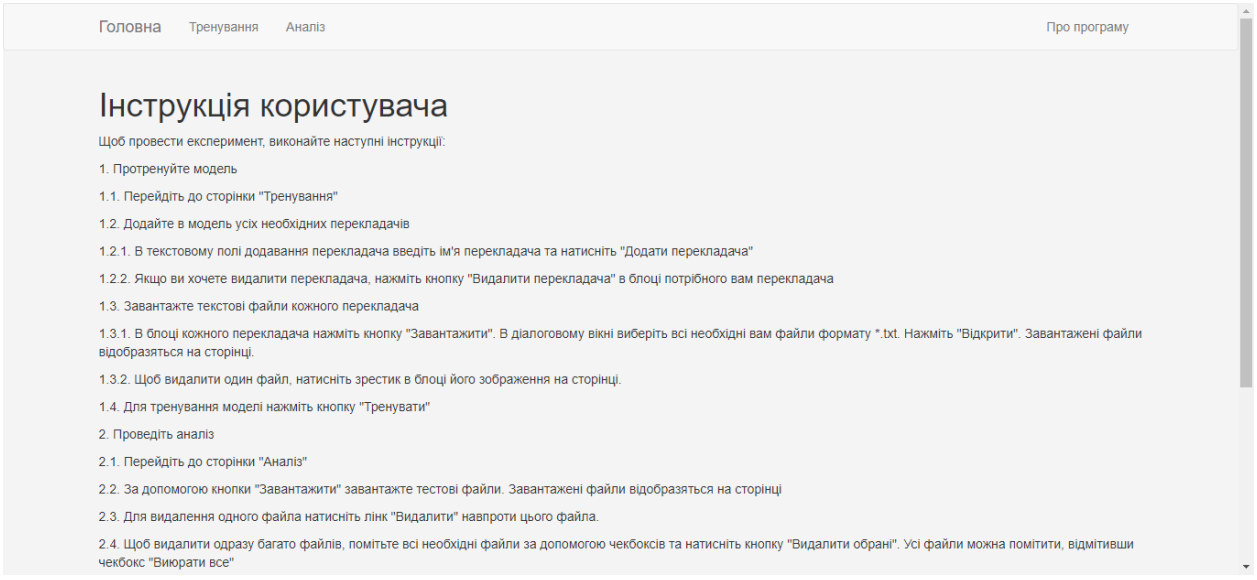


Рисунок 5.2 – Початкова інструкція

Таким чином, після прочитання інформації даної сторінки із допомогою, користувач може із легкістю починати роботу з програмою

5.1.3 Тренування моделі

При переході до сторінки «Тренування» за допомогою меню користувач бачить виходить на панель тренування моделі.

На рисунку 5.3 зображено сторінку «Тренування» при відсутності даних.

Рисунок 5.3 – Сторінка «Тренування» при відсутності даних

5.1.3.1 Додавання та видалення перекладача

Для того, щоб додати нового перекладача в модель, необхідно ввести його ім’я в текстове поле «Ім’я перекладача» та натиснути кнопку «Додати перекладача». Після натиснення кнопки на екрані з’явиться блок з введеним ім’ям.

Для того, щоб видалити перекладача разом з усіма його даними, необхідно натиснути «Видалити перекладача» в блоці відповідного перекладача.

На рисунку 5.4 показано сторінку з доданими в модель перекладачами.

Рисунок 5.4 – Сторінка «Тренування» із доданими перекладачами

5.1.3.2 Додавання та видалення перекладача

Щоб завантажити в модель переклади відповідного перекладача, необхідно в блоці цього перекладача натиснути кнопку «Завантажити файли». Після цього з’явиться стандартне діалогове вікно для завантаження багатьох файлів лише формату *.txt, яке показано на рисунку 5.5.

Рисунок 5.5 – Діалогове вікно для завантаження файлів

Після завантаження файлів з діалогового вікна, вони відобразяться в блоці перекладача, що показано на рисунку 5.6.

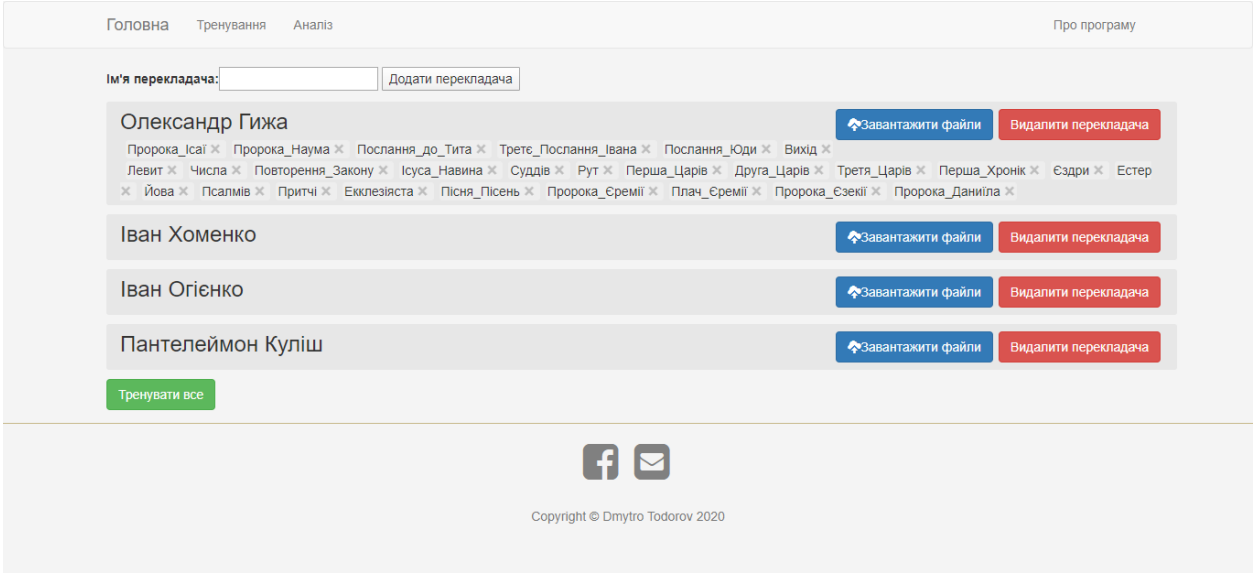


Рисунок 5.6 – Діалогове вікно для завантаження файлів

Для того, щоб видалити окремий файл, необхідно натиснути на хрестик навпроти назви цього файлу, що видно на рисунку 5.6.

5.1.3.3 Тренування моделі за введеними даними

Щоб тренувати модель, необхідно спочатку ввести дані, як це будо показано в пунктах 5.1.3.1 та 5.1.3.2, після чого натиснути кнопку «Тренувати».

5.1.4 Проведення аналізу

При натисненні на пункт меню «Аналіз» користувач попадає на сторінку проведення аналізу. Як

На рисунку 5.7 зображено сторінку «Аналіз» за відсутності даних. Як бачимо на рисунку, кнопка «Провести аналіз» заблокована, так як текстів для аналізу в систему ще не завантажено.

Рисунок 5.7 – Сторінка «Аналіз» за відсутності даних

5.1.4.1 Додавання та видалення текстів для аналізу

Щоб додати в систему тексти для проведення аналізу, необхідно натиснути кнопку «Завантажити файли». Тоді відкриється діалогове вікно для завантаження файлів, що зображене на рис. 5.5.

На рисунку 5.8 зображено вигляд сторінки «Аналіз» після завантаження файлів.

Рисунок 5.8 – Сторінка «Аналіз» із завантаженими файлами

Щоб видалити з вибірки один файл, достатньо натиснути «Видалити» навпроти назви файлу. Щоб видалити багато файлів, необхідно їх відмітити та натиснути кнопку «Видалити обрані». Щоб відмітити всі файли, достатньо відмітити «Вибрати все» в верху таблиці.

5.1.4.2 Проведення аналізу

Щоб провести аналіз, достатньо після вводу файлів натиснути кнопку «Аналіз». Після того, як система проведе аналіз, в браузері буде завантажено файл «Analysis.xlsx» з результатами аналізу.

На рисунку 5.9 ми можемо побачити завантаження файлу «Analysis.xlsx» в нижньому лівому куту.

Рисунок 5.9 – Завантаження файлу «Analysis.xlsx» з результатами аналізу

Файл «Analysis.xlsx» містить 2 листи: «Близькості» і «Результати ідентифікації»

На рисунку 5.10 показано вміст листа «Близькості», в якому наведено коефіцієнти близькості кожного твору тестової вибірки до кожного перекладача за критеріями n-грам. показано на рисунку 5.9

На рисунку 5.11 показано вміст листа «Результати аналізу», в якому наведено таблицю результатів ідентифікації кожного тексту за кожним критерієм а також загальні результати ідентифікації по кожному перекладачу.

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U
1	Іван Огієнко																				
2		Євангелія	Євангелія	Євангелія	Євангелія	Діяння_Сі	Перше_По	Друге_По	Послання	Послання	Послання	Послання	Перше_По	Друге_По	Перше_По	Друге_По	Послання	Послання	Послання	Перше_По	Друге
3	Слова	0,758669	0,790999	0,798764	0,801774	0,793783	0,783921	0,799747	0,813722	0,817375	0,806064	0,762948	0,819526	0,816569	0,718493	0,781941	0,795061	0,78094	0,822876	0,827061	0,845
4	3 грам	0,683736	0,693723	0,705587	0,725819	0,708205	0,736405	0,726995	0,742918	0,762357	0,722922	0,707511	0,770618	0,820031	0,651278	0,694675	0,718195	0,713745	0,786977	0,836568	0,853
5	4 грам	0,806943	0,81112	0,832529	0,855306	0,835262	0,861238	0,859318	0,865819	0,882155	0,856092	0,843342	0,910367	0,932593	0,786613	0,828242	0,849724	0,855056	0,883674	0,936458	0,951
6	5 грам	0,878836	0,883759	0,90258	0,921624	0,903166	0,927329	0,920643	0,919873	0,932272	0,921095	0,912867	0,962036	0,969308	0,868469	0,896676	0,913912	0,921054	0,940822	0,973551	0,975
7	6 грам	0,918832	0,919548	0,940181	0,954694	0,939357	0,961971	0,956554	0,952552	0,959517	0,960962	0,947758	0,98411	0,982905	0,915905	0,937343	0,950386	0,954389	0,971002	0,983012	0,990
8																					
9	Пантелеймон Куліш																				
10		Євангелія	Євангелія	Євангелія	Євангелія	Діяння_Сі	Перше_По	Друге_По	Послання	Послання	Послання	Послання	Перше_По	Друге_По	Перше_По	Друге_По	Послання	Послання	Послання	Перше_По	Друге
11	Слова	0,776684	0,799832	0,812226	0,8126	0,813871	0,801671	0,822637	0,834679	0,834112	0,820016	0,788059	0,842771	0,829518	0,76327	0,809253	0,805155	0,806645	0,839001	0,845375	0,857
12	3 грам	0,695951	0,701588	0,714919	0,731176	0,716964	0,74609	0,736382	0,747734	0,775613	0,722019	0,715705	0,775789	0,829012	0,671524	0,702879	0,725644	0,722726	0,800815	0,844078	0,861
13	4 грам	0,815235	0,816595	0,83789	0,86118	0,846004	0,872325	0,869984	0,871978	0,89189	0,860892	0,852384	0,911085	0,940507	0,802666	0,836873	0,859378	0,861397	0,898121	0,941849	0,956
14	5 грам	0,885394	0,885673	0,907379	0,927941	0,91293	0,937163	0,931652	0,930264	0,941292	0,927322	0,921169	0,962004	0,974154	0,884294	0,909628	0,926965	0,927857	0,953397	0,975668	0,983
15	6 грам	0,925303	0,924244	0,944692	0,959746	0,946921	0,968673	0,962801	0,959783	0,963826	0,966047	0,956047	0,983156	0,986331	0,928628	0,948434	0,962176	0,96178	0,977638	0,983392	0,99
16																					
17	Олександр Гижа																				
18		Євангелія	Євангелія	Євангелія	Євангелія	Діяння_Сі	Перше_По	Друге_По	Послання	Послання	Послання	Послання	Перше_По	Друге_По	Перше_По	Друге_По	Послання	Послання	Послання	Перше_По	Друге
19	Слова	0,776684	0,799832	0,812226	0,8126	0,813871	0,801671	0,822637	0,834679	0,834112	0,820016	0,788059	0,842771	0,829518	0,76327	0,809253	0,805155	0,806645	0,839001	0,845375	0,857
20	3 грам	0,695951	0,701588	0,714919	0,731176	0,716964	0,74609	0,736382	0,747734	0,775613	0,722019	0,715705	0,775789	0,829012	0,671524	0,702879	0,725644	0,722726	0,800815	0,844078	0,861
21	4 грам	0,815235	0,816595	0,83789	0,86118	0,846004	0,872325	0,869984	0,871978	0,89189	0,860892	0,852384	0,911085	0,940507	0,802666	0,836873	0,859378	0,861397	0,898121	0,941849	0,956
22	5 грам	0,885394	0,885673	0,907379	0,927941	0,91293	0,937163	0,931652	0,930264	0,941292	0,927322	0,921169	0,962004	0,974154	0,884294	0,909628	0,926965	0,927857	0,953397	0,975668	0,983
23	6 грам	0,925303	0,924244	0,944692	0,959746	0,946921	0,968673	0,962801	0,959783	0,963826	0,966047	0,956047	0,983156	0,986331	0,928628	0,948434	0,962176	0,96178	0,977638	0,983392	0,99

Рисунок 5.10 – Вміст листа «Близькості» файлу «Analysis.xlsx»

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U
1		Євангелія	Євангелія	Євангелія	Євангелія	Діяння_Сі	Перше_По	Друге_По	Послання	Послання	Послання	Послання	Перше_По	Друге_По	Перше_По	Друге_По	Послання	Послання	Послання	Перше_По	Друге
2	Слова	Іван Огіє	Іван Огіє	Іван Огіє	Іван Огіє	Іван Огіє	Іван Огіє	Іван Огіє	Іван Огіє	Іван Огіє	Іван Огіє	Іван Огіє	Іван Огіє	Іван Огіє	Іван Огіє	Іван Огіє	Іван Огіє	Іван Огіє	Іван Огіє	Іван Огіє	Іван О
3	3 грам	Іван Огіє	Іван Огіє	Іван Огіє	Іван Огіє	Іван Огіє	Іван Огіє	Іван Огіє	Іван Огіє	Іван Огіє	Іван Огіє	Пантелей	Іван Огіє	Іван Огіє	Іван Огіє	Іван Огіє	Іван Огіє	Іван Огіє	Іван Огіє	Іван Огіє	Іван О
4	4 грам	Іван Огіє	Іван Огіє	Іван Огіє	Іван Огіє	Іван Огіє	Іван Огіє	Іван Огіє	Іван Огіє	Іван Огіє	Іван Огіє	Іван Огіє	Іван Огіє	Іван Огіє	Іван Огіє	Іван Огіє	Іван Огіє	Іван Огіє	Іван Огіє	Іван Огіє	Іван О
5	5 грам	Іван Огіє	Іван Огіє	Іван Огіє	Іван Огіє	Іван Огіє	Іван Огіє	Іван Огіє	Іван Огіє	Іван Огіє	Іван Огіє	Іван Огіє	Пантелей	Іван Огіє	Іван Огіє	Іван Огіє	Іван Огіє	Іван Огіє	Іван Огіє	Іван Огіє	Іван О
6	6 грам	Іван Огіє	Іван Огіє	Іван Огіє	Іван Огіє	Іван Огіє	Іван Огіє	Іван Огіє	Іван Огіє	Іван Огіє	Іван Огіє	Іван Огіє	Пантелей	Іван Огіє	Іван Огіє	Іван Огіє	Іван Огіє	Іван Огіє	Іван Огіє	Іван Огіє	Іван О
7																					
8																					
9	Загальні результати ідентифікації:																				
10	Загальна кількість текстів: 23																				
11		Слова	3 грам	4 грам	5 грам	6 грам															
12	Іван Огіє	23	22	23	22	22															
13	Пантелей	0	1	0	1	1															
14	Олександ	0	0	0	0	0															
15	Іван Хоме	0	0	0	0	0															
16																					
17																					
18																					
19																					
20																					
21																					
22																					
23																					

Рисунок 5.11 – Вміст листа «Результати ідентифікації» файлу «Analysis.xlsx»

5.2 Випробування програмного продукту

5.2.1 Мета випробувань

Метою випробувань являється перевірка відповідності функцій комплексу задач інформаційно-аналітичної системи визначення лінгвістичних особливостей перекладача вимогам технічного завдання.

5.2.2 Загальні положення

Випробування проводяться на основі наступних документів:

– ГОСТ 34.603–92. Інформаційна технологія. Види випробувань автоматизованих систем;
– ГОСТ РД 50-34.698-90. Автоматизовані системи вимог до змісту документів.

5.2.3 Результати випробувань

За результатами тестів, була повністю перевірена правильна функціональність моделі. У таблицях нижче наведено перелік всіх випробувань. Було також наведено опис тестів.

Таблиця 5.1 – Ознайомлення із застосуванням

Дія:	Ознайомлення із застосуванням	
	Очікуваний результат	Результат тесту
Передумова		
Відкрити головну сторінку застосування в браузері	Користувач знаходиться на головній сторінці	пройдений

Продовження таблиці 5.1

Кроки тесту		
Натиснути пункт меню «Про програму»	Здійснюється перехід на сторінку «Про програму» із детальною інформацією про користування програмним забезпеченням	пройдений

Таблиця 5.2 – Додавання перекладача до моделі

Дія:	Додавання перекладача до моделі	
	Очікуваний результат	Результат тесту
Передумова		
Відкрити сторінку «Тренування»	Користувач знаходиться на сторінці тренування	пройдений
Кроки тесту		
Ввести ім’я автора в текстове поле	Ім’я автора відображається в текстовому полі	пройдений
Натиснути кнопку «Додати перекладача»	На сторінці з’являється блок із введеним ім’ям перекладача, кнопками «Завантажити файли» та «Видалити перекладача»	пройдений

Таблиця 5.3 – Видалення перекладача з моделі

Дія:	Видалення перекладача з моделі	
	Очікуваний результат	Результат тесту
Передумова		
Додати перекладача за тестом в таблиці 5.2	На сторінці відображається блок із введеним ім’ям перекладача, кнопками «Завантажити файли» та «Видалити перекладача»	пройдений
Кроки тесту		
Натиснути кнопку «Видалити перекладача» в потрібному блоці з перекладачем	Сторінка оновлюється, блок із вибраним перекладачем відсутній.	пройдений

Таблиця 5.4 – Додавання робіт перекладача

Дія:	Додавання робіт перекладача	
	Очікуваний результат	Результат тесту
Передумова		
Додати перекладача за тестом в таблиці 5.2	На сторінці відображається блок із введеним ім’ям перекладача, кнопками «Завантажити файли» та «Видалити перекладача»	пройдений
Передумова		

Продовження таблиці 5.4

Додати перекладача за тестом в таблиці 5.2	На сторінці відображається блок із введеним ім’ям перекладача, кнопками «Завантажити файли» та «Видалити перекладача»	пройдений
Кроки тесту		
Натиснути кнопку «Завантажити файли» в потрібному блоці з перекладачем	Відкривається діалогове вікно для завантаження файлів з обмеженням формату файлів тільки *.txt	пройдений
Виділити файли в діалоговому вікні та натиснути «Додати»	Діалогове вікно закривається. На сторінці в обраному блоці перекладача відображуються назви обраних файлів, поруч з кожною є кнопка видалення у вигляді хрестика.	пройдений

Таблиця 5.5 – Видалення роботи перекладача

Дія:	Видалення роботи перекладача	
	Очікуваний результат	Результат тесту
Передумова		
Додати файли за тестом в таблиці 5.4	На сторінці в обраному блоці перекладача відображуються назви обраних файлів, поруч з кожною є кнопка видалення у вигляді хрестика.	пройдений
Кроки тесту		
Натиснути хрестик поруч з одним з файлів	Файл видаляється з моделі і зникає зі сторінки	пройдений

Таблиця 5.6 – Тренування моделі

Дія:	Тренування моделі	
	Очікуваний результат	Результат тесту
Передумова		
Додати перекладачів та їх роботи в модель за тестами в таблицях 5.4 і 5.5	На сторінці відображаються блоки з перекладачами та відповідними їм текстовими файлами із вмістом робіт	пройдений
Кроки тесту		
Внизу натиснути кнопку «Тренувати все»	Проводиться тренування моделі. Створюються csv-таблиці з лінгвістичними параметрами перекладачів.	пройдений

Таблиця 5.7 – Додавання тестових файлів для аналізу

Дія:	Додавання файлів для аналізу	
	Очікуваний результат	Результат тесту
Передумова		
Відкрити сторінку «Аналіз»	Користувач знаходиться на сторінці «Аналіз»	пройдений
Кроки тесту		
Натиснути кнопку «Завантажити файли» в потрібному блоці з перекладачем	Відкривається діалогове вікно для завантаження файлів з обмеженням формату файлів тільки *.txt	пройдений

Продовження таблиці 5.7

Виділити файли в діалоговому вікні та натиснути «Додати»	Діалогове вікно закривається. На сторінці відображається таблиця з такими характеристиками: вверху таблиці чекбокс «Вибрати все», в кожній колонці зображено чекбокс, ім’я файлу та лінк «Видалити»	пройдений

Таблиця 5.8 – Видалення конкретного тестового файлу

Дія:	Видалення тестового файлу	
	Очікуваний результат	Результат тесту
Передумова		
Додати тестові файли за тестом в таблиці 5.7	На сторінці відображено таблицю із чекбоксами, іменами файлів та лінками «Видалити»	пройдений
Кроки тесту		
Натиснути лінк «Видалити» навпроти конкретного файлу	Рядок файлу в таблиці зникає.	пройдений

Таблиця 5.9 – Видалення багатьох тестових файлів

Дія:	Видалення багатьох тестових файлів	
	Очікуваний результат	Результат тесту
Передумова		
Додати тестові файли за тестом в таблиці 5.7	На сторінці відображено таблицю із чекбоксами, іменами файлів та лінками «Видалити»	пройдений

Продовження таблиці 5.9

Поставити помітку на необхідних файлах	Файл помічається галочкою	пройдений
При необхідності поставити помітку «Вибрати все»	Всі файли помічаються галочкою	пройдений
Кроки тесту		
Натиснути кнопку «Видаоити обрані»	Рядки з обраними файлами в таблиці зникають.	пройдений

Таблиця 5.10 – Проведення аналізу

Дія:	Проведення аналізу	
	Очікуваний результат	Результат тесту
Передумова		
Додати тестові файли за тестом в таблиці 5.7	На сторінці відображено таблицю із чекбоксами, іменами файлів та лінками «Видалити»	пройдений
Кроки тесту		
Натиснути кнопку «Провести аналіз»	В браузері завантажується файл під назвою «Analysis.xlsx» з результатами аналізу	пройдений
Післяумова		
Відкрити файл «Analysis.xlsx»	Файл відкрито, в ньому містяться листи «Близькості» та «Результати ідентифікації» з відповідними результатами аналізу	Пройдений

Висновок до розділу

В даному розділі було наведено інструкцію користувача.

За допомогою візуального відображення представлені екранні форми варіантів використання програмного забезпечення, проведено функціональне тестування додатку.

Інструкція містить інформацію, щодо здійснення тренування моделі та проведення аналізу ідентифікації перекладача.

6 ПРОВЕДЕННЯ ДОСЛІДЖЕНЬ

6.1 Мета досліджень

Нагадаємо, що на даний момент досліджень лінгвістичних параметрів перекладача не було знайдено. Тому, як було зазначено в розділі «Загальні положення. Цілі та задачі розробки» цілями створення системи є:

- визначення лінгвістичних особливостей, якими володіє перекладач;
- визначення можливості ідентифікації перекладача невідомого тексту.

Для досягнення цих цілей при створенні програмного продукту необхідно було створити датасет перекладачів для тренування та датасет творів для тестування, на основі датасетів провести експерименти з ідентифікації перекладача, за результатами експериментів зробити висновки щодо доцільності використання методу частотності грамів та цілих слів для досліджень у визначенні лінгвістичних особливостей перекладача.

6.2 Вхідні дані експериментів

Як зазначалося в розділі «Загальні положення. Цілі та задачі розробки», для проведення експериментів необхідно знайти перекладачів, якими було перекладано велику кількість творів, написаних різними авторами, при чому на кожний твір є переклади усіх перекладачів.

Знайти таку вибірку перекладачів в мережі інтернет досить важко. Проте, для проведення досліджень було вирішено взяти тексти Біблії.

Біблія – це зібрання з 66 різних книг, тому, коли ми аналізуємо текст Біблії, ми аналізуємо 66 різних книг. Кожен оригінал книги написано однією з трьох мов: іврит, арамейська, давньогрецька. Книги написано близько сорока різними авторами. Вони мають різні стилі: історичний, філософський, поетичний.

На сьогодні існує 5 повних перекладів Біблії з мов оригіналу на українську мову. Отже, виконавши експерименти над текстами Біблії, ми

зможемо визначити доцільність методів аналізу незалежно від автора, мови оригіналу, стилю. Єдиною спільною характеристикою усіх творів є релігійна тематика, але через різність часу написання та стилю за своїм словарним запасом ці книги все-таки відрізняються.

З п’яти повних перекладів Біблії українською мовою з загальних джерел в інтернеті вдалося дістати чотири переклади електронного формату. Це є переклади Олександра Гижі, Івана Хоменка, Івана Огієнка, Пантелеймона Куліша. Для проведення експерименту було вирішено для датасету тренування обрати тексти Старого Заповіту, що написані єврейською та арамейською мовами приблизно 30 різними авторами. В загальній сумі в Старому Заповіті 39 книг. Для датасету тестових текстів було обрано 27 текстів Нового Заповіту, написаних грецькою мовою сьома різними авторами. Такий розподіл текстів на тренувальну та тестову вибірку дозволить здійснити дослідження незалежно від мови оригіналу.

Вхідні дані експериментів описано в таблиці 6.1.

Таблиця 6.1 – Вхідні дані експериментів

	О. Гижа	І. Хоменко	І. Огієнко	П. Куліш
Тренування	39 текстів	39 текстів	39 текстів	39 текстів
Тестування	27 текстів	27 текстів	27 текстів	27 текстів

6.3 Опис експериментів

Отже, для тренування моделі було обрано по 39 текстів кожного автора.

Для проведення експериментів було взято по 27 текстів кожного автора та виконано їх ідентифікацію (тобто загалом було ідентифіковано 27*4 = 108 текстів).

В таблиці 6.2 показано результати експериментів. Значенню клітинки в таблиці відповідає кількість успішних ідентифікацій текстів перекладача відповідного рядка за параметром відповідного стовпця.

Таблиця 6.2 – Кількість успішних ідентифікацій текстів

	Слова	3 грам	4 грам	5 грам	6 грам	**Загально**
Олександр Гижа	19	25	27	27	27	**125**
Іван Хоменко	26	26	25	25	23	**125**
Іван Огієнко	4	2	5	8	12	**31**
Пантелеймон Куліш	25	26	26	24	24	**125**
Загально	**74**	**79**	**83**	**84**	**86**	

В останній колонці таблиці 6.2 представлена сума успішних ідентифікацій текстів перекладача відповідного рядка за всіма параметрами. В ній можемо бачити, що результати ідентифікації творів І. Огієнко приблизно в 4 рази гірші за результати ідентифікації творів інших перекладачів. Результати ідентифікації інших перекладачів приблизно однакові.

Тепер розрахуємо відсоток успішних ідентифікацій текстів для кожного параметра (він же є точністю алгоритму за параметром). Для цього загальну кількість успішних ідентифікацій розділимо на загальну кількість текстів (108 текстів текстів).

Точність ідентифікації для кожного параметра представлено в таблиці 6.3.

Таблиця 6.3 - Точність алгоритму залежно від параметра

	Слова	3 грам	4 грам	5 грам	6 грам
Точність	0,685185	0,731481	0,768519	0,777778	0,796296

На рисунку 6.1 показано гістограму точності алгоритму в залежності від параметра.

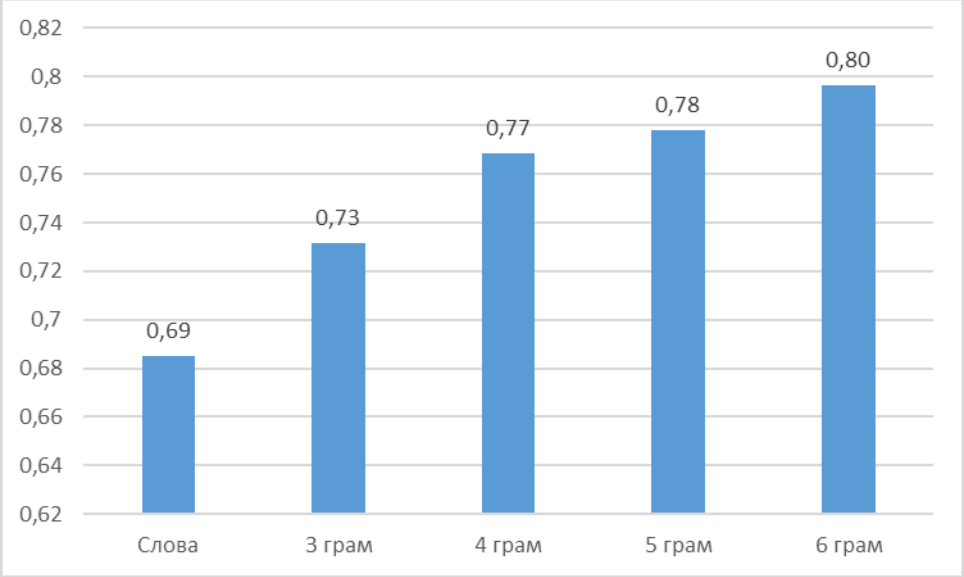


Рисунок 6.1 – Точність алгоритму за параметрами

На рисунку 6.1 ми бачимо, що в загальному випадку алгоритм є найбільш точним при використанні параметрів 4,5,6 грам.

Висновок до розділу

В результаті експериментів загальна середня точність ідентифікації, якщо рахувати за параметрами 4,5,6 грам, становила 77-80 %. Виявлено, що тексти І. Огієнко алгоритм розпізнав приблизно в 4 рази гірше, ніж інших перекладачів.

Загалом, на основі проведених досліджень можемо зробити висновки, що метод аналізу грамів можна застосовувати для ідентифікації перекладача. Так як метод аналізу грамів за суттю аналізує словниковий запас перекладача, то можемо зробити висновок, що одним з факторів, яким перекладач впливає на текст, є саме його словниковий запас.

ЗАГАЛЬНІ ВИСНОВКИ

Сьогодні в галузі комп'ютерної лінгвістики досліджено багато задач текстового аналізу, в тому числі задачі ідентифікації автора і задачі класифікації тексту. Проте задача ідентифікації перекладача невідомого перекладу ще не досліджена, тому можна сказати, що в даній роботі покладено початок дослідженням в розв'язанні цієї задачі.

В результаті виконання дипломного проекту було розроблено програмний продукт для проведення експериментів з дослідження лінгвістичних параметрів перекладача на основі методів аналізу грамів і цілих слів. Програмний продукт створений з можливістю вільно поповнювати базу творів перекладачів для тренування моделі. Надана можливість проводити експеримент одразу над багатьма текстами. Розроблено функціональність для створення звіту результатів ідентифікації.

В пояснювальній записці дипломного проекту було описано математичне обґрунтування запропонованого методу. Описано засоби розробки та функціональність програми. Розроблено інструкцію користувача та описано усі варіанти використання.

Крім всього, було створено дата сет з чотирьох перекладачів, кожен з яких переклав 66 творів різних авторів. На основі цих даних було проведено дослідження. В дослідженні для тренувальної моделі було взято по 39 творів з кожного перекладача, та для проведення ідентифікації – по 27 творів. Детально процес та результати досліджень описано в розділі «Проведення досліджень».

В результаті досліджень виявилося, що тексти трьох з чотирьох перекладачів було успішно ідентифіковано з імовірністю 93%, тексти четвертого – 44 %. Найкращі результати в загальному були отримані через аналіз 6 грамів (буквосполучень з 6 букв).

В результаті можемо сказати, що параметр грамів можна віднести до лінгвістичних параметрів перекладача, а метод аналізу грамів можна використовувати для ідентифікації перекладача. Але дані дослідження

потребують більшу вибірку перекладачів для експериментів, щоб отримати точніші результати. В подальшому для досліджень у цій сфері можна тестувати й інші вже відомі параметри ідентифікації, такі як довжина речень, пунктуація, морфологічний склад і т.д.

ПЕРЕЛІК ПОСИЛАНЬ

1. Хмелёв Д.В. Распознавание автора текста с использованием цепей А.А. Маркова // Вестн. МГУ. Сер. 9: Филология. 2000. № 2. С. 115–126. 19.

2. Хмелёв Д.В. Классификация и разметка текстов с использованием методов сжатия данных // Все о сжатии данных, изображений и видео. 2003. [Електронний ресурс] — Режим доступу: http://compression.ru/download/articles/classif/intro.html

3. Программные продукты и системы [Електронний ресурс] — Режим доступу http://www.swsys.ru/index.php?page=article&id=3703

4. Севбо И.П. Графическое представление синтаксических структур и стилистическая диагностика. Киев: Наук. дум., 1981. — 192 с.

5. Мартыненко Г.Я. Основы стилеметрии. Л.: ЛГУ, 1988. — 170 с.

6. Замекула О.І. Модифікований метод визначення авторства тексту на основі ланцюгів Маркова / Замекула О.І. // Національний Технічний Університет України «Київський Політехнічний Інститут імені Ігоря Сікорського» – Київ, 2019 – С. 16-24.

7. Юзьвак Д.Ю. Система класифікації текстів методами обробки природньої мови та машинного навчання / Юзьвак Д.Ю. // Національний Технічний Університет України «Київський Політехнічний Інститут імені Ігоря Сікорського» – Київ, 2019.

8. К.К. Боярский Введение в комьютерную лингвистику / К.К. Боярский // Санкт-Петербургский Национальный Исследовательский Университет информационных технологий, механики и оптики. – Санкт-Петербург, 2013.

9. Борисов Л.А., Орлов Ю.Н., Осминин К.П. Идентификация автора текста по распределению частот буквосочетаний // Препринты ИПМ им. М.В.Келдыша. 2013. № 27. 26 с.

10. Карпіловська Є.А. Вступ до прикладної лінгвістики: комп'ютерна лінгвістика: Підручник.— Донецьк: ТОВ «Юго-Восток, Лтд», 2006.— 188 с

11. Автоматическая обработка текстов на естественном языке и компьютерная лингвистика : учеб. пособие / Большакова Е.И., Клышинский Э.С., Ландэ Д.В., Носков А.А., Пескова О.В., Ягунова Е.В. — М.: МИЭМ, 2011. — 272 с.

12. Щипицина Л.Ю. Информационные технологии в лингвистике : учеб. Пособие / Л.Ю. Щипицина. — М. : ФЛИНТА : Наука, 2013. — 128 с.

13. Статистический анализ текстовой информации / Берман Н.Д., Левенец А.В., Сергеева Л.А. // Тихоокеанский государственный университет – 2016 – Сборник «Информационные технологии XXI века» – С. 282-286.

14. Django Documentation. – Режим доступу : https://docs.djangoproject.com/en/3.0/

15. Python Documentation. – Режим доступу : https://docs.python.org/3/

Додаток А

Інформаційно-аналітична система визначення лінгвістичних особливостей перекладача

(Найменування програми (документа))

DVD-R

(Вид носія даних)

36 арк, 95 Кб

(Обсяг програми (документа) , арк.,) Кб)

Київ – 2020 року

Додаток А

Файл models.py

```
from django.db import models
from django.db.models.signals import pre_delete
from django.dispatch.dispatcher import receiver
from django.core.files.storage import FileSystemStorage
import os
from django.conf import settings

class OverwriteStorage(FileSystemStorage):
    def get_available_name(self, name, max_length=None):
        if self.exists(name):
            os.remove(os.path.join(settings.MEDIA_ROOT, name))
        return name

class Translator(models.Model):
    translator_name = models.CharField("Ім'я перекладача", max_length=200,  unique=True)

    def __str__(self):
        return self.translator_name


class TrainText(models.Model):
    translator = models.ForeignKey(Translator, on_delete=models.CASCADE)
    file = models.FileField(default='some file')

    def __str__(self):
        position = self.file.name.find('-')
        dot = self.file.name.find('.txt')
        return self.file.name[position+1:dot] if position != -1 else self.file.name[:dot]


class TestText(models.Model):
    file = models.FileField(default='some file')

    def __str__(self):
        dot = self.file.name.find('.')
        return self.file.name[:dot]


class AnalysisParam(models.Model):
    name = models.CharField("Назва параметра", max_length=200)
    code = models.IntegerField("Код", default=0)

    def __str__(self):
        return self.name


class TrainDictionary(models.Model):
```

```
    translator = models.ForeignKey(Translator, on_delete=models.CASCADE)
    analysisParam = models.ForeignKey(AnalysisParam, on_delete=models.CASCADE)
    file = models.FileField(default='some file', storage=OverwriteStorage())

    def __str__(self):
        return self.file.name


class TestDictionary(models.Model):
    testText = models.ForeignKey(TestText, on_delete=models.CASCADE)
    analysisParam = models.ForeignKey(AnalysisParam, on_delete=models.CASCADE)
    file = models.FileField(default='some file', storage=OverwriteStorage())


@receiver(pre_delete, sender=TrainText)
def traintext_delete(sender, instance, **kwargs):
    # Pass false so FileField doesn't save the model.
    instance.file.delete(False)

@receiver(pre_delete, sender=TestText)
def testtext_delete(sender, instance, **kwargs):
    # Pass false so FileField doesn't save the model.
    instance.file.delete(False)

@receiver(pre_delete, sender=TrainDictionary)
def traindictionary_delete(sender, instance, **kwargs):
    # Pass false so FileField doesn't save the model.
    instance.file.delete(False)

@receiver(pre_delete, sender=TestDictionary)
def testdictionary_delete(sender, instance, **kwargs):
    # Pass false so FileField doesn't save the model.
    instance.file.delete(False)
```

Файл views.py

```
import os


from django.http import HttpResponseRedirect, HttpResponse
from django.views.generic.edit import FormView
from .forms import FileFieldForm, TranslatorForm, transForm, TestTextForm,
TrainTextForm
from django.shortcuts import render, get_object_or_404
from .models import Translator, TrainText, TestText, TrainDictionary
from .logic import Train
```

```
import mimetypes
from django.core.files.storage import FileSystemStorage

def home(request):
    return render(request, 'home.html')

def index(request):
    # translatorform(request)
    translatorForm = transForm()
    translator_list = Translator.objects.all()
    train_texts = TrainText.objects.all()
    test_texts = TestText.objects.all()
    return render(request, 'main.html',
                  {'translator_list': translator_list, 'train_texts': train_texts, 'test_texts':
test_texts,
                   'translatorForm': translatorForm})


def analysis(request):
    translator_list = Translator.objects.all()
    test_texts = TestText.objects.all()
    trainDictionary = TrainDictionary.objects.all()
    return render(request, 'analysis.html',
                  {'translator_list': translator_list, 'test_texts': test_texts, 'trainDictionary':
trainDictionary})


def makeAnalysis(request):
    if request.method == 'POST':
```

```
        data = request.POST.copy()
        print(data)
        translator_list = Translator.objects.all()
        checked_translator_list = []
        for translator in translator_list:
            if str(translator) in data.getlist('translators'):
                checked_translator_list.append(translator)
        test_texts_list = TestText.objects.all()
        checked_test_texts_list = []
        for test_text in test_texts_list:
            if str(test_text) in data.getlist('test-texts'):
                checked_test_texts_list.append(test_text)

        test = Train(translator_list,test_texts_list)
        file_path = test.test()
        if os.path.exists(file_path) and file_path != 0:
            with open(file_path, 'rb') as fh:
                response = HttpResponse(fh.read(), content_type="application/vnd.ms-
excel")
                response['Content-Disposition'] = 'inline; filename=' +
os.path.basename(file_path)
                return response
    return HttpResponse("<script>alert( 'Привет' );</script>")

def addTranslator(request):
    if request.method == 'POST':
        form = TranslatorForm(request.POST)
        if form.is_valid():
            name = form.cleaned_data['name']
```

```
    if request.method == 'POST':
        form = transForm(request.POST)
        if form.is_valid():
            form.save()
    return index(request)


def deleteTranslator(request, tr_id):
    translator_obj = get_object_or_404(Translator, id=tr_id)
    translator_obj.delete()
    if request.method == 'POST':
        form = TranslatorForm(request.POST)
        if form.is_valid():
            name = form.cleaned_data['name']
    return index(request)


def about(request):
    return render(request, 'about.html')


def uploadTestTexts(request):
    if request.method == "POST":
        files = request.FILES.getlist('document')
        for f in files:
            form = TestTextForm()
            instance = form.save(commit=False)
            instance.file = f
```

```
            instance.save()

    return analysis(request)


def uploadTranslatorFiles(request, tr_id):
    print(tr_id)
    translator_obj = get_object_or_404(Translator, id=tr_id)
    if request.method == "POST":
        files = request.FILES.getlist('document')
        for f in files:
            form = TrainTextForm()
            instance = form.save(commit=False)
            instance.file = f
            instance.file.name = translator_obj.translator_name + "-" +
instance.file.name
            instance.translator = translator_obj
            instance.save()

    return index(request)


def deleteTranslatorFile(request, file_id):
    file_obj = get_object_or_404(TrainText, id=file_id)
    file_obj.delete()
    if request.method == 'POST':
        form = TranslatorForm(request.POST)
        if form.is_valid():
            file = form.cleaned_data['file']
```

```
    return index(request)


def deleteTestFile(request, file_id):
    file_obj = get_object_or_404(TestText, id=file_id)
    file_obj.delete()
    if request.method == 'POST':
        form = TranslatorForm(request.POST)
        if form.is_valid():
            file = form.cleaned_data['file']

    return analysis(request)
def deleteCheckedTestFiles(request):
    data = request.POST.copy().getlist('selected[]')
    print(data)

    for file_id in data:
        print(int(file_id))
        file_obj = get_object_or_404(TestText, id=int(file_id))
        file_obj.delete()
    return analysis(None)

def train(request):
    if request.method == "POST":
        print("Start")
        translator_list = Translator.objects.all()
        trains = Train(translator_list,None)
        trains.train()
```

```
    return index(request)
```

Файл urls.py

```
from django.urls import path
from . import views
from django.conf.urls import url
from django.conf import settings
from django.conf.urls.static import static

urlpatterns = [
    path('', views.home, name='home'),
    path('train-model/', views.index, name='index'),
    path('analysis/', views.analysis, name='analysis'),
    path('about/', views.about, name='about'),
    url(r'add-translator/', views.addTranslator, name='addTranslator'),
    url(r'delete-translator/(?P<tr_id>[\d]+)', views.deleteTranslator,
name='deleteTranslator'),
    url(r'delete-translator-file/(?P<file_id>[\d]+)', views.deleteTranslatorFile,
name='deleteTranslatorFile'),
    url(r'delete-test-file/(?P<file_id>[\d]+)', views.deleteTestFile,
name='deleteTestFile'),
    url(r'upload_translator_files/(?P<tr_id>[\d]+)', views.uploadTranslatorFiles,
name='upload_translator_files'),
    url(r'upload/', views.uploadTestTexts, name='upload'),
    url(r'train/', views.train, name='train'),
    url(r'make-analysis/', views.makeAnalysis, name='makeAnalysis'),
    url(r'^deleteCheckedTestFiles/$', views.deleteCheckedTestFiles)
]
#if settings.DEBUG:
```

```
#urlpatterns += static(settings.MEDIA_URL, document_root=settings.MEDIA_ROOT)
```

Файл logic.py

```
import os
import xlsxwriter
from .models import Translator, TrainText, TestText, AnalysisParam, TestDictionary, TrainDictionary, TotalNumEl
import csv
from django.core.files import File


class Train:
    translators = []
    totals = dict()
    total_name = "_Total"
    Exceldocument = "Analysis.xlsx"
    testTexts = []
    analysisParams = None
    stopwords = []

    def __init__(self, translators, testTexts):
        self.translators = translators
        self.testTexts = testTexts
        self.analysisParams = AnalysisParam.objects.all().order_by('code')
        self.stopwords = []
        with open("trained/stop_words.txt", "r") as f:
            for line in f.readlines():
                self.stopwords.append(line.rstrip('\n'))
```

```
def stop_words(self, text):

    words = text.split()
    without_stop_text = ""
    for stopword in self.stopwords:
        while stopword in words:
            words.remove(stopword)
    for word in words:
        without_stop_text += word + " "
    '''
    words = text.replace('\n',' ')
    for stop_word in self.stop_words:
        while stop_word + ' ' in words:
            words.replace(stop_word + ' ', ' ')
    while '  ' in words:
        words.replace('  ',' ')
        '''
    return without_stop_text

def canonize(self, text):
    n = 8
    endings = [[]]
    with open("trained/endings.txt", "r") as f:
        for i in range(n):
            endings.append([])
        for line in f.readlines():
            endings[len(line) - 2].append(line.rstrip('\n'))
    words = text.split()
```

```
        canonized_text = ""
        for word in words:
            if word[-4:] == "ться":
                word = word[0:-4]
            if word[-3:] == "тся":
                word = word[0:-3]
            if word[-2:] == "сь" or word[-2:] == "ся" or word[-2:] == "ть":
                word = word[0:-2]
            for i in range(n, -1, -1):
                if len(word) > i + 2:
                    for end in endings[i]:
                        if word[-i - 1:] == end:
                            word = word[0:-i - 1]
            canonized_text += word + " "
        return canonized_text


    def count(self, text, length):
        res = dict()
        if length == 0:
            text = self.canonize(text)
            for key in text.replace('\n', ' ').split():
                if key in res:
                    res[key] += 1
                else:
                    res[key] = 1
        else:
            alphabet = set("йцукенгшщзхїфівапролджєячсмитьбюґ")
            for i in range(0, len(text) - length + 1):
                key = text[i:i + length]
```

```
            if set(key) < alphabet:
                if key in res:
                    res[key] += 1
                else:
                    res[key] = 1
        total = 0
        for key in res.keys():
            total += res[key]
        res[self.total_name] = total
        return res

    def count_freq(self, res, mode):
        if mode == "train":
            for key in res.keys():
                if not key == self.total_name:
                    res[key] *= 1 / res[self.total_name]
        elif mode == "test":
            for col in range(len(res)):
                for key in res[col].keys():
                    if not key == self.total_name:
                        res[col][key] *= 1 / res[col][self.total_name]
        return res

    def total_det_gram(self, res):
        max_length = 1000
        total_gram = dict()
        for col in range(len(res)):
            for key in res[col].keys():
                if key in total_gram:
```

```
                total_gram[key] += res[col][key]
            else:
                total_gram[key] = res[col][key]
        '''
        sorted_keys = [i[0] for i in sorted(total_gram.items(), key=lambda item: item[1], reverse=True)]
        for i in range(max_length,len(sorted_keys)):
            del total_gram[sorted_keys[i]]
        '''
        #total_gram[self.total_name] *= 2
        return total_gram

    def createDictionary(self, texts, mode):
        data = []
        for sh_i in self.analysisParams:
            print("Started ", sh_i)
            res = []
            for text in texts:
                res.append(self.count(text, sh_i.code))
            if mode == "train":
                res = self.total_det_gram(res)

            total_res = self.count_freq(res,mode)
            data.append(total_res)
        print("Finished dict")
        return data

    def writeDictionary(self, translator, translator_data):
        for num,analysisParam in enumerate(self.analysisParams):
```

```
            total_res = translator_data[num]
            analysisParamCode = analysisParam.code
            print("Started" + str(analysisParamCode))
            #analysisParam = AnalysisParam.objects.get(code=analysisParamCode)
            path = "trained/" + translator.translator_name + " " + analysisParam.name +
".csv"
            w = csv.writer(open(path, "w", newline=''))
            i = 0
            for key, val in total_res.items():
                i += 1
                w.writerow([key, val])
            print(i, path)
            obj, created = TrainDictionary.objects.get_or_create(translator=translator,
analysisParam=analysisParam)
            w = csv.reader(open(path, "r", newline=''))
            if created:
                obj.file = File(open(path, 'r'))
                obj.save()
            else:
                obj.file.save(path, File(open(path, 'r')))
                print(obj.file.name)


    def clean(self, paths):
        english                                                                    =
"qwertyuioplkjhgfdsazxcvbnmQWERTYUIOPLKJHGFDSAZXCVBNM"
        texts = []
        for path in paths:
            res = ""
            with open(path, "r", encoding="UTF-8") as f:
```

```
            for line in f.readlines():
                if not line.isupper():
                    line = line.replace(chr(65279), '')
                    line = line.replace(chr(110), '')
                    line = line.replace(chr(1105), '')
                    for s in "1234567890!@#$%^&*()-=,:./;'\"?><↑…[]«»”“„–|—`’`—":
                        while s in line:
                            line = line.replace(s, '')
                    while "°" in line:
                        line = line.replace('°', 'ї')
                    while "i" in line:
                        line = line.replace('i', 'і')
                    while "c" in line:
                        line = line.replace('c', 'с')
                    for el in english:
                        while el in line:
                            line = line.replace(el, '')
                    while "  " in line:
                        line = line.replace("  ", " ")
                    if not ("Глава" in line and len(line) < 9):
                        res += line[:-1]
            texts.append(res.lower())
        return texts


    def difference(self, translator_dictionary, test_data): # на входе словари конкретного грамма
        results = []
        for test_dictionary in test_data:
            total = 0
```

```
            for key in test_dictionary.keys():
                if not key in translator_dictionary.keys():
                    translator_dictionary[key] = 0
            for key in translator_dictionary.keys():
                if not key == self.total_name:
                    if key in test_dictionary.keys():
                        total += abs(test_dictionary[key] - translator_dictionary[key])
                    #else:
                        #total += abs(test_dictionary[key])
            results.append(total)
        return results

    def analysis(self, test_data, train_data):
        differences = []
        for translator_dictionary in train_data:
            gram_translator = []
            for id,sh_i in enumerate(self.analysisParams):
                gram_translator.append(self.difference(translator_dictionary[id],
test_data[id]))
            differences.append(gram_translator)
        return differences

    def writeExcel(self, differences):
        booklist = [text.file.name[:text.file.name.find('.')] for text in self.testTexts]
        translate_list = [translator.translator_name for translator in self.translators]
        workbook = xlsxwriter.Workbook(self.Exceldocument)
        worksheets = []
        worksheets.append(workbook.add_worksheet("Близькості"))
        for tr_id, translator in enumerate(translate_list):
```

```
            start_row = tr_id * (len(self.analysisParams) + 3)
            worksheets[-1].write(start_row, 0, translator)
            for col in range(len(booklist)):
                worksheets[-1].write(start_row + 1, col + 1, booklist[col])
            for id,sh_i in enumerate(self.analysisParams):
                worksheets[-1].write(start_row + 2 + id, 0, sh_i.name)
                for col in range(len(booklist)):
                    worksheets[-1].write(start_row + 2 + id, col + 1,
differences[tr_id][id][col])
        worksheets.append(workbook.add_worksheet("Результати ідентифікації"))
        min_results = [[1 for i in booklist] for i in self.analysisParams]
        determined_translator = [["" for i in booklist] for i in self.analysisParams]
        for tr_id, translator in enumerate(translate_list):
            for id,sh_i in enumerate(self.analysisParams):
                for col in range(len(booklist)):
                    if differences[tr_id][id][col] < min_results[id][col]:
                        min_results[id][col] = differences[tr_id][id][col]
                        determined_translator[id][col] = translator
        for col in range(len(booklist)):
            worksheets[-1].write(0, col + 1, booklist[col])
        tr_results = []
        for id, sh_i in enumerate(self.analysisParams):
            tr_dict = dict()
            for translator in translate_list:
                tr_dict[translator] = 0
            tr_results.append(tr_dict)

        for id,sh_i in enumerate(self.analysisParams):
            worksheets[-1].write(id + 1, 0, sh_i.name)
```

```
            for col in range(len(booklist)):
                worksheets[-1].write(id + 1, col + 1, determined_translator[id][col])
                tr_results[id][determined_translator[id][col]] += 1
        start_row = len(self.analysisParams) + 3
        worksheets[-1].write(start_row, 0, "Загальні результати ідентифікації:")
        worksheets[-1].write(start_row + 1, 0, "Загальна кількість текстів: " +
str(len(booklist)))
        for tr_id, translator in enumerate(translate_list):
            worksheets[-1].write(start_row + 3 + tr_id, 0, translator)
            for id, sh_i in enumerate(self.analysisParams):
                worksheets[-1].write(start_row + 2, id + 1, sh_i.name)
                worksheets[-1].write(start_row + 3 + tr_id, id + 1,
tr_results[id][translator])
        workbook.close()


    def takeTrain(self):
        translatorRes = []
        for translator in self.translators:
            analysisParamRes = []
            for analysisParam in self.analysisParams:
                dictionary = dict()
                trainDictionary = TrainDictionary.objects.get(translator=translator,
analysisParam=analysisParam)
                with open(trainDictionary.file.path, 'r', newline='\n') as fh:
                    rd = csv.reader(fh, delimiter=',')
                    for row in rd:
                        dictionary[row[0]] = float(row[1])
                analysisParamRes.append(dictionary)
            translatorRes.append(analysisParamRes)
```

```
        return translatorRes


    def train(self):
        for translator in self.translators:
            print(translator)
            trainTexts = TrainText.objects.filter(translator=translator)
            paths = [text.file.path for text in trainTexts]
            texts = self.clean(paths)
            translator_data = self.createDictionary(texts,"train")
            self.writeDictionary(translator, translator_data)


    def test(self):
        if self.translators != [] and self.testTexts != []:
            #self.analysisParams = AnalysisParam.objects.all().order_by('code')
            paths = [text.file.path for text in self.testTexts]
            texts = self.clean(paths)
            texts_dictionary = self.createDictionary(texts, "test")
            print("Анализировал тексты")
            translators_data = self.takeTrain()
            print("Считал csv")
            differences = self.analysis(texts_dictionary, translators_data)
            print("Провел анализ")
            self.writeExcel(differences)
            print("Записал в эксель")
            return self.Exceldocument
        return 0
```

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
“КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ імені ІГОРЯ СІКОРСЬКОГО”
Кафедра автоматизованих систем обробки інформації і управління

УЗГОДЖЕНО
Керівник проєкту
_____ *Олексій Фіногенов*
(підпис) (вл. ім’я, прізвище)
“13” квітня 2020 р.

ЗАТВЕРДЖУЮ
В.о. завідувача кафедри
_____ *Олександр ПАВЛОВ*
(підпис) (вл. ім’я, прізвище)
“14” квітня 2020 р.

«Інформаційно-аналітична система визначення лінгвістичних параметрів перекладача»

ТЕХНІЧНЕ ЗАВДАННЯ

Шифр *ДП 6126.01.000 ТЗ*

на 9 сторінках

Київ – 2020 року

ЗМІСТ

1 ЗАГАЛЬНІ ПОЛОЖЕННЯ

1.1 Повне найменування системи та її умовне позначення

Повна назва системи: Інформаційно-аналітична система визначення лінгвістичних параметрів перекладача.

Умовне позначення: TranslatorAnalysis.

1.2 Найменування організації-замовника та організацій-учасників робіт

Замовником проекту є кафедра Автоматизованих систем обробки інформації та управління НТУУ «КПІ ім. Ігоря Сікорського». Представником замовника є Олійник Юрій Олександрович.

Розробником системи є студент факультету інформатики та обчислювальної техніки НТУУ «КПІ ім. Ігоря Сікорського»: студент групи ІС-61 Тодоров Дмитро Віталійович.

1.3 Перелік документів, на підставі яких створюється система

При розробці системи і створенні проектно-експлуатаційної документації Виконавець повинен керуватися вимогами наступних нормативних документів:

- ДСТУ 19.201-78. Технічне завдання. Вимоги до змісту і оформлення;
- ДСТУ 34.601-90. Комплекс стандартів на автоматизовані системи. Автоматизовані системи. Стадії створення;
- ДСТУ 34.201-89. Інформаційні технології. Комплекс стандартів на автоматизовані системи. Види, комплексність і позначення документів при створенні автоматизованих систем.

1.4 Планові терміни початку і закінчення роботи зі створення системи

Плановий термін початку роботи над створенням системи – 13 квітня 2020 року.

Плановий термін по закінченню роботи над створенням системи – не пізніше 1 червня 2020 року.

2 ПРИЗНАЧЕННЯ І МЕТА СТВОРЕННЯ КОМПЛЕКСУ ЗАДАЧ

2.1 Призначення системи

Призначенням системи є визначення лінгвістичних особливостей перекладача, за допомогою яких можна було б ідентифікувати перекладача невідомого тексту.

2.2 Цілі створення системи

Цілями розробки системи є:

- визначення лінгвістичних особливостей, якими володіє перекладач;
- визначення можливості ідентифікації перекладача невідомого тексту.

Для досягнення поставлених цілей необхідно реалізувати наступні задачі:

- знайти перекладачів, якими було перекладано велику кількість творів, написаних різними авторами, при чому на кожний твір є переклади усіх перекладачів;
- створити датасет перекладів цих творів;
- реалізувати алгоритм визначення частотності грамів;
- реалізувати алгоритм визначення «мішку слів»;
- проаналізувати створений датасет за розробленими алгоритмами, за проведеним аналізом визначити лінгвістичні параметри досліджуваних перекладачів;
- провести множину експериментів за таким принципом: взяти текст одного з перекладачів, що не увійшов у датасет та провести ідентифікацію перекладача цього тексту;
- за результатами ідентифікації зробити висновки щодо доцільності використання даних методів для досліджень у визначенні лінгвістичних особливостей перекладача.

3 ХАРАКТЕРИСТИКА ОБ'ЄКТА АВТОМАТИЗАЦІЇ

Щоб використовувати систему, її необхідно встановити на глобальний або локальний сервер. В разі встановлення системи на глобальний сервер, користувачу необхідно мати доступ до мережі інтернет.

Працювати з системою можна за допомогою будь-якого браузера. Після того, як користувач створив систему, він має доступ до функціоналу системи, має змогу створювати базу перекладачів, завантажувати файли з перекладами а також файли для тестування.

Об'єктом автоматизації є задача ідентифікації перекладача невідомого тексту.

4 ВИМОГИ ДО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

4.1 Вимоги до функціональних характеристик

Система має виконувати наступні функції:

- додавання та видалення перекладачів;
- додавання та видалення файлів з текстами перекладачів
- додавання та видалення тестових файлів для ідентифікації перекладача тексту, що міститься в файлі;
- тренування моделі;
- проведення аналізу для визначення результатів ідентифікації.

Вимоги до надійності

Програма повинна відповідати всім фукнціональни вимогам, вказаним вище.

Програма повинна відповідати користувачеві на некоректно введені дані, видаючи на головний екран відповідне повідомлення.

При збоях в роботі апаратних засобів, програма повинна коректно завершити роботу.

Продукт водночас повинен бути і надійним і функціональним. У разі виникнення збоїв в програмі, потрібно сповіщати користувача, вказавши причину збою та шляхи вирішення проблеми. Будь-які непередбачені ситуації мають бути задокументовані у звіті, який при необхідності надсилається розробнику для визначення причини збою в роботі та усуненні помилок, які могли привести до нестабільної роботи програмного продукту.

4.2 Вимоги до складу і параметрів технічних засобів

Програмний продукт є веб-застосуванням, що складається з клієнта та сервера.

Для коректної роботи серверної частини веб застосування необхідно використовувати веб-сервер, що може бути представлений локальною обчислювальною машиною. Сервер мінімально повинен мати такі характеристики:

- процесор з тактовою частотою не нижче 2 ГГц;
- об’єм оперативної пам’яті (не менше 2 ГБ);
- жорсткий диск (не менше 40 ГБ);
- операційна система Windows 7 і вище;
- Python 3.5.
- Django 3.0

Для коректної роботи клієнтської частини веб застосування необхідно:

- підключення до мережі Інтернет.

5 СТАДІЇ І ЕТАПИ РОЗРОБКИ

5.1 Основні етапи виконання робіт з розробки системи ведення наукової роботи.

№ п/п	Назва етапу роботи	Термін виконання етапу	Результат виконання
1	Підготовка технічного завдання на розробку програмного продукту	15.02.2020	
2	Створення архітектури системи	17.02.2020	
3	Технічне проектування – функціональність, модулі, задачі, цілі тощо	21.02.2020	
4	Узгодження з керівником інтерфейсу користувача	07.03.2020	
5	Розробка програми	14.04.2020	
6	Налагодження програми	22.04.2020	
7	Тестування програми	16.05.2020	
8	Здача готового програмного продукту замовнику	27.05.2020	

6 ПОРЯДОК КОНТРОЛЮ ТА ПРИЙМАННЯ СИСТЕМИ

6.1 Види випробувань

Види, склад, об’єм і методи випробувань програмного продукту повинні бути викладені в програмі і методиці випробувань системи, що розробляється в складі робочої документації.

UNICHECK

Власник документу:
Попенко Володимир Дмитрович

ID перевірки:
1003979406

Дата перевірки:
12.06.2020 03:24:23 EEST

Тип перевірки:
Doc vs Internet + Library

Дата звіту:
12.06.2020 14:03:01 EEST

ID користувача:
77149

Назва документу: Todorov_is61

ID файлу: 1003994457 Кількість сторінок: 80 Кількість слів: 11718 Кількість символів: 95163 Розмір файлу: 2.12 MB

19.4% Схожість

Найбільша схожість: 7.76% з джерело https://ela.kpi.ua/bitstream/123456789/27652/1/Zamekula_magistr.pdf

13% Схожість з Інтернет джерелами 108 ..... Page 82

17.9% Текстові збіги по Бібліотеці акаунту 36 ..... Page 84

1.21% Цитат

Цитати 1 ..... Page 85

Вилучення переліку посилань вимкнено

0% Вилучень

Вилучений текст відсутній

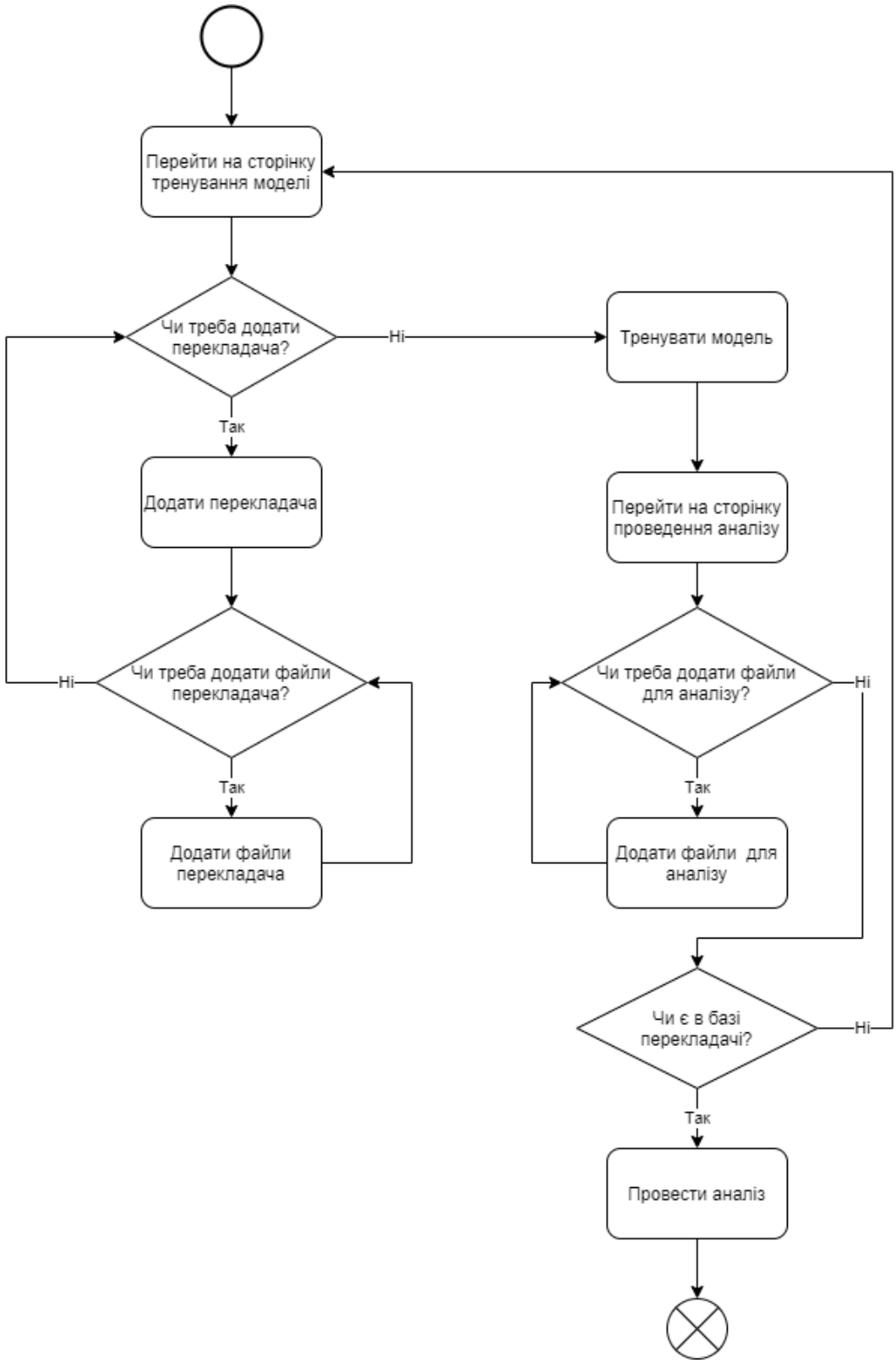
Підміна символів

Заміна символів 1

Графічний матеріал
до дипломного проєкту

на тему: Інформаційно-аналітична система визначення лінгвістичних параметрів перекладача

Київ – 2020 року



					ДП 6126.02.000 ССД		
Зм.	Арк.	№ документа	Підпис	Дата	Схема структурна процесу діяльності	Літера	Маса / Масштаб
Розробив		Тодоров Д.В.					
						Аркуш 1	Аркушів 1
Перевірив		Фіногенов О.Д.					
Т. кон.					Інформаційно-аналітична система визначення лінгвістичних параметрів перекладача	КПІ ім. Ігоря Сікорського кафедра АСОІУ гр. ІС-61	
Н. кон.		Тєлишева Т.О.					
Затвердив		Фіногенов О.Д.					

					ДП 6126.03.000 ССП			
					Схема структурна послідовності	Літера	Маса	Масштаб
Зм.	Арк.	№ документа	Підпис	Дата				
Розробив		Тодоров Д.В.				Аркуш 1		Аркушів 1
Перевірив		Фіногенов О.Д.						
Т. кон.					*Інформаційно-аналітична система визначення лінгвістичних параметрів перекладача*	*КПІ ім. Ігоря Сікорського кафедра АСОІУ гр. ІС-61*		
Н. кон.		Тєлишева Т.О.						
Затвердив		Фіногенов О.Д.						

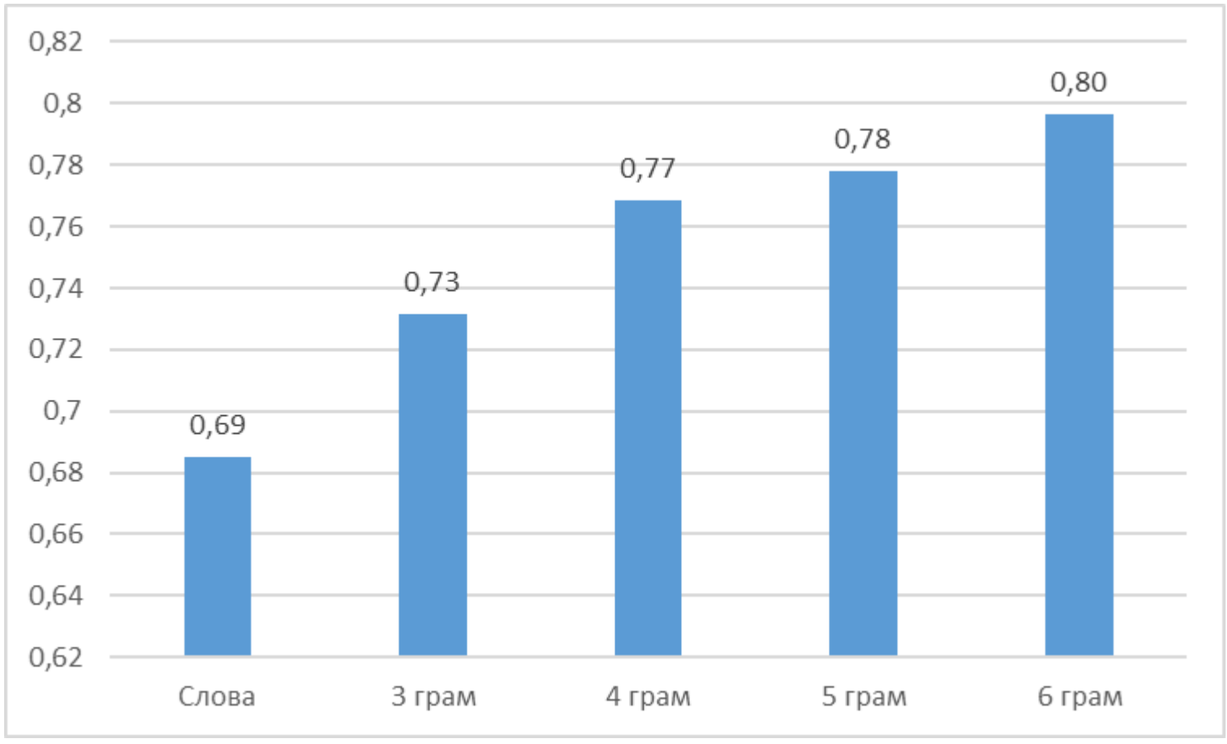
					ДП 6126.04.000 СБД		
Зм.	Арк.	№ документа	Підпис	Дата	Схема бази даних	Літера	Маса / Масштаб
Розробив		Тодоров Д.В.					
Перевірив		Фіногенов О.Д.				Аркуш 1	Аркушів 1
Т. кон.					Інформаційно-аналітична система визначення лінгвістичних параметрів перекладача	КПІ ім. Ігоря Сікорського кафедра АСОІУ гр. ІС-61	
Н. кон.		Тєлишева Т.О.					
Затвердив		Фіногенов О.Д.					

					ДП 6126.05.000 ССК	
Зм.	Арк.	№ документа	Підпис	Дата	Схема структурна класів програмного забезпечення	Літера / Маса / Масштаб
Розробив		Тодоров Д.В.				
Перевірив		Фіногенов О.Д.				Аркуш 1 / Аркушів 1
Т. кон.					Інформаційно-аналітична система визначення лінгвістичних параметрів перекладача	КПІ ім. Ігоря Сікорського кафедра АСОІУ гр. ІС-61
Н. кон.		Тєлишева Т.О.				
Затвердив		Фіногенов О.Д.				

ДП 6126.06.000 ССК

					ДП 6126.06.000 ССК		
Зм.	Арк.	№ документа	Підпис	Дата	Схема структурна компонентів	Літера	Маса / Масштаб
Розробив		Тодоров Д.В.					
Перевірив		Фіногенов О.Д.				Аркуш 1	Аркушів 1
Т. кон.					Інформаційно-аналітична система визначення лінгвістичних параметрів перекладача	КПІ ім. Ігоря Сікорського кафедра АСОІУ гр. ІС-61	
Н. кон.		Тєлишева Т.О.					
Затвердив		Фіногенов О.Д.					

Рішення з математичного забезпечення



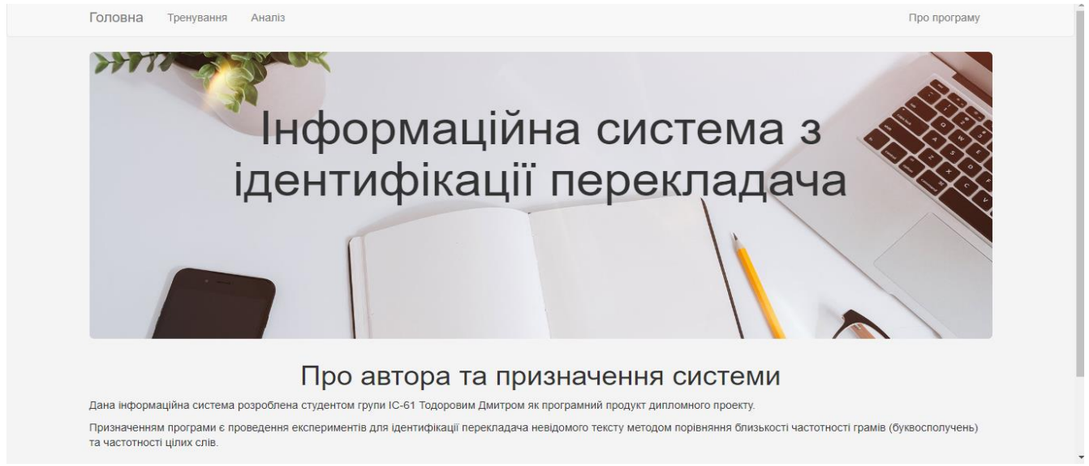
Демонстраційний плакат до дипломного проекту

„Інформаційно-аналітична система визначення лінгвістичних параметрів перекладача ”

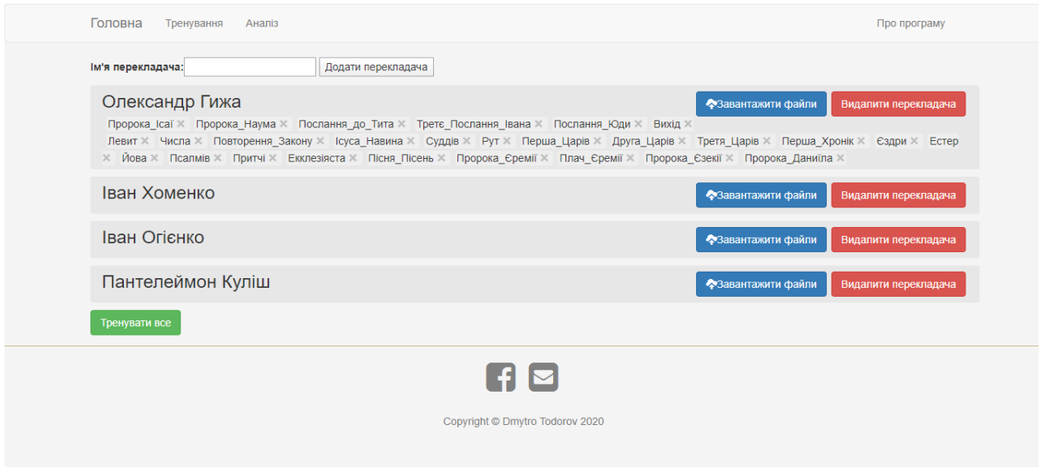
Виконав студент гр. ІС-61 Тодоров Д.В.

Керівник ДП Фіногенов О.Д.

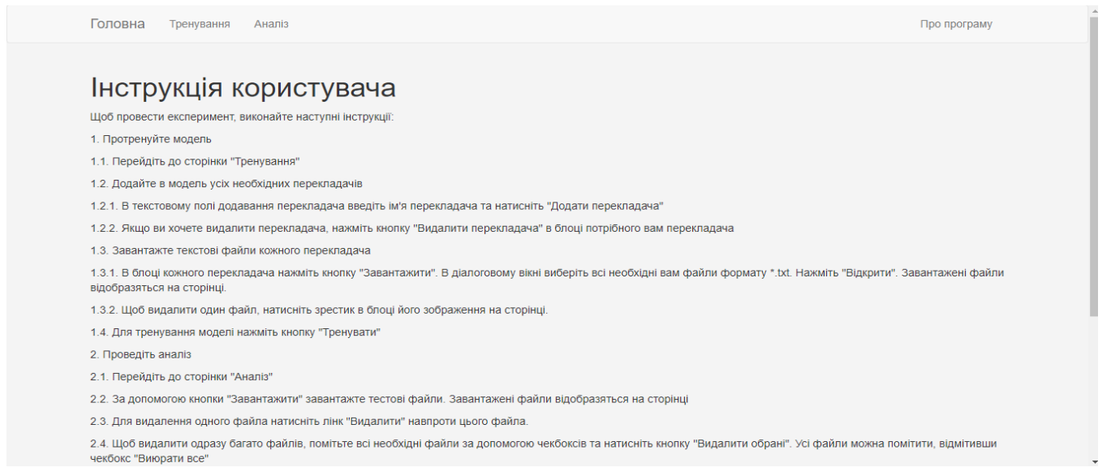
Екранна форма головної сторінки веб-застосунку



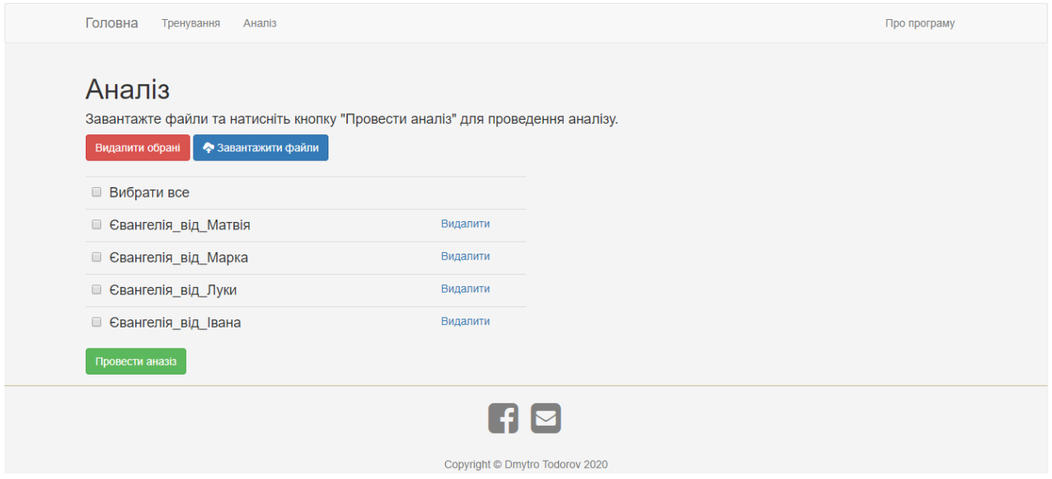
Екранна форма сторінки з тренуванням моделі



Екранна форма сторінки з інструкцією користувача

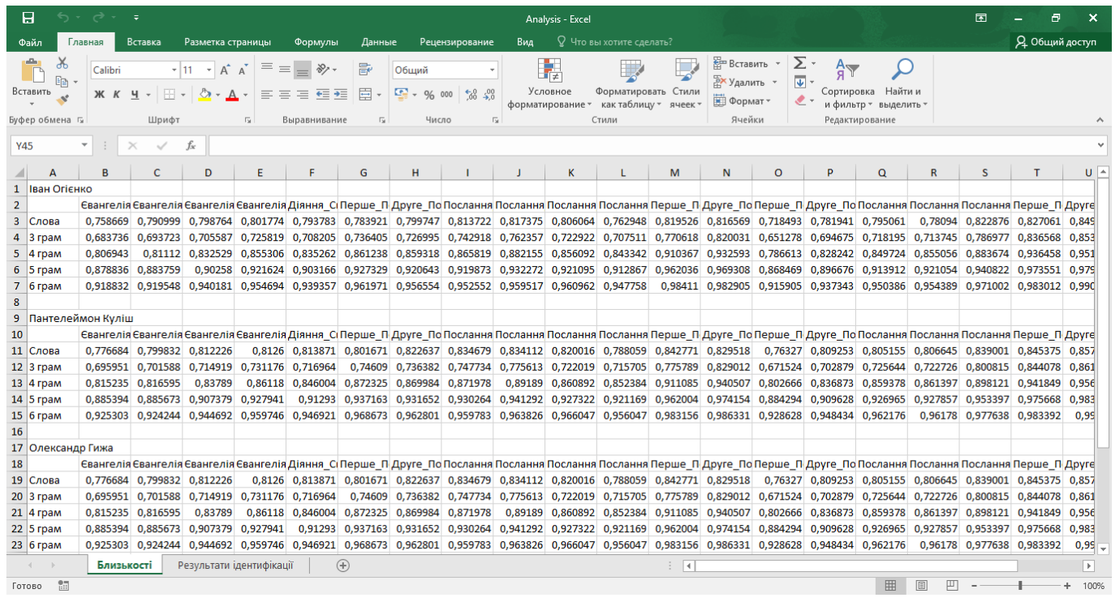


Екранна форма сторінки проведення аналізу



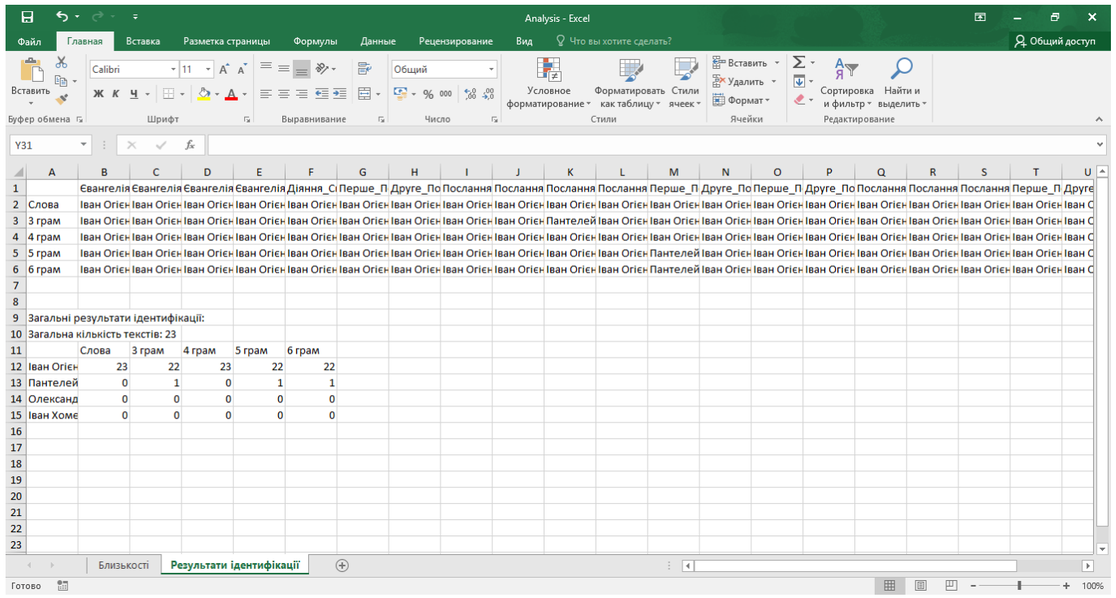
					ДП 6126.07.000 КЕ	
Зм.	Арк.	№ документа	Підпис	Дата	*Креслення вигляду екранних форм*	Літера / Маса / Масштаб
Розробив		Тодоров Д.В.				
Перевірив		Фіногенов О.Д				Аркуш 1 / Аркушів 2
Т. кон.						
Н. кон.		Тєлишева Т.О.			*Інформаційно-аналітична система визначення лінгвістичних параметрів перекладача*	*КПІ ім. Ігоря Сікорського кафедра АСОІУ гр. ІС-61*
Затвердив		Фіногенов О.Д.				

Екранна форма таблиці «Близькості» Excel-звіту



	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U
1	Іван Огієнко																				
2		Євангелія	Євангелія	Євангелія	Євангелія	Діяння_Сі	Перше_П	Друге_По	Послання	Послання	Послання	Послання	Перше_П	Друге_По	Перше_П	Друге_По	Послання	Послання	Послання	Перше_П	Друге
3	Слова	0,758669	0,790999	0,798764	0,801774	0,793783	0,783921	0,799747	0,813722	0,817375	0,806064	0,762948	0,819526	0,816569	0,718493	0,781941	0,795061	0,78094	0,822876	0,827061	0,845
4	3 грам	0,683736	0,693723	0,705587	0,725819	0,708205	0,736405	0,726995	0,742918	0,762357	0,722922	0,707511	0,770618	0,820031	0,651278	0,694675	0,718195	0,713745	0,786977	0,836568	0,853
5	4 грам	0,806943	0,81112	0,832529	0,855306	0,835262	0,861238	0,859318	0,865819	0,882155	0,856092	0,843342	0,910367	0,932593	0,786613	0,828242	0,849724	0,855056	0,883674	0,936458	0,951
6	5 грам	0,878836	0,883759	0,90258	0,921624	0,903166	0,927329	0,920643	0,919873	0,932272	0,921095	0,912867	0,962036	0,969308	0,868469	0,896676	0,913912	0,921054	0,940822	0,973551	0,975
7	6 грам	0,918832	0,919548	0,940181	0,954694	0,939357	0,961971	0,956554	0,952552	0,959517	0,960962	0,947758	0,98411	0,982905	0,915905	0,937343	0,950386	0,954389	0,971002	0,983012	0,990
8																					
9	Пантелеймон Куліш																				
10		Євангелія	Євангелія	Євангелія	Євангелія	Діяння_Сі	Перше_П	Друге_По	Послання	Послання	Послання	Послання	Перше_П	Друге_По	Перше_П	Друге_По	Послання	Послання	Послання	Перше_П	Друге
11	Слова	0,776684	0,799832	0,812226	0,8126	0,813871	0,801671	0,822637	0,834679	0,834112	0,820016	0,788059	0,842771	0,829518	0,76327	0,809253	0,805155	0,806645	0,839001	0,845375	0,857
12	3 грам	0,695951	0,701588	0,714919	0,731176	0,716964	0,74609	0,736382	0,747734	0,775613	0,722019	0,715705	0,775789	0,829012	0,671524	0,702879	0,725644	0,722726	0,800815	0,844078	0,861
13	4 грам	0,815235	0,816595	0,83789	0,86118	0,846004	0,872325	0,869984	0,871978	0,89189	0,860892	0,852384	0,911085	0,940507	0,802666	0,836873	0,859378	0,861397	0,898121	0,941849	0,956
14	5 грам	0,885394	0,885673	0,907379	0,927941	0,91293	0,937163	0,931652	0,930264	0,941292	0,927322	0,921169	0,962004	0,974154	0,884294	0,909628	0,926965	0,927857	0,953397	0,975668	0,983
15	6 грам	0,925303	0,924244	0,944692	0,959746	0,946921	0,968673	0,962801	0,959783	0,963826	0,966047	0,956047	0,983156	0,986331	0,928628	0,948434	0,962176	0,96178	0,977638	0,983392	0,99
16																					
17	Олександр Гижа																				
18		Євангелія	Євангелія	Євангелія	Євангелія	Діяння_Сі	Перше_П	Друге_По	Послання	Послання	Послання	Послання	Перше_П	Друге_По	Перше_П	Друге_По	Послання	Послання	Послання	Перше_П	Друге
19	Слова	0,776684	0,799832	0,812226	0,8126	0,813871	0,801671	0,822637	0,834679	0,834112	0,820016	0,788059	0,842771	0,829518	0,76327	0,809253	0,805155	0,806645	0,839001	0,845375	0,857
20	3 грам	0,695951	0,701588	0,714919	0,731176	0,716964	0,74609	0,736382	0,747734	0,775613	0,722019	0,715705	0,775789	0,829012	0,671524	0,702879	0,725644	0,722726	0,800815	0,844078	0,861
21	4 грам	0,815235	0,816595	0,83789	0,86118	0,846004	0,872325	0,869984	0,871978	0,89189	0,860892	0,852384	0,911085	0,940507	0,802666	0,836873	0,859378	0,861397	0,898121	0,941849	0,956
22	5 грам	0,885394	0,885673	0,907379	0,927941	0,91293	0,937163	0,931652	0,930264	0,941292	0,927322	0,921169	0,962004	0,974154	0,884294	0,909628	0,926965	0,927857	0,953397	0,975668	0,983
23	6 грам	0,925303	0,924244	0,944692	0,959746	0,946921	0,968673	0,962801	0,959783	0,963826	0,966047	0,956047	0,983156	0,986331	0,928628	0,948434	0,962176	0,96178	0,977638	0,983392	0,99

Екранна форма таблиці «Результати аналізу» Excel-звіту



	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U
1		Євангелія	Євангелія	Євангелія	Євангелія	Діяння_Сі	Перше_П	Друге_По	Послання	Послання	Послання	Послання	Перше_П	Друге_По	Перше_П	Друге_По	Послання	Послання	Послання	Перше_П	Друге
2	Слова	Іван Огіє	Іван Огіє	Іван Огіє	Іван Огіє	Іван Огіє	Іван Огіє	Іван Огіє	Іван Огіє	Іван Огіє	Іван Огіє	Іван Огіє	Іван Огіє	Іван Огіє	Іван Огіє	Іван Огіє	Іван Огіє	Іван Огіє	Іван Огіє	Іван Огіє	Іван С
3	3 грам	Іван Огіє	Іван Огіє	Іван Огіє	Іван Огіє	Іван Огіє	Іван Огіє	Іван Огіє	Іван Огіє	Іван Огіє	Іван Огіє	Іван Огіє	Іван Огіє	Пантелей	Іван Огіє	Іван Огіє	Іван Огіє	Іван Огіє	Іван Огіє	Іван Огіє	Іван С
4	4 грам	Іван Огіє	Іван Огіє	Іван Огіє	Іван Огіє	Іван Огіє	Іван Огіє	Іван Огіє	Іван Огіє	Іван Огіє	Іван Огіє	Іван Огіє	Іван Огіє	Іван Огіє	Іван Огіє	Іван Огіє	Іван Огіє	Іван Огіє	Іван Огіє	Іван Огіє	Іван С
5	5 грам	Іван Огіє	Іван Огіє	Іван Огіє	Іван Огіє	Іван Огіє	Іван Огіє	Іван Огіє	Іван Огіє	Іван Огіє	Іван Огіє	Іван Огіє	Іван Огіє	Іван Огіє	Пантелей	Іван Огіє	Іван Огіє	Іван Огіє	Іван Огіє	Іван Огіє	Іван С
6	6 грам	Іван Огіє	Іван Огіє	Іван Огіє	Іван Огіє	Іван Огіє	Іван Огіє	Іван Огіє	Іван Огіє	Іван Огіє	Іван Огіє	Іван Огіє	Іван Огіє	Іван Огіє	Пантелей	Іван Огіє	Іван Огіє	Іван Огіє	Іван Огіє	Іван Огіє	Іван С
7																					
8																					
9	Загальні результати ідентифікації:																				
10	Загальна кількість текстів: 23																				
11		Слова	3 грам	4 грам	5 грам	6 грам															
12	Іван Огіє	23	22	23	22	22															
13	Пантелей	0	1	0	1	1															
14	Олександ	0	0	0	0	0															
15	Іван Хоме	0	0	0	0	0															
16																					
17																					
18																					
19																					
20																					
21																					
22																					
23																					

						ДП 6126.07.000 КЕ	
Зм.	Арк.	№ документа	Підпис	Дата	Креслення вигляду екранних форм	Літера	Маса / Масштаб
Розробив		Тодоров Д.В.					
Перевірив		Фіногенов О.Д				Аркуш 2	Аркушів 2
Т. кон.							
Н. кон.		Тєлишева Т.О.			Інформаційно-аналітична система визначення лінгвістичних параметрів перекладача	КПІ ім. Ігоря Сікорського кафедра АСОІУ гр. ІС-61	
Затвердив		Фіногенов О.Д.					