Міністерство освіти і науки України
Університет митної справи та фінансів

Факультет інноваційних технологій
Кафедра комп’ютерних наук та інженерії програмного забезпечення

Дипломна робота бакалавра

на тему «Комп'ютерна технологія ідентифікації авторства тексту»

Виконав: студент групи К19-2
Спеціальність 122 «Комп’ютерні науки»
Прядка Олексій Іванович
Керівник к.ф.м.н., доц. Фірсов О. Д.
Рецензент
_____
_____

Дніпро – 2023

АНОТАЦІЯ

Прядка О. І. Комп'ютерна технологія ідентифікації авторства тексту.

Кваліфікаційна робота на здобуття освітнього ступеня бакалавр за спеціальністю 122 «Комп’ютерні науки». – Університет митної справи та фінансів, Дніпро, 2023.

У даній роботі досліджувався процес ідентифікації авторства тексту за допомогою комп'ютерних технологій. Під час дослідження якої булли розглянуті методи та засоби для вирішення цієї задачі. Вибір було зроблено на користь використання методів машинного навчання. Було розглянуто повний процес вирішення задачі ідентифікації авторства за допомогою методів машинного навчання, від підготовки текстів до оцінки отриманих результатів, досліджувалась ефективність різних типів моделей, та засобів приставляння даних. У підсумку даного дослідження вдалося досягти 90-95% точності при ідентифікації авторства тексту що являєтеся доволі хорошим результатом.

Як підсумок етапу досліджень було розроблено веб-додаток для ідентифікації авторства тексту. Програмний додаток було насисано з використанням новітніх технологій він має зручний користувацький інтерфейс і просунуту систему відслідковування помилок та може розпізнати як текст написаний одним автором так і той що написали у співавторстві.

Ключові слова: нормалізація, тонізація, лематизація, стоп слово, машинне навчання, класична модель, глибока модель , LSTM , GRU, веб-додаток.

ЗМІСТ

ВСТУП

Ще з давніх часів аж по сьогоднішній день тексти були залишаються основним способом для отримання , зберігання або розповсюдження інформації, тому потреба в ідентифікації їх авторства існувала завжди та в самих різних сферах нашого повсякденного життя. Знання про авторство тексту надає багато переваг та може багато на що вплинути , до прикладу знання про авторство тексу може вплинути на його загальне сприйняття ставлячи під сумнів достовірність інформації в цьому тексті або допомогти визначити чи має даний текст якусь художню історичну або наукову цінність.

Після появи та стрімкого розвитку мережі інтернет інформаційний простір навколо нас почав дуже швидко розширюватися за рахунок того що багато людей отримали доступ до засобів які дозволяють споживати , створювати та розповсюджувати майже нескінчену кількість текстової інформації майже кожної хвилини тому загальні об’єми цієї інформації постійно збільшуються і відбувається цей процес просто неймовірною швидкістю. Невважаючи на такі стрімкі темпи збільшення кількості інформації її якість залишає бажати кращого, станом на сьогодні більша частина текстової інформації яка нас оточує є не структурованою, підробленою або недостовірною тому потреба в ідентифікації авторства текстової інформації стала актуальною як ніколи раніше.

Для вирішення задачі ідентифікації авторства текстів було розроблено багато різноманітних способів. Але щоб вирішити цю задачу в умовах такої великої кількості інформації з допомогою існуючих методів ідентифікації буде потрібно докласти для цього чималих зусиль через те що обробляти таку велику кількість інформації вручну стало дуже складно. Тому все частіше виникає потреба в певному рівні автоматизації даного процесу. Для отримання необхідного рівня автоматизації процесу ідентифікації авторства доцільно буде

звернутися до сфери комп'ютерних технологій. Вона наддає багато різноманітних методів та інструментів, які можуть бути корисними для вирішення такого роду задач. Основними перевагами використання комп'ютерних технологій для ідентифікації авторства тексту є:

- Швидкість: комп'ютерні алгоритми можуть аналізувати великі обсяги тексту в надзвичайно короткий проміжок часу.
- Об'єктивність: комп'ютерні алгоритми використовують тільки перевірені алгоритми для аналізу особливостей тексту та не піддаються емоційному впливу чи упередженим думкам під час процесу аналізу.

Об’єктом досліджень в роботі виступає комп’ютерна технологія ідентифікації авторства тексту.

Метою цієї дипломної роботи є дослідження можливостей та доцільності використання комп’ютерних технологій для вирішення задачі ідентифікації авторства тексту. У відповідності до мети в роботі ставились та вирішувались наступні завдання дослідження :

- Ознайомитися з предметною областю.
- Ознайомитися з дослідженнями в предметній області.
- Дослідити наявні методи для ідентифікації авторства тексту.
- Дослідити процес ідентифікації авторства тексту методами машинного навчання.
- Дослідити ефективність використання класичних моделей машинного навчання для вирішення задачі ідентифікації авторства тексту.
- Розробити моделі глибокого машинного навчання для вирішення задачі ідентифікації авторства тексту.
- Розробити додаток для ідентифікації авторства тексту на основі проведених досліджень.
- Протестувати додаток для ідентифікації авторства тексту.

Структура роботи:

- Розділ 1. «Методи ідентифікації авторства тексту» В даному розділі робиться огляд задачі ідентифікації авторства , піднімаються питання актуальності даного процесу, наявних при цьому проблем та способів вирішення даної задачі.
- Розділ 2. «Комп'ютерна технологія ідентифікації авторства» В цьому розділі було розглянуто повний алгоритм вирішення задачі в сфері обробки природної мови та технології що для цього використовуються.
- Розділ 3. Програмний додаток LAV В даному розділі було розглянуто процес розробки додатку LAV , було показано алгоритм його роботи, надано інструкцію користування додатком та проведений процес його тестування.

  Розділ 4. «Обчислювальний експеримент» В даному розділі було протестовано чотири популярних моделі класичного навчання для багато класової класифікації а також проведено роботу з моделями глибокого навчання шляхом створення і перевірки двох моделей типу LSTM та GRU.

РОЗДІЛ 1. МЕТОДИ ІДЕТИФІКАЦІЇ АВТОРСТВА ТЕКСТУ

1.1. Вступ

Ідентифікація авторства тексту – це процес спроби ідентифікації ймовірного авторства документа на основі документів, авторство яких відомо [1]. Вона може бути корисною в різних сферах нашого життя наприклад в сфері журналістики у правоохоронних органах або літературознавстві. В сфері журналістики знання авторства може бути корисним для перевірки достовірності опублікованої або отриманої інформації. В свою чергу в сфері правоохоронних органів встановлення авторства потрібного тексту може стати важливою доказовою базою для розкриття кримінальної справи. А в сфері літературознавства знання авторства тексту меже допомогти прослідкувати за творчою еволюцією письменника або навіть виявити нові стилі та жанри.

Процес ідентифікації авторства базується на пошуку та подальшому аналізу стилістичних особливостей письма та мовлення автора для того щоб виявити особливості які будуть притаманними лише йому це може бути що завгодно наприклад використання маловживаної лексики або особливості розстановки розділових знаків у тексті головне щоб це була унікальна сукупність факторів яка б дозволила ідентифікувати саме потрібного автора. Ідентифікація авторства тексту є дуже складною та об’ємною за часом задачею яка потребує до себе максимальної уваги тому що в процесі ідентифікації завжди потрібно приймати до уваги дуже велику кількість різноманітних факторів та інформації яка відноситься до кожного конкретного автора через що в процесі ідентифікації можуть виникати різні проблеми та помилки пов’язані з наявністю людського фактору що у підсумку може призвести до погіршення отриманих результатів

.

1.2. Огляд досліджень в предметній області

Ідентифікація авторства текстів є дуже корисною але водночас доволі складною задачею. Це спонукає як шукати нові так і вдосконалювати старі способи для вирішення цієї задачі заради полегшення її процесу, що призводить до появи безлічі досліджень в рамках даної теми. Наприклад стаття авторів Yülüce, İ та Dalkılıç, F. під назвою «Author Identification with Machine Learning Algorithms. International Journal of Multidisciplinary Studies and Innovative» [2] в ній досліджувалась різні методи класичного машинного навчання за допомогою яких було проведено експеримент з ідентифікації автора тексту. В даному експерименті були використані опорні векторні машини (SVM), наївний метод Байєса, багатошаровий перцептрон (MLP), модель логістичної регресії (LR), стохастичний градієнтний спуск (SGD) та методи ансамблевого навчання, включаючи надзвичайно випадкові дерева (Extra Trees). Кожен з даних методів був застосований на трьох наборах даних різного розміру які були сформовані з газетних статей і включали в себе 10, 15 та 20 авторів. Вектори даних формувались з допомогою статистичного показника (TF-IDF) з використанням 1-грамових та 2-грамових токенів слів. По результатам даного експерименту можна з упевненістю сказати що використання методів машинного навчання для ідентифікації авторства тексту є доволі ефективним рішенням бо під час експерименту вдалося добитись точності розпізнання автора в районі 93-97% що є доволі хорошим результатом.

Також з цікавих публікацій можна виділити роботу М. І. Лупей «Визначення авторської належності українськомовного тексту за допомогою системи для ідентифікації авторства»[3]. У даній роботі автор описує систему для ідентифікації авторської належності тексту, основними елементами якої є блоки стемінгу, та вбудовування, класифікації та візуалізації результатів. При

дослідженні роботи системи на даних творів українських письменників було отримано результат класифікації на рівні 98% при попарному порівнянні двох авторів.

Продивившись публікації наведені вище може скластися хибне враження про те що всі дослідження в даній предметній області проводяться лише з використанням класичних методів машинного навчання але це не так ось до прикладу робота Б. О. Подшиваленко під назвою «Застосування методів статистичного аналізу для розв'язання задачі ідентифікації текстів»[4] У ній досліджуються математичні методи ідентифікації автора тексту. За допомогою методу аналізу ієрархій обирається оптимальний метод для вирішення проблеми за певними критеріями метод статистичного аналізу для ідентифікації тексту української літератури.

Є цікава стаття від авторів Shriya TP Gupta, Jajati Keshari Sahoo, Rajendra та Kumar Roul під назвою «Authorship Identification using Recurrent Neural Networks»[5]. В цій статті розказують про використання підходу глибокого навчання для вирішиггя задачі ідентифікації авторства. В досліджені було розглянуто роботу з рекурентними мережами LSTM та GRU в даній статті моделі навчались на корпусі даних C50 та BBC з застосуванням різних алгоритмів для вбудовування даних. У підсумку виявилось що хоча GRU завчай має меншу точність в порівняні з LSTM але якщо мова йде про відносно невеликі набори даних то доцільніше використовувати саме GRU тому що на таких наборах вона зазвичай перевершує LSTM і при цьому працює набагато швидше

1.3 Основні підходи для ідентифікації авторства

Основними підходами для вирішення задачі ідентифікації авторства тексту можна вважати : використання методів статистичного аналізу, залучення експертів та використання методів машинного навчання. Розглянемо докладніше що передбачає використання кожного з даних підходів паралельно висвітливши їх сильні та слабкі сторони.

Використання статистичних методів – це підхід який використання різних статистичних методів для аналізу та порівняння різних характеристик текстів наприклад, можна порівняти частоту зустрічі слів, фраз та речень. Статистичні методи можна поділити на одномірні та багатовимірні Одномірні методи – це методи аналізу даних які застосовуються у випадках, якщо існує єдиний вимірювач для оцінки кожного елемента вибірки або їх вимірників декілька, але кожна змінна аналізується окремо від усіх інших[6]. Багатовимірні методи застосовуються якщо для оцінки даних кожного елемента вибірки кили використовуються два або більше вимірників, а змінні аналізуються одночасно[6].

Переваги використання статистичних методів:

- Можна працювати з невеликою кількістю текстів.
- Не треба мати глибоких знань з лінгвістики або інших суміжних наук.

Недоліки використання статистичних методів:

- Є вразливим до зміни авторського стилю
- Не враховує контексту та семантики тексту
- Не може відрізнити авторів зі схожим стилем письма

Експертний аналіз тексту – такий підхід передбачає залучення лінгвістів або інших експертів з суміжних галузей, котрі займаються аналізом текстів та

мають достатній багаж досвіду для визначення стилістичних та інших характеристик тексту.

Перевагами експертного аналізу є:

- Висока точність визначення авторства
- Можливість виявлення унікальних характеристик авторського стилю, які не можна виявити іншими методами.

Недоліки експертного аналізу тексту:

- Оцінки експерта є суб'єктивними.
- Потрібно витрачати час на пошуки експертів.
- Висока ціна за послуги експерта

Використання методів машинного навчання – цей підхід полягає у використанні різних комп'ютерних алгоритмів та методів машинного навчання для розпізнавання стилю та характеристик тексту, що можуть свідчити про його автора.

Підхід має такі переваги:

- Висока точність визначення авторства, особливо при використанні великої кількості текстових даних для навчання.
- Можливість використання для автоматизованого аналізу великих обсягів тексту.
- Можливість виявлення унікальних характеристик авторського стилю, які не можуть бути виявлені статистичними методами.

Недоліки використання методів машинного навчання:

- Потреба у великій кількості текстових даних для навчання.
- Обмеженість в розпізнаванні схожих авторів або співавторства.

1.4. Існуючі програмні засоби для ідентифікації авторства

У цьому підрозділі наведено огляд трьох найпопулярніших існуючих програмних засобів для ідентифікації авторства тексту.

Plagiarisma – це програмна платформа для перевірки тексту на запозичення її основною перевагою підтримка понад 190 мов. Вона дозволяє перевіряти тексти та виявляти збіги текстами які індексуються пошуковими системами Google та Bing. Ця програма є повністю безкоштовною та має як онлайн версію так і версію що може бути завантажена та встановлена на комп'ютер.

StrikePlagiarism.com – це інтернет платформа для виявлення плагіату яка може перевіряти оригінальність наданого їй тексту в автоматичному режимі. Дана платформа має три основних переваги. Перша перевага це наявність простого та зручного інтерфейсу користувача. Друга перевага це відсутність обмежень на обсяг тексту. Третя перевага даної платформи це підтримка та можливість роботи з багатьма форматами текстових документів таких як DOC,DOCX, ODT, TXT та PDF.

Unichek – онлайн-інструмент для швидкої перевірки плагіату текстів, створений українською командою. Сервіс здійснює швидку перевірку як в Інтернеті, так і в базах наукових праць ВНЗ. Використовуйте формати DOC, DOCX, PDF, ODT, RTF, HTML з необмеженою кількістю одночасних користувачів. Перевірте плагіат за лічені секунди та надайте найточніші результати в реальному часі.

1.5. Висновки до розділу 1. Методи ідентифікації авторства тексту

В підрозділі 1.1 «Огляд предметної області»: Було дано визначення поняття ідентифікації авторства тексту ,розглянуто можливі сфери застосування даного процесу, теперішня доцільність його використання та наявні проблеми.

В підрозділі 1.2 «Огляд досліджень в предметній області» будо розглянуто та проаналізовано деякі цікаві дослідження в сфері ідентифікації авторства це робилось з ціллю підкріплення слів про актуальність даного процесу які було сказано ще минулого розділу та більшого поглиблення в предметну область

В підрозділі 1.3 «Основні підходи для ідентифікації авторства» було розглянуто основні підходи для ідентифікації авторства тексту а саме:

- З використанням статистичних методів.
- З використанням експертного аналізу.
- З використанням машинного навчання.

Та виявлено їхні сильні та слабкі сторони

В підрозділі 1.4 «Існуючі програмні засоби» в даному розділі розбирались переваги використання готових програмних рішень для ідентифікації авторства тексту.

РОЗДІЛ 2. КОМП'ЮТЕРНА ТЕХНОЛОГІЯ ІДЕНТИФІКАЦІЇ АВТОРСТВА

Беручи до уваги інформацію з попереднього розділу можна сказати що для вирішення задачі ідентифікації авторства тексту існує декілька фундаментально різних між собою підходів кожен з яких має свої переваги і недоліки та потребує специфічних навичок для свого застосування. Не буває нічого що можна використати в усіх випадках життя при виборі підходу для вирішення цієї задачі в першу чергу слід розібратись в наступних питаннях:

- Який час був відведений на виконання поставленої задачі?
- Наскільки великий об’єм роботи потрібно зробити?
- Який бюджет було виділено?
- Наскільки часто потрібно виконувати задачі такого типу?
- Давши відповіді на ці питання стане легше обрати підхід який буде доцільний саме в даній момент часу.

Обираючи найкращий підхід для вирішення задачі кваліфікаційної роботи між використанням методів статистичного аналізу, експертної оцінки та машинного навчання вибір був зроблений на користь останніх тому що дивлячись на вирішення задачі ідентифікації авторства саме в контексті комп'ютерних технологій то використання машинного навчання дасть найбільшу кількість переваг, першим аргументом на його користь є те що методи машинного навчання можуть надати необхідний рівень автоматизації процесу роботи з текстами що дозволить працювати з об’ємами даних які можуть бути набагато більші за ті що можна адекватно обробити в ручному режимі , другим аргументом є те що така автоматизована тексту обробка буде проходити набагато швидше та матиме набагато кращий кінцевій кінцевий результат ніж робота в ручному режимі яка займатиме такий самий проміжок часу.

2.1. Ідентифікація авторства методами машинного навчання

Перш ніж говорити про сам процес ідентифікації авторства потрібно розібратись що таке машинне навчання – це галузь штучного інтелекту та інформатики, яка зосереджена на використанні даних та алгоритмів для імітації способу людського навчання[7]. Ця галузь зосереджена на вирішені достатньо великої кількісті різноманітних задач, таких як класифікація генерація, або аналіз даних. Вона розділена на різні сфери досліджень кожна з яких має свій набір технік та алгоритмів для вирішення типу задач певного типу. Дивлячись на задачу ідентифікації авторства в контексті галузі машинного навчання її можна віднести до задач класифікації тексту вирішним яких займаються в сфері обробки природньої мови (NLP) – це сфера машинного навчання, орієнтована на людські мови. Сюди входять як писемна, так і розмовна мови[8]. Вирішення будь якої задачі з обробки природної мови (в нашому випадку це задача класифікації) можна поділити на такі основні етапи:

- Етап підготовки текстових даних
- Етап вбудовування даних
- Етап підготовки та навчання моделей
- Етап оцінки результатів

Дані етапи повинні виконуватися один за одним в такому самому порядку як вони представлені у списку вище. Далі в розділі буде надано загальну інформацію про те що відбувається на кожному з цих етапів та розібрано основну термінологію яка використовується в сфері обробки природної мови.

Етап підготовки текстових даних – це комплексний процес підготовки текстових даних для того щоб мати можливість їх подальшого використання в методах машинного навчання. Цей етап включає в себе такі процеси як збір, нормалізація тексту, вилучення стоп-слів та лематизацію. Кінцевим результатом

виконання даного етапу є створення корпусу. В сфері NLP корпус – це сукупність текстів яка може являти собою збір оглядів фільмів, коментарів або авторських творів [8] .

Перше що потрібно зробити при створені корпусу це провести нормалізацію тексту – це процес який потрібен для очищення тексту від зайвої інформації зазвичай це знаки пунктуації та різний регістр символів. Кінцевим результатом виконання процесу нормалізації зазвичай виступає текст який переведений до нижнього регістру не має розділових знаків.

Друге що потрібно зробити при створені корпусу це вилучити з нормалізованого тексту усі стоп-слова – це слова, які зустрічаються у великому обсязі текстів і не мають особливого смислового навантаження.[4]

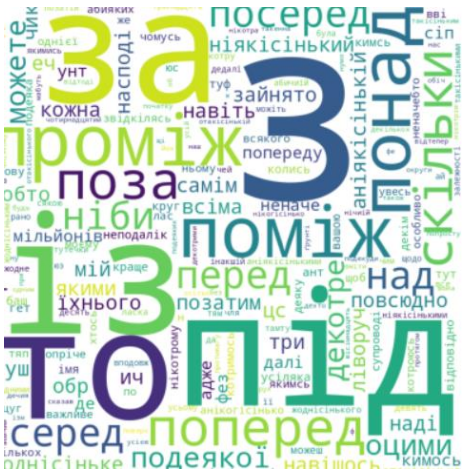


Рисунок 2.1 – Приклад деяких українських стоп-слів.

Вилучення стоп-слів допомагає полегшити подальший аналіз корпусу в моделях машинного навчання підвищує точність отриманих результатів.

Третє що потрібно зробити при створені корпусу це провести лематизацію тексту. Лематизація – це метод нормалізації, тобто групування змінених термінів до їхньої базової форми, обумовленої відповідною частиною мови в наданому

тексті[8]. Наприклад такі слова «пішла , пішов» буде замінено на слово «піти» або слова «знав, знає» на слово «знати».

Рисунок 2.2 – Приклад слів з тексту після лематизації.

Основна мета лематизації полягає в зниженні розмірності словникового простору та поліпшенні точності обробки тексту моделями машинного навчання. Після виконання всіх попередніх дій буде отримано який буде підготовлений для подальшої роботи тому переждемо до наступного етапу.

Етап вбудовування даних – це комплексний процес який перетворює слова або токени тексту в числові вектори, що використовуються для подальшого аналізу та розуміння тексту комп'ютерами основна ціль цього процесу полягає в тому щоб представити близькі за контекстом та або семантикою слова близькими числовими векторами. Цей етап включає в себе такі процеси як тонізація та застосування методів вбудовування даних. Перше що потрібно зробити на цьому етапі це провести тонізацію даних.

Токенізація – це процес поділу великого фрагменту тексту на менші частини. Зазвичай це робиться для того, щоб кожну невелику частину або токен можна було зіставити зі значущою одиницею інформації[8].

```
['Жид' ,'мудрує', 'як' 'би' 'кого' 'надуть']
```

Рисунок 2.3 – Приклад тексту після токенізації.

Токенами називаються окремі одиниці тексту, на які він буде розбитий. А на те саме як він буде розбитий впливає обраний тип N-грами – це комбінації з N слів разом таких комбінацій може бути безліч але зазвичай їх поділяють на 1-гами , 2-грами та 3-грами , число в назві показує скільки слів містить один токен

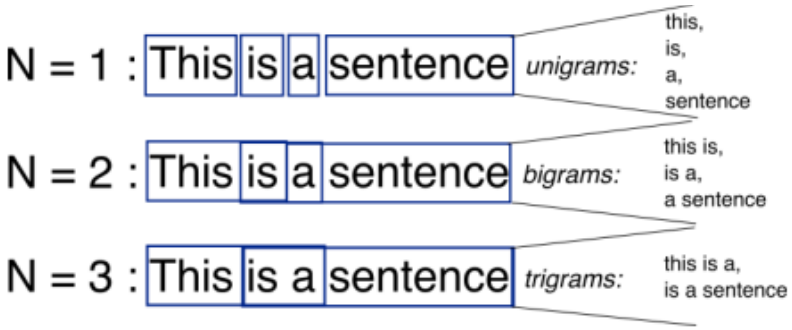


Рисунок 2.4 – Приклад застосування N-грам (https://suyashkhare619.medium.com/how-to-deal-with-multi-word-phrases-or-n-grams-while-building-a-custom-embedding-eec547d1ab45).

Після завершення процесу токенізації потрібно обрати один з методів вбудовування даних для переводу тексту в числові вектори. Основними методами вбудовування даних є мішок слів та TF-IDF.

Мішок слів – це найпростіший метод вбудовування слів у числові вектори. Він не часто використовується на практиці через надмірне спрощення мови, але зазвичай зустрічається у прикладах та навчальних посібниках[8].Він використовується для представлення тексту найпростішій формі шляхом підрахунку кількості зустрічей конкретного слова в тексті.

	1 This	2 movie	3 is	4 very	5 scary	6 and	7 long	8 not	9 slow	10 spooky	11 good	Length of the review(in words)
Review 1	1	1	1	1	1	1	1	0	0	0	0	7
Review 2	1	1	2	0	0	1	1	0	1	0	0	8
Review 3	1	1	1	0	0	0	1	0	0	1	1	6

Рисунок 2.5 – Приклад кодування тексту мішком слів (https://www.analyticsvidhya.com/blog/2020/02/quick-introduction-bag-of-words-bow-tf-idf/).

Використання мішка слів має наступні недоліки

- Збільшення словникового запасу призводить до збільшення довжини векторів тексту. Такі вектори потребують більших обчислювальних можливостей для свого опрацювання.
- Вектори також міститимуть багато надлишкової інформації у вигляді «0» шо призведе до зниження точності їх аналізу.
- Вектори не зберігають інформації про граматику, та впорядкування слів у тексті. Що погано впиває на результати аналізу за наявності складних речень

TF-IDF – це метод вбудовування який на відміну від мішка слів, враховує відносну важливість кожного терміну для документа[8]. TF-IDF обчислюється таким чином:

$$\text{TF-IDF} = \text{TF} * \text{IDF}$$

TF є відносною частотою терміну і може бути розрахована наступним чином:

$$\text{TF} = \frac{\text{Кількість разів, коли термін з'являється в документі}}{\text{Загальна кількість термінів у документі}}$$

IDF є зворотною частотою документа і дозволяє зменшити вагу загальних термінів. і може бути розрахована наступним чином:

$$\text{IDF} = \log \frac{\text{Загальна кількість документів}}{\text{Кількість документів, в яких з'являється термін}}$$

Term	Review 1	Review 2	Review 3	IDF	TF-IDF (Review 1)	TF-IDF (Review 2)	TF-IDF (Review 3)
This	1	1	1	0.00	0.000	0.000	0.000
movie	1	1	1	0.00	0.000	0.000	0.000
is	1	2	1	0.00	0.000	0.000	0.000
very	1	0	0	0.48	0.068	0.000	0.000
scary	1	1	0	0.18	0.025	0.022	0.000
and	1	1	1	0.00	0.000	0.000	0.000
long	1	0	0	0.48	0.068	0.000	0.000
not	0	1	0	0.48	0.000	0.060	0.000
slow	0	1	0	0.48	0.000	0.060	0.000
spooky	0	0	1	0.48	0.000	0.000	0.080
good	0	0	1	0.48	0.000	0.000	0.080

Рисунок 2.6 – Приклад кодування тексту з допомогою TF-IDF (https://www.analyticsvidhya.com/blog/2020/02/quick-introduction-bag-of-words-bow-tf-idf/).

Для звершення етапу вбудовування потрібно обрати і скористатися одним з вищенаведених методів після чого буде отримано корпус даних повністю готовий до використання в сфері машинного навчання.

Етап підготовки та навчання моделей є найважливішим для вирішення будь якої задачі в будь-якій сфері машинного навчання. Цей етап включає в себе такі процеси як створення або вибір існуючої моделі яка може підійти для вирішення поставленої задачі та навчання цієї моделі на підготовлених даних.

Модель машинного навчання – це алгоритм, який був навчений розпізнавати певні типи шаблонів[9]. Моделі відіграють ключову роль при вирішені буть яких задач пов'язаних з машинним навчанням та будуються для прогнозування ймовірностей закономірностей або визначення послідовностей.

Моделі машинного навчання можна розділити на дві категорії а саме класичного та глибокого навчання.

Моделі класичного навчання – це моделі які базуються на класичних математичних алгоритмах [10].

Переваги моделей класичного навчання:

- Добре підходять для невеликих наборів даних
- Легкі у використані
- Не вимагає великої обчислювальних потужності

Недоліки моделей класичного навчання:

- Низька точність на великих наборах даних.
- Важко виявляти складні функції.

Моделі глибокого навчання – це моделі які базуються використані на нейронних мережах [10].

Переваги моделей глибокого навчання:

- Підходить для задач високої складності
- Хороша точність на великих наборах даних

Недоліки моделей глибокого навчання:

- Вимагає значної обчислювальної потужності

В рамках даної кваліфікаційної роботи застосовувались як класичні так і моделі глибокого навчання які підходять для вирішення задачі ідентифікації авторства. Розглянемо алгоритм роботи деяких класичних моделей:

Logistic Regression Незважаючи на свою назву, вона реалізована як лінійна модель для класифікації, а не як регресія, вона добре підходить як для бінарної:

$$\hat{p}(X_i) = \text{expit}\,(X_i w + w_0) = \frac{1}{1 + \exp\,(-X_i w - w_0)}$$

Так і для багаточленної класифікації.

$$\hat{p}_k(X_i) = \frac{\exp\left(X_i W_k + W_{0,k}\right)}{\sum_{l=0}^{K-1} \exp\left(X_i W_l + W_{0,l}\right)}$$

У цій моделі ймовірності, що описують можливі результати одного випробування, моделюються за допомогою логістичної функції[11].

Decision Tree Це непараметричний метод навчання, який використовується для класифікації та регресії. Метою є створення моделі, яка прогнозує цінність цільова змінна шляхом вивчення простих правил прийняття рішень, виведених з даних Функції. [11]

Multinominal NB Реалізує наївний алгоритм Байєса для мультиноміально розподілених даних і є використовуються в класифікації даних. Її параметри оцінюється згладженою версією максимальної правдоподібності, тобто підрахунком відносної частоти:

$$\hat{\theta}_{yi} = \frac{N_{yi} + \alpha}{N_y + \alpha n}$$

Multi-layer Perceptron (MLP) Це алгоритм, який маючи набір ознак і ціль, він може вивчити нелінійний аппроксиматор функцій для будь-якого класифікація або регресія. Вона відрізняється від логістичної регресії тим, що між вхідним і вихідним шаром може бути один або кілька нелінійних шари, звані прихованими шарами[11].

Розглянемо алгоритм роботи деяких глибоких моделей:

Long Short-Term Memory (LSTM) Це особливий тип рекурентних мереж глибокого навчання які були розроблені запам’ятовування послідовностей протягом великого періоду часу в першу чергу за рахунок вирішення проблеми зникаючого градієнта для цього в LSTM комірках використовують механізми “gating” (шлюзів) які контролюють процес запам'ятовування.

Шлюзи LSTM поділяються на:

- Input gate (Вхідні шлюзи) ці шлюзи вирішують, яка інформація буде зберігатися в довгостроковій пам'яті. Вони працюють лише з інформацією з поточного введення та короткочасної пам’яті з попереднього кроку. На цих воротах інформація фільтрується від змінних, які не є корисними[12].
- Forget gate (Шлюзи забуття) ці шлюзи вирішують яку інформацію з довгострокової пам’яті потрібно зберігати чи відкидати, це робиться шляхом множення вхідної довгострокової пам’яті на вектор забуття, створений поточним введенням і вхідною короткою пам’яттю[12].
- Output gate (Вихідні шлюзи) ці шлюзи візьмуть поточний вхід, попередню короткочасну пам’ять і нещодавно обчислену довгострокову пам’ять для створення нової короткочасної пам’яті, яка буде передана клітині на наступному часовому кроці. Вихідні дані поточного часового кроку також можна отримати з цього прихованого стану[12].

Використання мереж LSTM є досить ефективним способом для вирішення поставлених задач але є один суттєвий якщо порівнювати її наприклад з класичними моделями виявитеся що обчислення даних нейронних мереж відбувається досить повільно через їх складну структуру.

Gated Recurrent Unit (GRU) Це тип рекурентних мереж глибокого навчання який подібний до LSTM, але має менш складну реалізацію тому обчислюється значно швидше. GRU подумує механізм “gating” і прихований стан для контролю потоку інформації. Шлюзи GRU поділяються на:

- Update gate (Оновити шлюзи) ці шлюзи відповідають за визначення обсягу попередньої інформації, яку потрібно передати в наступний стан[12].
- Reset gate (Шлюзи скидання) ці шлюзи потрібні щоб вирішити, якою частиною минулої інформації потрібно знехтувати[12].

Під час роботи GRU першим в роботу , вступає шлюз скидання, він зберігає відповідну інформацію з минулого кроку в новому вмісті пам’яті, потім множить вхідний вектор та прихований стан на їхні ваги і обчислює по елементне множення між шлюзами скидання та попередньо прихованим станом. Після чого до цього всього застосовується нелінійна функція активації та генерується наступна послідовність.

Після завершення етапу роботи з моделями пора переходити до останнього але від того не менш вважливого етапу оцінки результатів. Етап оцінки результатів допомагає визначити, наскільки добре модель виконує поставлену задачу перед нею задачу. Основні метрики оцінки моделей:

- Матриця плутанини – це матриця яка показує кількість правильних та хибних класифікацій для кожного класу.
- Точність – це відношення кількості правильно класифікованих прикладів до загальної кількості прикладів.
- Втрати – це міра помилки моделі, яка вимірює, наскільки добре модель прогнозує вихідні значення.
- F-міра – середнє значення між точністю та повнотою. Потрібна для оцінки балансу між точністю і повнотою.

2.2. Висновки до розділу 2 Комп'ютерна технологія ідентифікації авторства

В підрозділі 2.1 «Ідентифікація авторства методами машинного навчання»: В цьому підрозділі було розглянуто повний алгоритм вирішення задачі в сфері обробки природної мови. Були визначені етапи створення, а саме: - Етап підготовки текстових даних, етап вбудовування даних, етап підготовки та навчання моделей та етап оцінки результатів. Кожен з етапів був розглянутий окремо і докладно описаний.

У першому підрозділі другого розділу була виконана підготовка та структурування даних для моделі, а саме нормалізація, очистка, та зниження розмірності словникового простору для поліпшенні точності обробки тексту.

Після підготовки даних була застосована токенізація та методи вбудовування даних. У якості основного методу був вибраний метод «мішок слів». Були оцінені переваги та недоліки методи, після чого він був задіяний. Для даного процесу були наведені супровідні формули і наочні таблиці представлення даних.

По завершенню підготовчих етапів був проведений підбір способів навчання моделі. Був виконаний порівняльний аналіз існуючих методів, їх переваги і недоліки та у результаті вибрані класичні моделі та моделі глибокого навчання які підходять для вирішення задачі ідентифікації авторства.

РОЗДІЛ 3. ПРОГРАМНИЙ ДОДАТОК LAV

3.1. Програмне середовище

. Метою його розробки було застосування результатів досліджень в області комп'ютерних технологій для ідентифікації авторства тексту.

Вся робота над програмним кодом здійснювалась в Visual Studio Code – це легкий, та дуже потужний редактор вихідного коду, який доступний на усіх популярних операційних системах. Він поставляється підтримкою багатьох мов програмування та технологій.

Цей додаток було реалізовано на мові програмування python тому що виходячи з умов поставленого завдання він здатний надати найбільшу кількість переваг порівняно з іншими мовами.

Використання python надало наступні переваги:

- Кросплатформеність дозволяє написати код лише один раз і потім просто використовувати його на різних платформах.
- Універсальність дасть можливість за потреби повністю змінити тип застосунку, python підтримує написання прикладних програм та веб-застосунків.
- Легкий синтаксис дозволяє менше відволікатися на нюанси самої мови і більше концентруватися на вирішені поставленої задачі.
- Велика база готових бібліотек дозволяє використовувати багато можливостей не замислюючись над тим як вони працюють в середні концентруючи свою увагу на вирішні задачі.

Для розробки додатку будо застосовано наступні бібліотеки:

Для роботи з моделями машиного навчання було використано бібліотеку Keras – це API глибокого навчання, написаний на Python, що працює поверх

платформи машинного навчання TensorFlow. Він був розроблений з акцентом на забезпечення швидких експериментів. Можливість перейти від ідеї до результату якомога швидше є ключем до проведення хороших досліджень[7].

Для роботи за даними використовувалась бібліотека Pandas – це бібліотека яка використовується для роботи з наборами даних[8].

Для роботи з укрвїнськми текстами використовувались бібліотеки simplemma – це бібліотека яка забезпечує простий і багатомовний підхід до пошуку базових форм слів. та langdetect – це бібліотека для розпізнання на якій мові було написано текст.

Для розробки інтерфейсу користувача було використано бібліотеку Streamlit – це бібліотека відкритим кодом, яка дозволяє легко створювати та ділитися красивими, користувацькими веб-програмами для машинного навчання та науки про дані[9]. Данна бібліотека дозволила створити гарний та функціоналіний користувацький веб інтерфейс без потреби виходити за екосистему python.

3. 2. Програмна реалізація

Програмний додаток LAV частково реалізує загальний алгоритм вирішення задач в сфері обробки природної мови. Але з однією відмінністю додаток не займається навчанням моделей а має в собі уже готові, вона має наступний алгоритм роботи

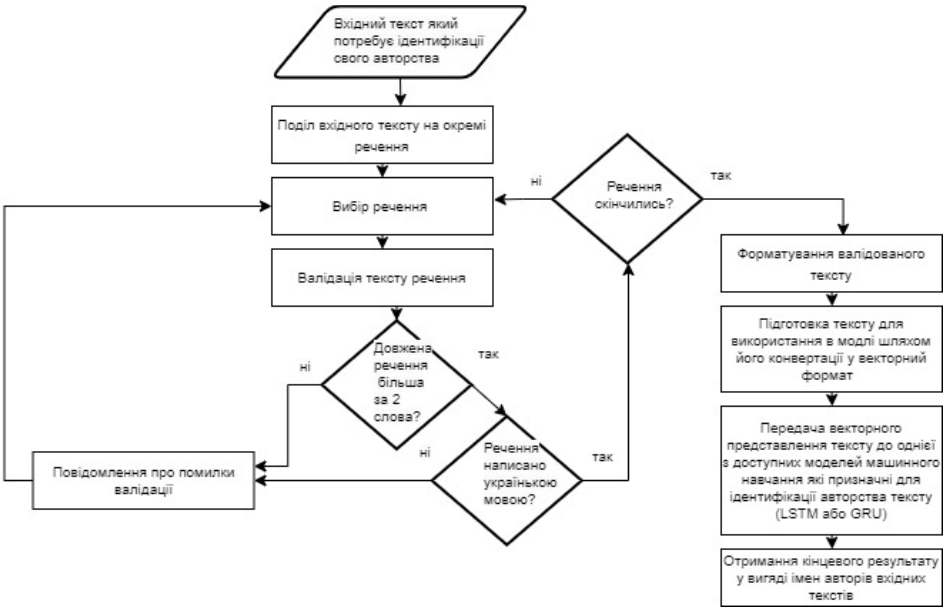


Рисунок 3.2.1 – Алгоритм роботи додатку LAV.

Додаток розроблявся на технологіях, вказаних в розділі «Програмне середовище» та має наступнну кодову стуктуру:

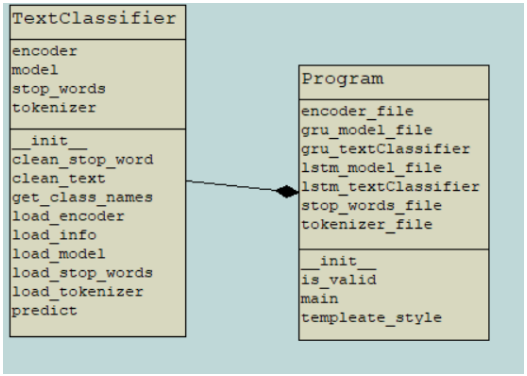


Рисунок 3.2.2 – Структура класів та функцій

Додаток має об'єктно орієнтовану структуру його функціонал розбитий на два основних класи:

- TextClassifier відповідає за всю роботу моделями машинного навчання.
- Program відповідає за роботу з функціями класу і відображає користувацький інтерфейс.

Дані класи мають наступний набір методів:

TextClassifier:

- load_model(model_file): завантажує модель з розпізнання автора
- load_tokenizer(tokenizer_file): завантажує токенізатор
- load_encoder(encoder_file): завантажує кодувальник .
- load_stop_words(stop_words_file): завантажує список стоп-слів.
- clean_stop_word(text): видаляє стоп-слова з тексту.
- clean_text(input_text): очищує та підготовляє текст для подальшої обробки в моделі ідентифікації авторства.
- predict(text): класифікує текст та повертає авторів.
- get_class_names(): повертає список назв класів моделі.

Program

- templateate_style(): стилізуйте шаблон Streamlit, приховуючи певні елементи та спеціальні стилі кнопок.
- is_valid(text): Перевіряє, текст на наявність в ньому української мови .
- main(): обєднує всі вищеперераховані функції стартує додаток.

3.3. Інстукція користувачу

Для запуску програми треба встановити інтерпретатор мови python версії 3.10 потім перейти в папку з програмою та створити віртуальне оточення для встановлення бібліотек для цього потрібно відкрити термінал або командний рядок. Та виконати наступну команду «python3 -m venv lav». Ця команда створить нове віртуальне середовище з назвою «lav» у поточній папці. За потреби замість «lav» можна використовувати іншу бажану назву. Після створення віртуального середовища його потрібно активувати для різних платформ для цього існують різні команди:

- Для Windows – «lav\Scripts\activate»
- Для macOS/Linux – «source lav/bin/activate»

Після запуску віртуального середовища потрібно повернутися в корінь папки програми і виконати команду «pip install requirements.txt» це встановить усі необхідні залежності які потрібні для запуску програмного додатку.

Додаток працює на базі бібліотеки Streamlit тому для її запуску потрібно виконати наступну команду «streamlit run lav.py» вона запустить локальний сервер на комп'ютері користувача і додаток буде доступний за посиланням «http://localhost:8501» . Зараз процес запуску є досить складним але за потреби його можна зробити значно простішим за рахунок переміщення додатку з локальної машини на віддалений хостинг який підтримує програмні додатки написані на мові програмування python , або скористатися хмарним сховищем від розробників бібліотеки streamlit яке називається «Streamlit Community Cloud».

Така зміна розташування додатку буде корисна тим що усі дії повязані з налаштуванням перед запуском доведется виконати лише один раз і в подальшому запускати цей доданок можна буде запускати просто переходячи наявним на нього посиланням.

Коли користувач переходить за посиланням, перед ним з’являється вікно програмного додатку LAV. Його інтерфейс є одно сторінковим та працює у реальному часі без потреби поновляти сторінку браузера.

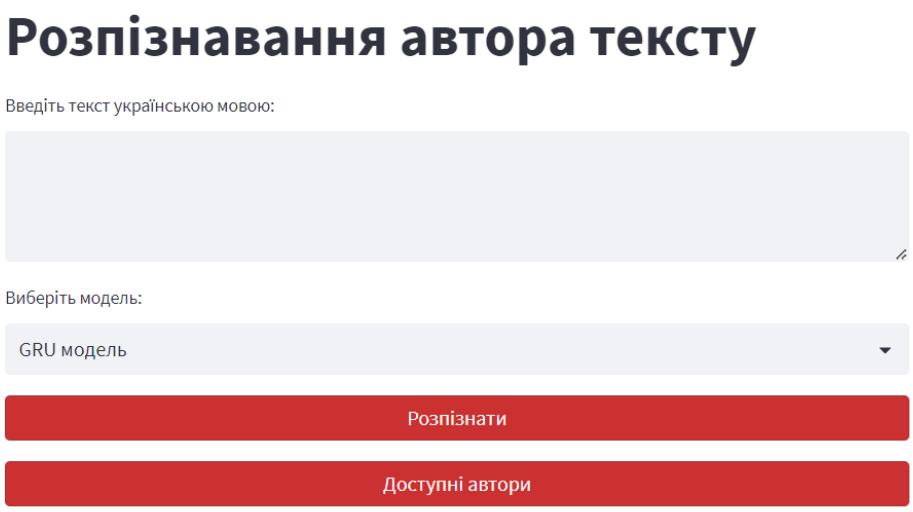


Рисунок 3.3.1 – Початкове відображення інтерфейсу користувача.

Перед початком процесу ідентифікації авторства текстів рекомендується переглянути список доступних для ідентифікації авторів для цього потрібно натиснути кнопку «Доступні автори»

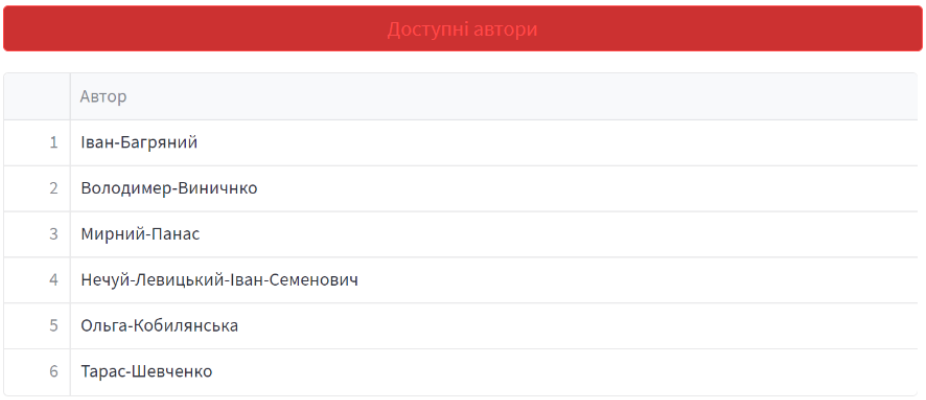


Рисунок 3.3.2 – Список авторів доступних для ідентифікації.

На даний момент LAV здатен розпізнавати твори тільки шістьох українських авторів такі обмеження це лише перша версія даної цього додатку у наступних версіях кількість авторів буде збільшуватися.

Для того щоб розпочати процес ідентифікації авторства потрібно ввести текст у поле для вводу текстів і натиснути на кнопку «Розпізнати» після чого під нею з’явитеся таблиця яка покаже авторство введеного тексту.

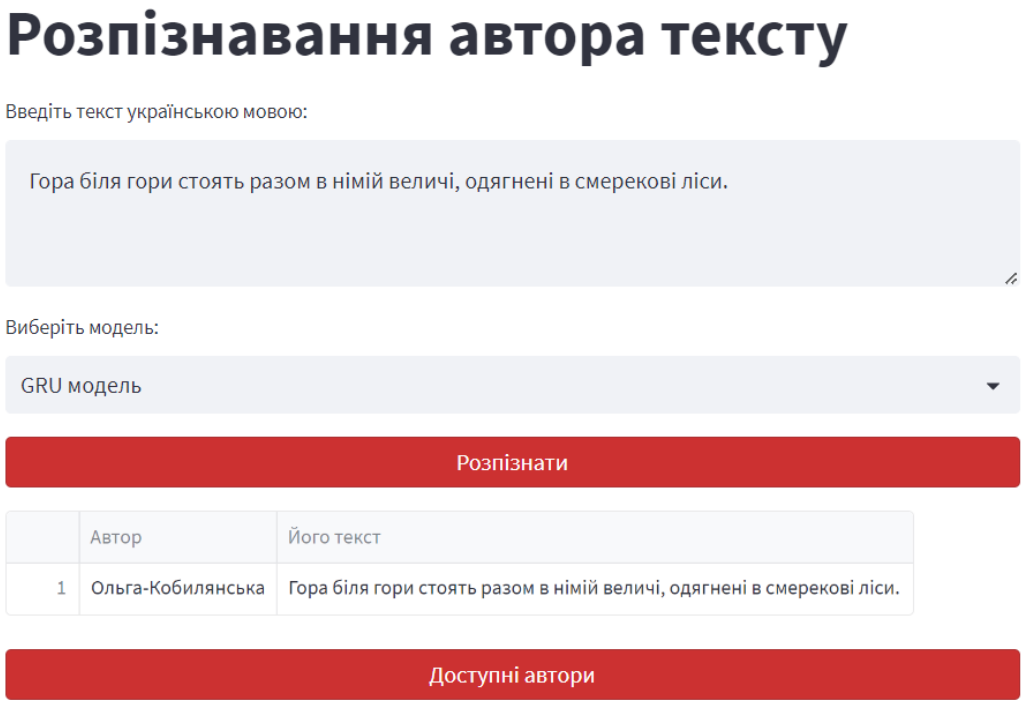


Рисунок 3.3.3 – Результати ідентифікації на прикладі окремого речення.

В цій реалізації програмного додатку процес ідентифікації авторства відбувається завдяки двом рекурентним моделям глибокого машинного навчання різних типів а саме: LSTM та GRU. Вибір саме цих моделей був зумовлений тим що вони показали показали найкращі результати під час процесу досліджень. Для того щоб обрати тип моделі потрібно натиснути на елемент списку з назвою «Виберіть модель»

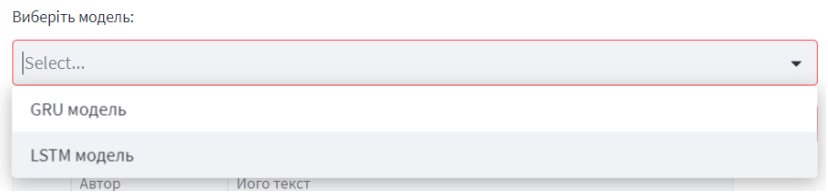


Рисунок 3.3.4 – Список вибору типу моделі.

Дані моделі видають результати які дуже близькі один до одного але все ж мають деякі відмінності які буде розглянуто вже в розділі тестування цього додатку. Достаток працює з фрагментами тексту у форматі окремих речень що дає змогу ідентифікувати авторство фрагментів тексту які були які були написані в співавторстві або зібрані з фрагментів текстів різних авторів.

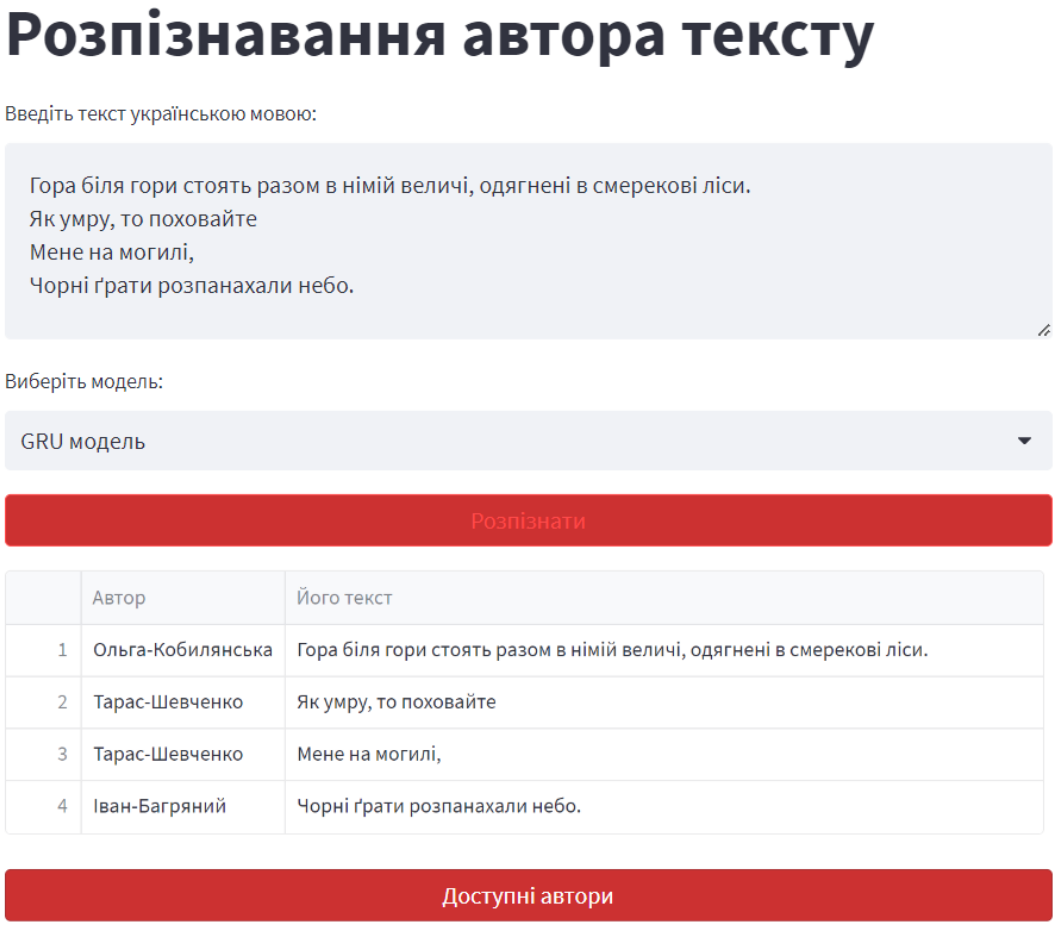


Рисунок 3.3.5 – Результати ідентифікації фрагменту тексу зібраного з речень різних авторів.

Щоб збільшити точність процесу ідентифікації авторства тексту в даному додатку передбачається дві основних вимоги до вхідного тексту а саме: Одне речення повинно бути довшим за два слова і текст речення повинен бути українською мовою при порушені даних вимог речення не буде ідентифіковано і додаток виведе відповідну помилку.

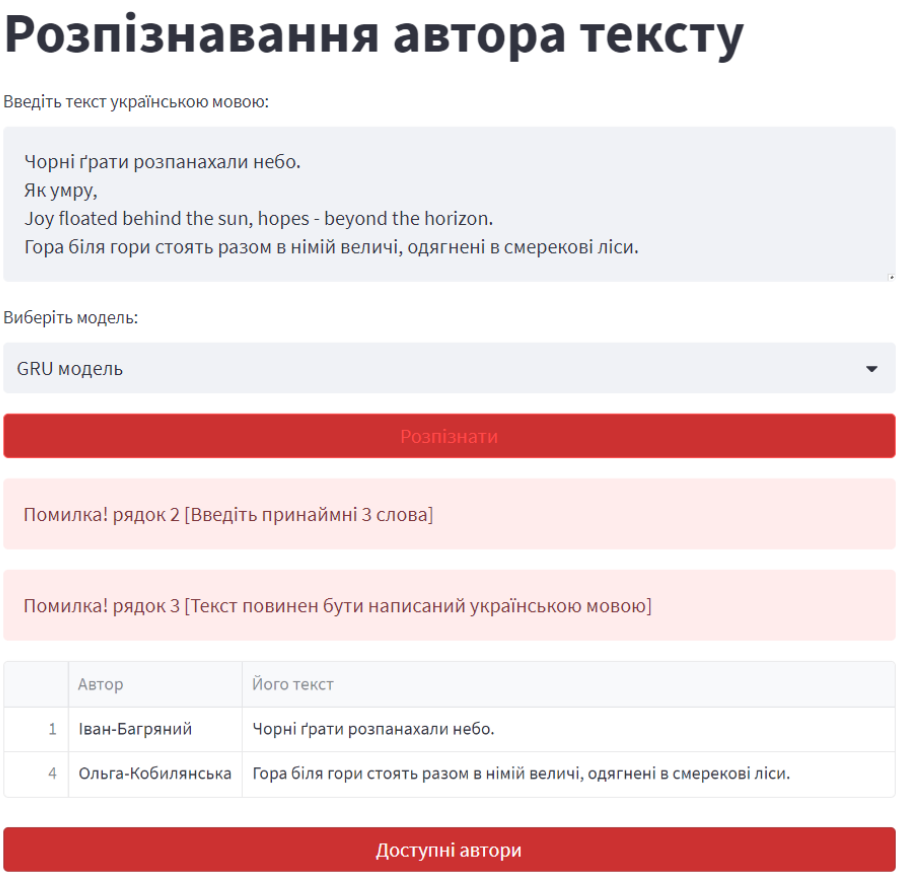


Рисунок 3.3.5 – Робота системи перевірки даних на відповідність вимогам.

3.4. Тестування роботи додатку

Метою даного розділу є проведення невеликого тестування програмного додатку LAV для того щоб впевнитися що він працює правильно. Він був протестований на творах: Івана Багряного, Ольги Кобилянської і Тараса Шевченка. Тексти подавались у наступному форматі: по два фрагмента тексту для кожного автора ,фрагменти не більше чотирьох речень. В процесі тестування було отримано наступні результати:

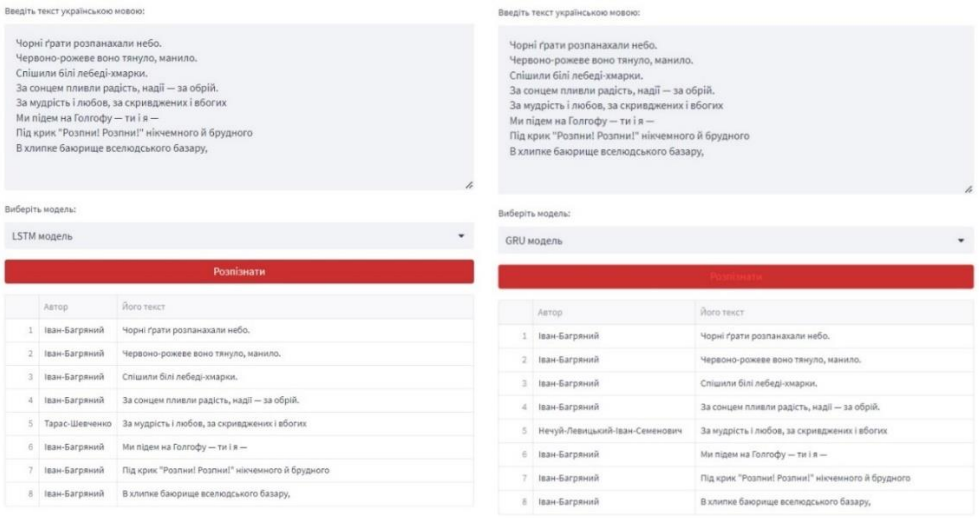


Рисунок 3.4.1 – Результати процесу тестування, автор Іван Багряний.

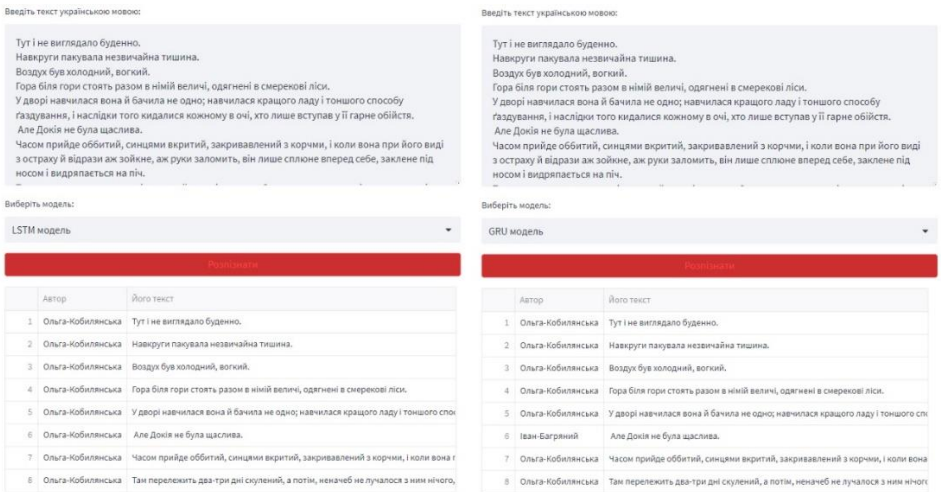


Рисунок 3.4.2 – Результати процесу тестування, автор Ольга Кобилянська.

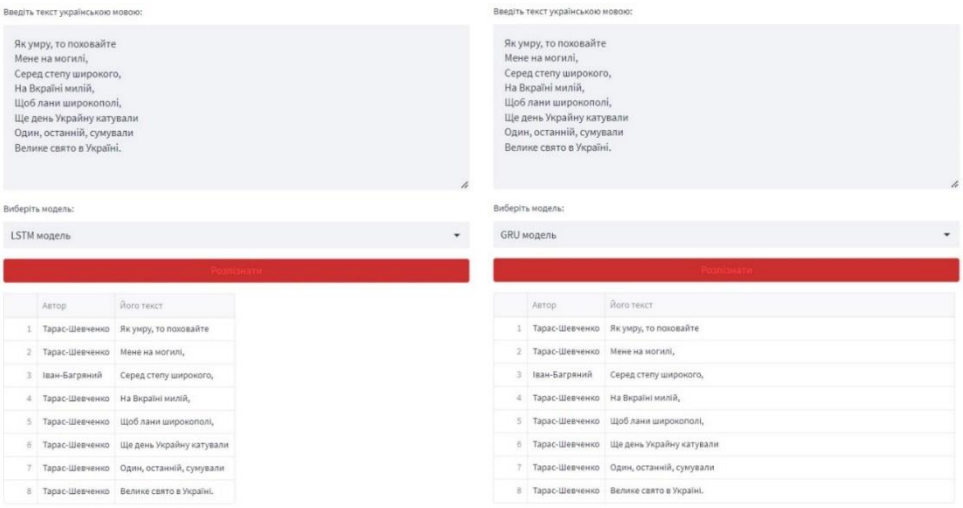


Рисунок 3.4.3 – Результати процесу тестування, автор Тарас Шевченко.

Тестування показало що середній показник успішної ідентифікації авторства тексту становлять 7 з 8 речень що є приблизно 87.5% шансом на успішну ідентифікацію автора – це доволі непоганий результат.

Якщо дивитись на результати спроб ідентифікації авторства для кожного типу моделей окремо то показники будуть наступні: Модель GRU змогла розпізнати 7 з 8 речень при трьох спробах ідентифікації тексту що можна інтерпретувати як приблизно 87.5% шанс на успіх. З моделлю LSTM все трохи цікавіше вона змогла розпізнати 7 з 8 в речень при першій та третій спробі та 8 з 8 при другій що в середньому дає 91.67% шанс на успішну ідентифікацію авторства тексту. І більше ніж результати роботи CRU на 4.17% тому буде доцільно використовувати її як основну модель для ідентифікації авторства тексту. А GRU як допоміжну.

3.5. Висновки до розділу 3 Програмний додаток LAV

У даному розділі була поставлена задача реалізації змодельованої у розділі 2 системи з машинним навчанням, були виконані такі етапи роботи: моделювання структури системи, вибір інструментів реалізації, вибір середовища реалізації, дизайн і проектування інтерфейсу системи та програмна реалізація системи.

Для наочності подачі інформації у даному розділі були наведені блок-схеми, таблиці та зображення інтерфейсу системи. Окрім цього був наведений список і опис кожної з задіяних у системі функцій.

У результаті був створений додаток, що може працювати з фрагментами тексту, представленими у вигляді окремих речень, що дає змогу ідентифікувати авторство фрагментів тексту які були які були написані в співавторстві або зібрані з фрагментів текстів різних авторів.

Після виконання повної програмної реалізації додатку LAV було виконане тестування його працездатності. Для цього попередньо були вибрані і структуровані тестові дані у вигляді творів деяких українських письменників. Тестування проводилося на двох моделях, а саме GRU та LSTM. У результаті було визначено, що модель GRU успішно ідентифікує авторство тексту у 87.5% випадків, а модель LSTM дає 91.67% шанс на успішну ідентифікацію. Був зроблений висновок, що модель LSTM необхідно використовувати у якості основної моделі системи, а модель GRU у якості допоміжної. Отже, таким чином була досягнута найкраща продуктивність даного додатку LAV. У результаті всі поставлені у цьому розділі задачі були виконані.

РОЗДІЛ 4. ОБЧИСЛЮВАЛЬНИЙ ЕКСПЕРИМЕНТ

4.1Програмне середовище

Під час обчислювального експерименту виконувалась в хмарному сервісі Google Colaboratory – цей web-сервіс спрямований на спрощення процесу досліджень в галузі машинного навчання. Сервіс надає доступ до машин з підключеною відеокартою та великим об’ємом оперативної пам’яті що значно прискорює процес роботи. Також сервіс надає вже підготовлену екосистему для процесу досліджень за рахунок того що основні бібліотеки які можуть знадобитися для роботи в сфері машинного навчання вже встановлено за умовчанням. Експеримент проводився з використанням мови python та наступних бібліотек ось декілька з уже згаданих раніше Keras був використаний для роботи з моделями глибокого навчання. Pandas використовувався для роботи з даними. Simplemma використовувались для роботи з укрвїнськми текстамми.

Що стосується використання нових бібліотек: Нові бібліотеки для роботи з даними: NLTK ¬ це пакет бібліотек і програм для символьної і статистичної обробки тексту він використовується для розробки програм, що працюють з мовою, зокрема для комп'ютерної або емпіричної лінгвістикою, когнітивістикою, штучним інтелектом, інформаційним пошуком і машинним навчанням. Та NumPy – це бібліотека що додає підтримку великих багатовимірних масивів та матриць, разом з великою колекцією функцій математичної обробки даних цих масивів. Для представлення графіків та діаграм було використано: Matplotlib – це бібліотека яка дозволяє створювати графіки, діаграми, гістограми та інші візуалізації даних Scikit-learn – це бібліотека класичних методів машинного навчання яка містить набір інструментів для класифікації, регресії, тощо

4.2. Корпус даних

Дані є основою для будь-якої задачі в галузі машинного навчання, оскільки саме на їх основі будується модель, для вирішення поставленої задачі В рамках даної кваліфікаційної роботи було сформовано окремий корпус даних, який містить тексти шести відомих українських авторів в ньому є твори: Івана Багряного, Володимира Винниченка, Панаса, Мирного Ольги Кобилянської , Івана Нечуй-Левицького, і Тараса Шевченка. Корпус представлений у форматі CSV файлу та має наступні характеристики:


```
Data columns (total 2 columns):
 #   Column   Non-Null Count    Dtype
---  ------   --------------    -----
 0   authors  1223994 non-null  object
 1   text     1223994 non-null  object
dtypes: object(2)
memory usage: 18.7+ MB
```

Рисунок 4.2.1 – Характеристики корпусу українських авторів.

Цей корпус є достатньо великим і тому добре підходить для навчання мовних моделей , представлений у табличному форматі є нормалізованим та збалансованим.

	authors	text
0	Ольга-Кобилянська	взяти поле політок ви хотіти частю хліб казати...
1	Нечуй-Левицький-Іван-Семенович	жупан пускати потонути дно
2	Тарас-Шевченко	кайдани погнати
3	Ольга-Кобилянська	шаліла заметільниця страшний чувана бачити ніч...
4	Іван-Багряний	сизий сон

Рисунок 4.2.2 – Представлення корпусу даних у табличному форматі.

Колонка «authors» зберігає в собі імена авторів а колонка «text» зберігає очищені від розділових знаків та стоп-слів тексти які пройшли процес лематизації та були розбиті на окремі речення кожне з яких належить відповідному автору.

Етап підготови даних Це комплексний процес підготовки текстових даних до їх використання в моделях машинного навчання. Алгоритм процесу підготовки даних виглядає наступним чином:

Рисунок 4.2.3 – Алгоритм процесу підготовки даних.

Докладніше про процеси які відбувалися на кожному етапі підготовки даних написано нижче:

Збір даних На цьому етапі вирішувалась доля тематики набору даних у підсумку вибір зроблено на користь творів української літератури тому було знайдено та зібрано набір текстів української літератури. Всі тести були взяті з відкритих джерел та збережено docx тому по завершені даного етапу корпус виглядає наступним чином:

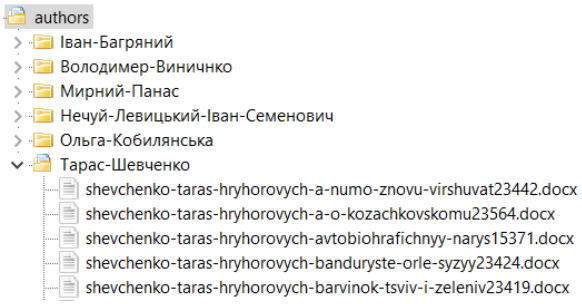


Рисунок 4.2.4 – Структура корпусу після етапу збору даних.

Після завершення даного можна переходити до етапу нормалізації даних.

Нормалізація даних – В даному конкретному випадку процес нормалізації працює в два етапи він передбачає об’єднання усіх творів кожного з авторів в окремий великий файл для кожного автора щоб полегшити подальший процес роботи с даними завдяки тому , що ці дані будуть зібрані в одному місці а потім процес нормалізації тексту вже в об’єднаних текстах авторів. Для вирішення даної задачі було написано скрипт результати його роботи відображено нижче:

```
Створення файлу даних...
Тексти було збережено output/txt/Іван-Багряний.txt 34505 речень
Тексти було збережено output/txt/Володимер-Виничнко.txt 47303 речень
Тексти було збережено output/txt/Мирний-Панас.txt 34598 речень
Тексти було збережено output/txt/Нечуй-Левицький-Іван-Семенович.txt 50082 речень
Тексти було збережено output/txt/Ольга-Кобилянська.txt 44978 речень
Тексти було збережено output/txt/Тарас-Шевченко.txt 37067 речень
```

Рисунок 4.2.5 – Об’єднання текстів автора в один файл.

Даний скрипт даний скрипт пройшовся по усьому тексту який був йому доступний записавши усі твори кожного з авторів в окремі відповідні їх іменам файли. Це виглядає наступним чином:

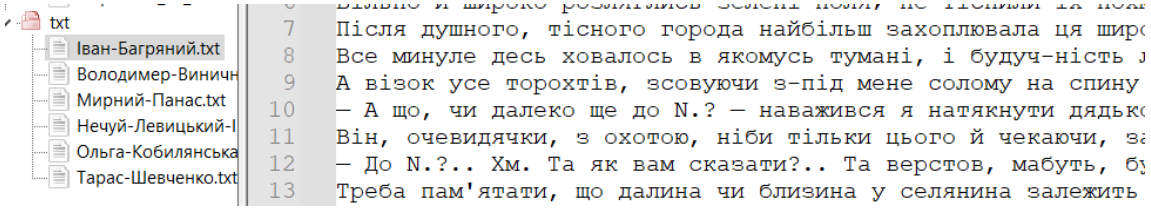


Рисунок 4.2.6 – Формат корпусу після об’єднання текстів автора в один файл.

Після чого скрипт проводить процес нормалізації тексту за усними канонами даного етапу.

```
через тиждень нимидорі знов приснився сон
ніби вона блукає з миколою вночі по якомусь
широкому степу
```

Рисунок 4.2.7 – Вид даних після нормалізації.

За етапом нормалізації тексту йде етап вилучення стоп-слів для проведення даного процесу існує декілька готових програмних рішень але в даному випадку було написано власне програмне рішення яке б краще підходило для роботи з текстами написаними українською мовою. Воно приставляє з себе скрипт на мові python і влаштоване наступним чином спочатку було зібрано базу самих популярних українських стоп-слів та збережено їх до текстового файлу запускаєтеся скрипт який проходить по уже нормалізованому тексту і перевіряє його на наявність співпадінь слів з тексту з словами що знаходяться в файлі з стоп-словами якщо співпадіння є то такі слова видаляються з основного тексу. Базу слів і процес роботи скрипту відображений нижче.

```
Пребробка набору даних для аналізу...
Видалення стоп-слів..
Усього  2101 стоп-слів
Стоп слова знайдені в тексті 1394 слів
```

Рисунок 4.2.8 – Процес роботи скрипту який видаляє стоп-слова.

```
['а', 'аби', 'абиде', 'абиким', 'абикого', 'абиколи', 'абикому', 'абикуди', 'абихто', 'абичий', 'абичийого', 'абичийому',
'абичим', 'абичию', 'абичия', 'абичие', 'абичиєму', 'абичиєю', 'абичиєї', 'абичиї', 'абичиїй', 'абичиїм', 'абичиїми',
'абичиїх', 'абичого', 'абичому', 'абищо', 'абияка', 'абияке', 'абиякий', 'абияким', 'абиякими', 'абияких', 'абиякого',
'абиякому', 'абиякою', 'абиякої', 'абияку', 'абиякі', 'абиякій', 'абиякім', 'або', 'абощо', 'авжеж', 'авось', 'ага',
'ад', 'адже', 'аж', 'ажень', 'аз', 'ай', 'але', 'ало', 'амінь', 'ант', 'ану', 'ані', 'аніде', 'аніж', 'анізащо',
'аніким', 'анікого', 'анікогісінько', 'аніколи', 'анікому', 'аніскільки', 'аніхто', 'анічим', 'анічого', 'анічогісінько',
'анічому', 'аніщо', 'аніяка', 'аніяке', 'аніякий', 'аніяким', 'аніякими', 'аніяких', 'аніякого', 'аніякому', 'аніякою',
```

Рисунок 4.2.9 – Фрагмент бази українських стоп слів.

Після завершення даного можна переходити до етапу лематизації.

Для проведення процесу лематизації корпусу даних було написано скрипт з використанням бібліотеки simplemma даний скрипт проходить по доступному йому тексту та приводить слова до базові форми.

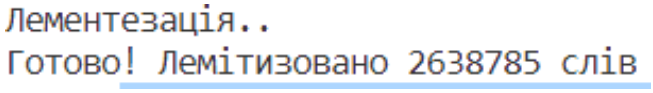


Рисунок 4.2.10 – Процес роботи скрипту який проводить лематизацію даних.

Після закінчення процесу лематизації та попередніх процесів обробки етап підготовки даних можна вважати завершеним залишається один нюанс а саме форма представлення корпусу оскільки зараз він приставляє з себе лише набір даних в текстовій формі. Така форма представлення даних передбачає багато непотрібної роботи з цими даними в подальшому тому краще конвертувати дані в інший формат. Під час дослідження цієї теми вибір було зроблено користь табличного представлення даних воно представляє дані у вигляді таблиці що дозволяє зберігати їх в структурованому форматі тому буде легше працювати у подальшому Кожен стовпець у такій таблиці відображає різні ознаки документу в даному випадку з наявного корпусу даних можна виділити дві ознаки, перша це автор а друга це його текст.

Для конвертації даних у табличну форму було написано скрипт який виконує роботу в декілька простих кроків:

- створюється масив формату ключ та його значення (автор та усі його тексти) .
- Крок другий скрипт розбиває текст автора на речення після чого прив’язує до кожного отриманого речення відповідного йому автора (створюється масив масивів)
- Крок третій зберігає скрипт бере кожне значення створеного масиву масивів та зберігає його в таблицю у довільному порядку.

```
Створення набору даних для аналізу...
Іван-Багряний : 34505  речень
Володимер-Виничнко : 47303  речень
Мирний-Панас : 34598  речень
Нечуй-Левицький-Іван-Семенович : 50082  речень
Ольга-Кобилянська : 44978  речень
Тарас-Шевченко : 37067  речень
```

Рисунок 4.2.11 – Робота скрипту який конвертує дані в табличний вид.

Після роботи цього скрипту буде отримано корпус даних наступного виду

	authors	text
0	Ольга-Кобилянська	взяти поле політок ви хотіти частю хліб казати...
1	Нечуй-Левицький-Іван-Семенович	жупан пускати потонути дно
2	Тарас-Шевченко	кайдани погнати
3	Ольга-Кобилянська	шаліла заметільниця страшний чувана бачити ніч...
4	Іван-Багряний	сизий сон

Рисунок 4.2.12 – Представлення перших записів в корпусі даних.

Після усіх пройдених етапів було отримано корпус даних який можна використовувати в моделях машинного навчання . Але перед тим як приступати до навчання моделей на основі даного корпусу потрібно звернути увагу на ще одну невелику але важливу деталь а саме на те що автори які є в цьому корпусі мають різну кількість речень у своїх творах тим самим спричинюючи дисбаланс в корпусі даних.

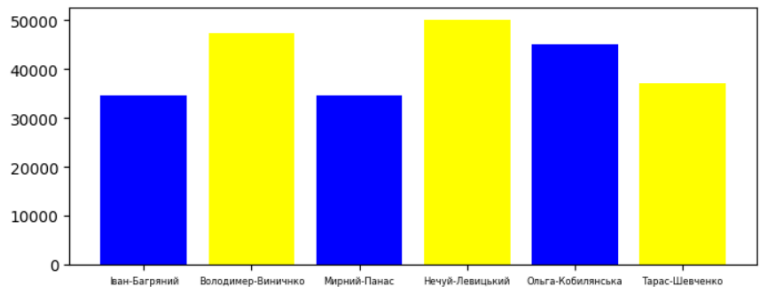


Рисунок 4.2.13 – Графічне відображення дисбалансу даних в корпус

Дисбаланс даних в корпусі може призвести до погіршення якості навчання моделі, Існує багато методів для вирішення проблеми дисбалансу від збільшення обсягів даних текстів до зменшення кількості об’єктів класифікації. В даному випадку для вирішення цієї проблеми було використано техніку нормалізації даних під назвою даунсемплінг – це спеціальна техніка для обробки даних яку можна використовувати для балансування яке це робиться шляхом випадкового відбору якоїсь підмножини з основних даних. Для застосування цього методу до корпусу даних було змінено скрипт який перетворює дані в табличний вид і тепер перед тим як зберегти дань до таблиці він застосовує техніку даунсемплінгу.

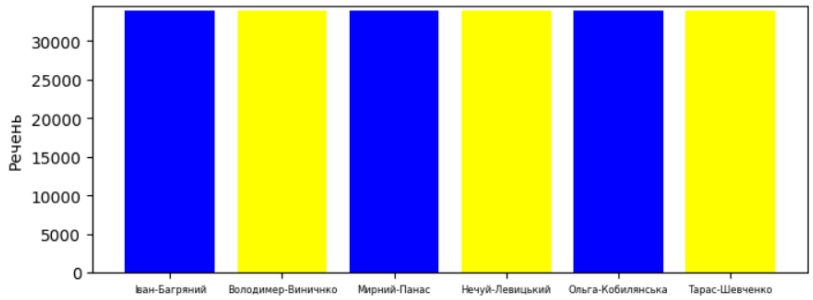


Рисунок 4.2.14 – Графічне відображення балансованого корпусу даних

Після завершення даного процесу корпус можна вважати повністю готовим для подальшого використання в моделях машинного навчання.

4.3. Дослідження ефективності класичних моделей машинного навчання

В рамках даного підрозділу було проведено аналіз ефективності чотирьох класичних моделей машинного навчання які підходять для вирішення задачі ідентифікації авторства тексту. В термінах NLP дану задачу можна класифікувати як задачу багатоколоскової класифікації – призначення тексту двох або більше категорій. В даному випадку категорією можна вважати автора тексту. Моделі для дослідження обирались спираючись на статтю [2]

Було досліджено ефективність наступних моделей:

- Logistic Regression
- Decision Tree
- Multinominal NB
- Multi-layer Perceptron

Відні дані: Корпус авторів української літератури сформований в розділі 4.2

Методи вбудовування: Мішок слів та TF-IDF

Способи тонізації : 1-грами та 1-грами + 2-грами

Алгоритми моделей реалізовано з допомогою бібліотеки Scikit-learn

Результати досліджень:

	Model	Accuracy	Precision	Recall	F1-score
1	LogisticRegression	84.26%	84.75%	84.26%	84.34%
2	DecisionTreeClassifier	95.80%	95.83%	95.80%	95.81%
3	MultinomialNB	77.08%	77.46%	77.08%	77.03%
4	MLPClassifier	86.39%	86.72%	86.39%	86.45%

Рисунок 4.3.1 – Результати роботи моделей мішок слів та 1-грами

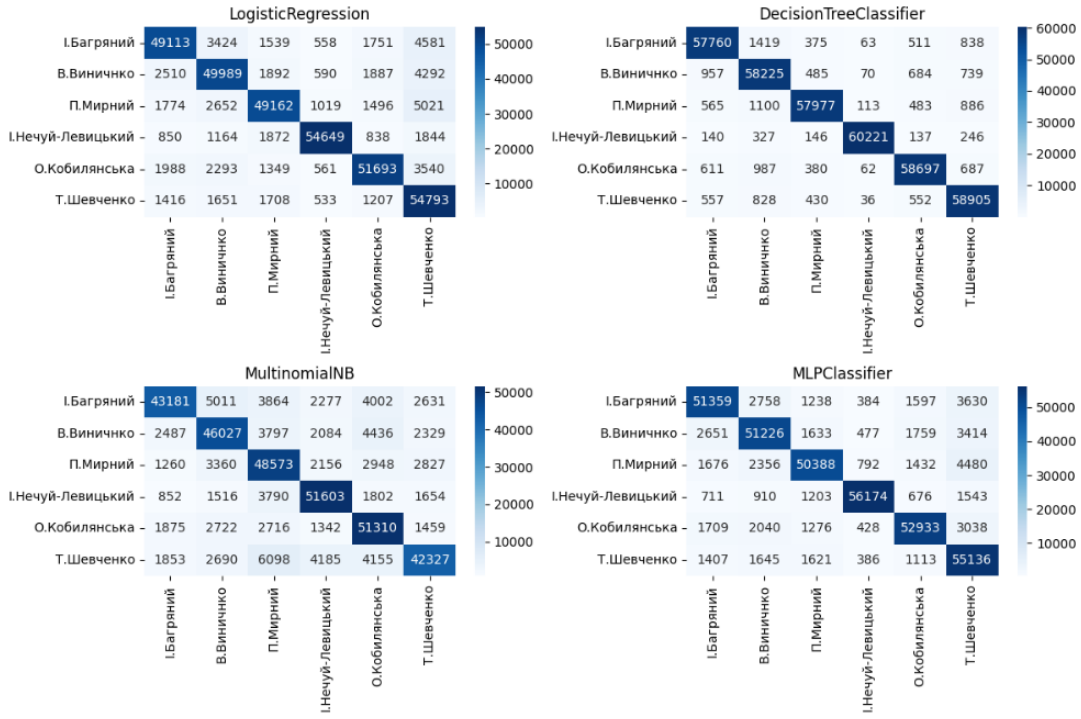


Рисунок 4.3.2 – Матриця похибки моделей мішок слів та 1-грами

	Model	Accuracy	Precision	Recall	F1-score
1	LogisticRegression	80.10%	80.48%	80.10%	79.94%
2	DecisionTreeClassifier	46.67%	52.10%	46.67%	46.15%
3	MultinomialNB	78.78%	78.88%	78.78%	78.77%
4	MLPClassifier	86.45%	86.42%	86.45%	86.40%

Рисунок 4.3.3 – Результати роботи моделей мішок слів та 1-грами та 2-грами

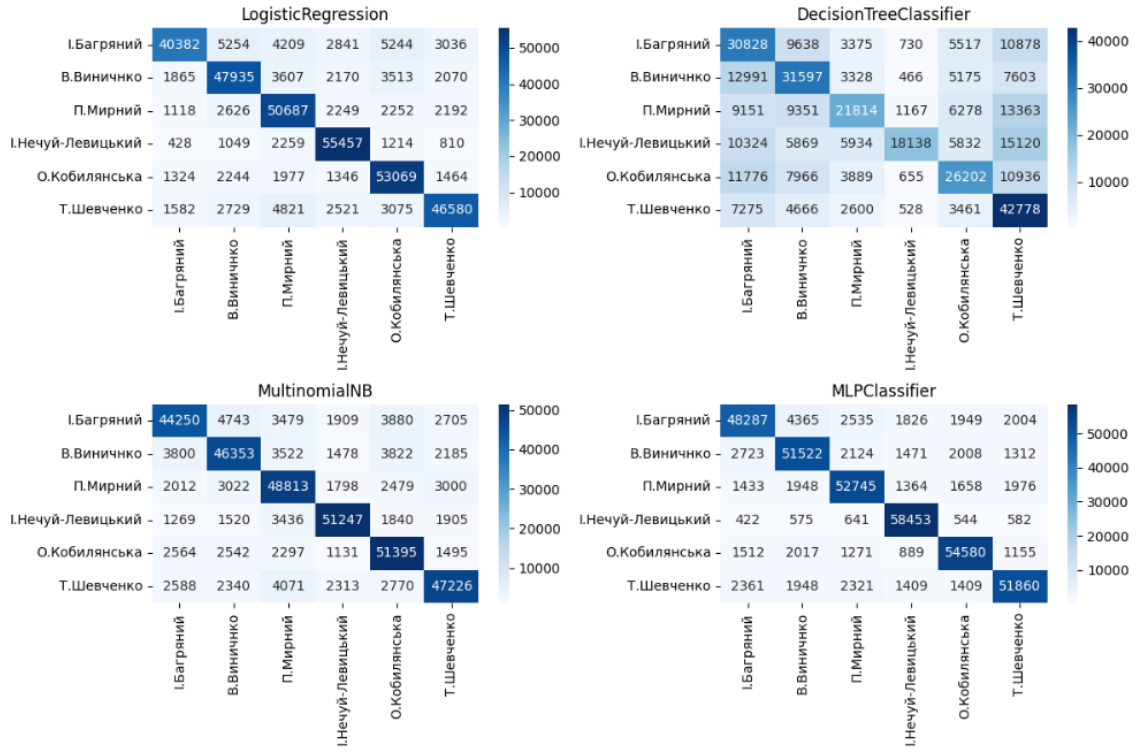


Рисунок 4.3.3 – Матриця похибки моделей мішок слів та 1-грами та 2-грами

	Model	Accuracy	Precision	Recall	F1-score
1	LogisticRegression	82.50%	82.63%	82.50%	82.52%
2	DecisionTreeClassifier	95.69%	95.72%	95.69%	95.70%
3	MultinomialNB	79.26%	79.45%	79.26%	79.23%
4	MLPClassifier	84.88%	85.05%	84.88%	84.91%

Рисунок 4.3.4 – Результати роботи моделей TF-IDF та 1-грами

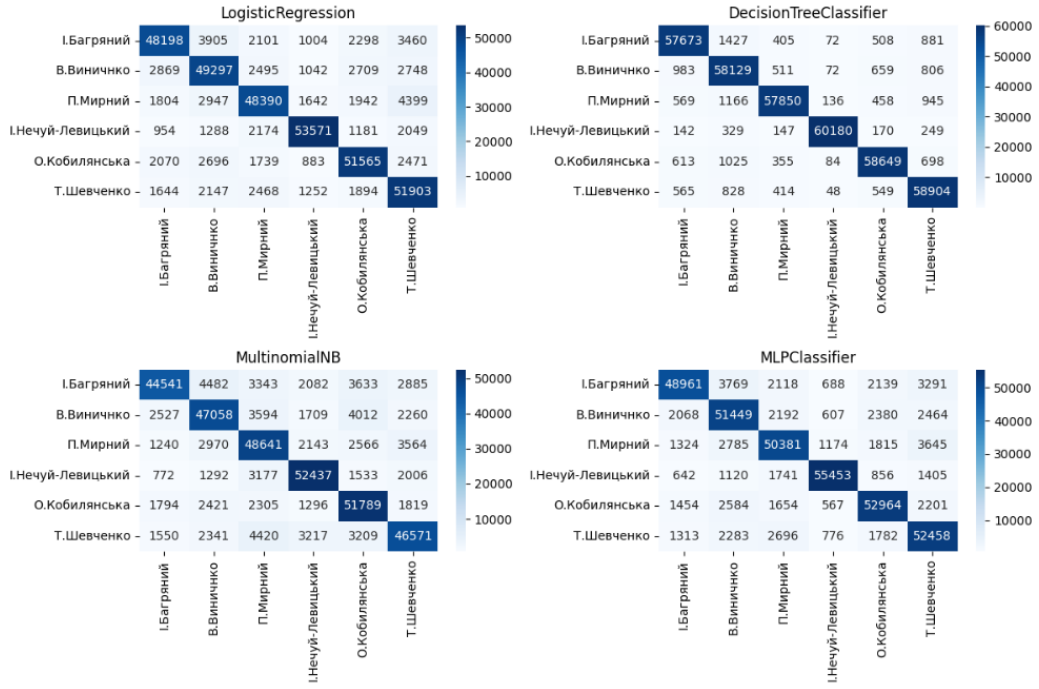


Рисунок 4.3.5 – Матриця похибки моделей TF-IDF та 1-грами

	Model	Accuracy	Precision	Recall	F1-score
1	LogisticRegression	84.56%	84.63%	84.56%	84.56%
2	DecisionTreeClassifier	94.47%	94.52%	94.47%	94.48%
3	MultinomialNB	79.98%	80.02%	79.98%	79.96%
4	MLPClassifier	88.58%	88.67%	88.58%	88.60%

Рисунок 4.3.6 – Результати роботи моделей TF-IDF та 1-грами з 2-грами

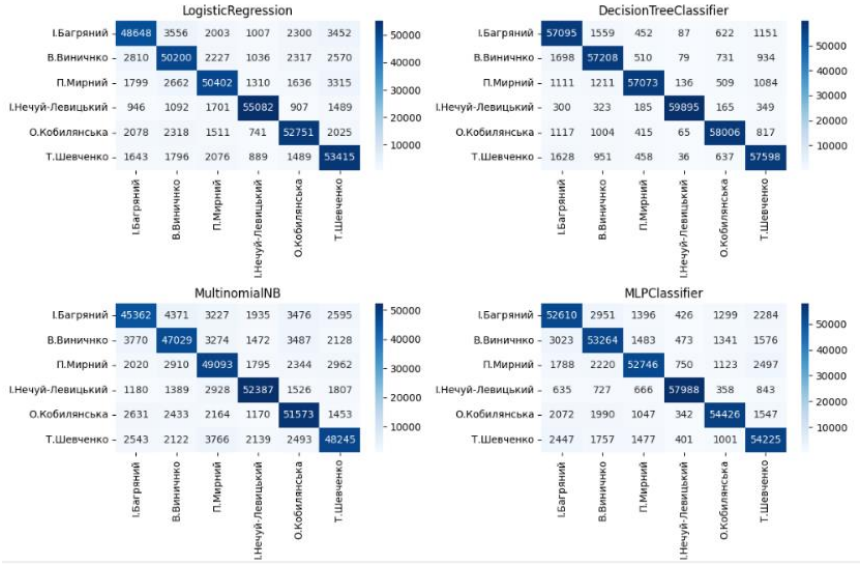


Рисунок 4.3.7 – Матриця похибки моделей TF-IDF та 1-грами з 2-грамами

4.4. Дослідження ефективності глибоких моделей машинного навчання

В рамках даного підрозділу було проведено аналіз ефективності двох моделей глибокого машинного навчання а саме на базі архітектури LSTM та GRU

Архітектура розроблених моделей :

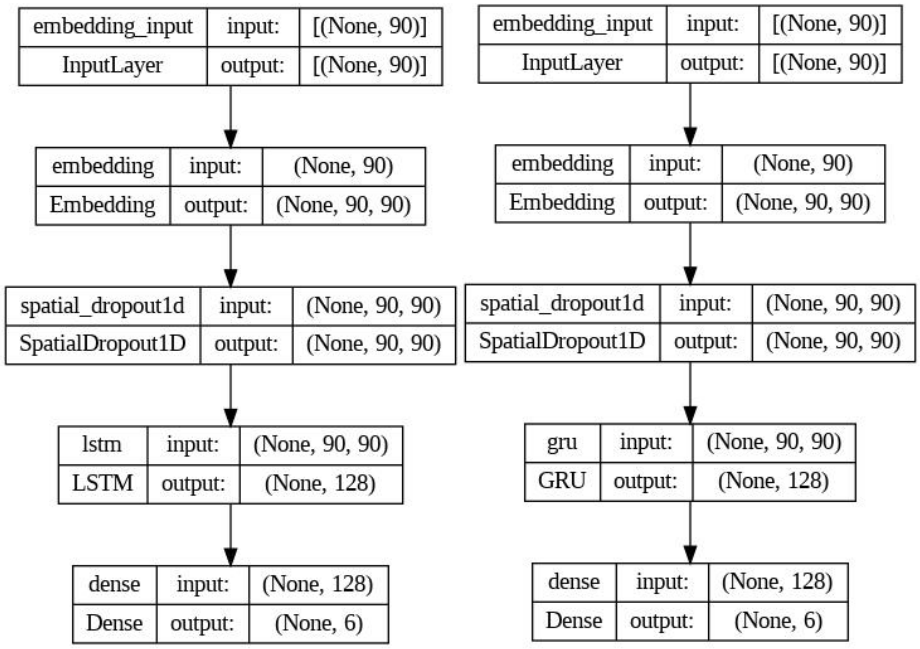


Рисунок 4.4.1 – Архітектура глибоких моделей LSTM та GRU

Моделі було реалізовано з використанням бібліотеки Keras вони мають схожу архітектуру кожна з них складається шару моделі та допоміжних шарів:

Допоміжні шари:

- Embedding: цей шар використовується для перетворення вхідного тексту в числову послідовність за допомогою векторного вбудовування в даному випадку кожне вбудовування має розмірність 90 слів.
- SpatialDropout1D: шар випадково вимикає вихідні функції нейронів Це допомагає уникнути перенавчання моделі .

- Dense: використовує функцію активації для вирішення задачі класифікаціїв даному випадку на 6 класів.

Шари моделі:

- LSTM шар використовує LSTM нейрони для аналізу.
- GRU шар використовує GRU нейрони для аналізу.

Процес навчання моделей:

Відні дані: Корпус авторів української літератури сформований в розділі 4.2

Час навчання 10 епох

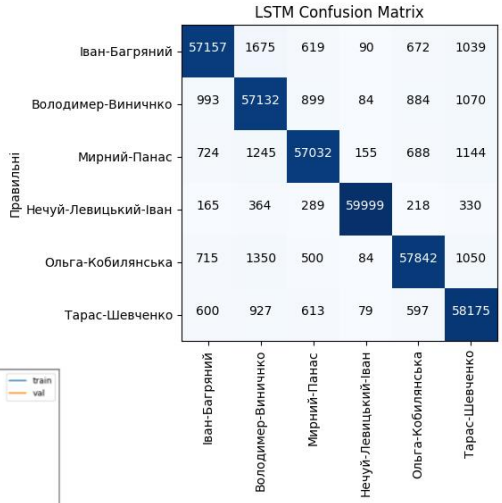
```
Epoch 1/10
3347/3347 [==============================] - 1014s 303ms/step - loss: 0.6286 - accuracy: 0.7707 - val_loss: 0.4265 - val_accuracy: 0.8446
Epoch 2/10
3347/3347 [==============================] - 962s 288ms/step - loss: 0.3779 - accuracy: 0.8598 - val_loss: 0.3433 - val_accuracy: 0.8723
Epoch 3/10
3347/3347 [==============================] - 952s 284ms/step - loss: 0.3147 - accuracy: 0.8812 - val_loss: 0.2956 - val_accuracy: 0.8887
Epoch 4/10
3347/3347 [==============================] - 962s 287ms/step - loss: 0.2714 - accuracy: 0.8969 - val_loss: 0.2574 - val_accuracy: 0.9028
Epoch 5/10
3347/3347 [==============================] - 958s 286ms/step - loss: 0.2380 - accuracy: 0.9092 - val_loss: 0.2274 - val_accuracy: 0.9142
Epoch 6/10
3347/3347 [==============================] - 944s 282ms/step - loss: 0.2122 - accuracy: 0.9188 - val_loss: 0.2026 - val_accuracy: 0.9232
Epoch 7/10
3347/3347 [==============================] - 939s 280ms/step - loss: 0.1918 - accuracy: 0.9264 - val_loss: 0.1835 - val_accuracy: 0.9309
Epoch 8/10
3347/3347 [==============================] - 934s 279ms/step - loss: 0.1755 - accuracy: 0.9323 - val_loss: 0.1681 - val_accuracy: 0.9367
Epoch 9/10
3347/3347 [==============================] - 934s 279ms/step - loss: 0.1625 - accuracy: 0.9369 - val_loss: 0.1546 - val_accuracy: 0.9416
Epoch 10/10
3347/3347 [==============================] - 940s 281ms/step - loss: 0.1521 - accuracy: 0.9408 - val_loss: 0.1430 - val_accuracy: 0.9459
```

Рисунок 4.4.2 – Процес навчання LSTM

```
Epoch 1/10
3347/3347 [==============================] - 882s 261ms/step - loss: 0.6177 - accuracy: 0.7738 - val_loss: 0.4087 - val_accuracy: 0.8507
Epoch 2/10
3347/3347 [==============================] - 812s 242ms/step - loss: 0.3638 - accuracy: 0.8659 - val_loss: 0.3310 - val_accuracy: 0.8772
Epoch 3/10
3347/3347 [==============================] - 810s 242ms/step - loss: 0.3051 - accuracy: 0.8858 - val_loss: 0.2832 - val_accuracy: 0.8948
Epoch 4/10
3347/3347 [==============================] - 807s 241ms/step - loss: 0.2623 - accuracy: 0.9014 - val_loss: 0.2458 - val_accuracy: 0.9083
Epoch 5/10
3347/3347 [==============================] - 816s 244ms/step - loss: 0.2303 - accuracy: 0.9129 - val_loss: 0.2174 - val_accuracy: 0.9180
Epoch 6/10
3347/3347 [==============================] - 814s 243ms/step - loss: 0.2065 - accuracy: 0.9211 - val_loss: 0.1957 - val_accuracy: 0.9268
Epoch 7/10
3347/3347 [==============================] - 806s 241ms/step - loss: 0.1874 - accuracy: 0.9283 - val_loss: 0.1762 - val_accuracy: 0.9333
Epoch 8/10
3347/3347 [==============================] - 802s 240ms/step - loss: 0.1722 - accuracy: 0.9336 - val_loss: 0.1634 - val_accuracy: 0.9382
Epoch 9/10
3347/3347 [==============================] - 799s 239ms/step - loss: 0.1615 - accuracy: 0.9376 - val_loss: 0.1511 - val_accuracy: 0.9430
Epoch 10/10
3347/3347 [==============================] - 794s 237ms/step - loss: 0.1522 - accuracy: 0.9410 - val_loss: 0.1426 - val_accuracy: 0.9463
```

Рисунок 4.4.3 – Процес навчання GRU

	precision	recall	f1-score	support
Іван-Багряний	0.95	0.93	0.94	61252
Володимер-Виничнко	0.91	0.94	0.92	61062
Мирний-Панас	0.95	0.94	0.94	60988
Нечуй-Левицький-Іван	0.99	0.98	0.98	61365
Ольга-Кобилянська	0.95	0.94	0.94	61541
Тарас-Шевченко	0.93	0.95	0.94	60991
accuracy			0.95	367199
macro avg	0.95	0.95	0.95	367199
weighted avg	0.95	0.95	0.95	367199



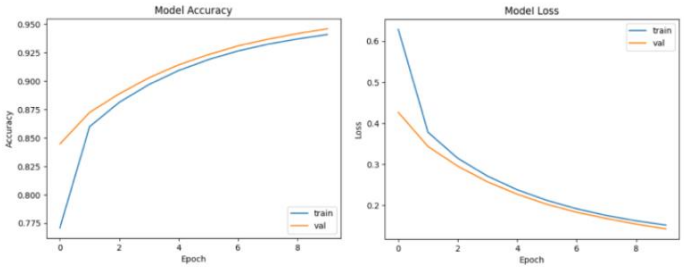
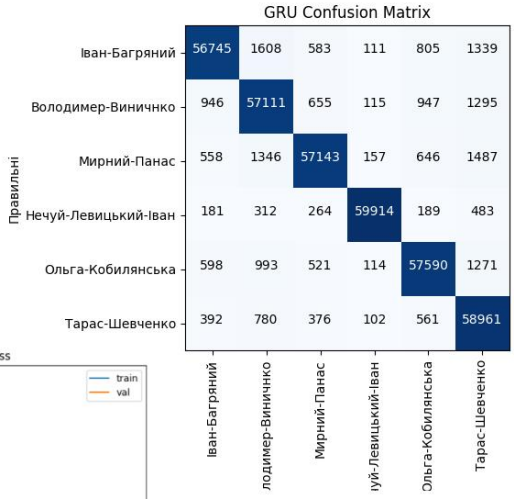


Рисунок 4.4.3 – Результати навчання LSTM

	precision	recall	f1-score	support
Іван-Багряний	0.95	0.93	0.94	61191
Володимер-Виничнко	0.92	0.94	0.93	61069
Мирний-Панас	0.96	0.93	0.95	61337
Нечуй-Левицький-Іван	0.99	0.98	0.98	61343
Ольга-Кобилянська	0.95	0.94	0.95	61087
Тарас-Шевченко	0.91	0.96	0.94	61172
accuracy			0.95	367199
macro avg	0.95	0.95	0.95	367199
weighted avg	0.95	0.95	0.95	367199



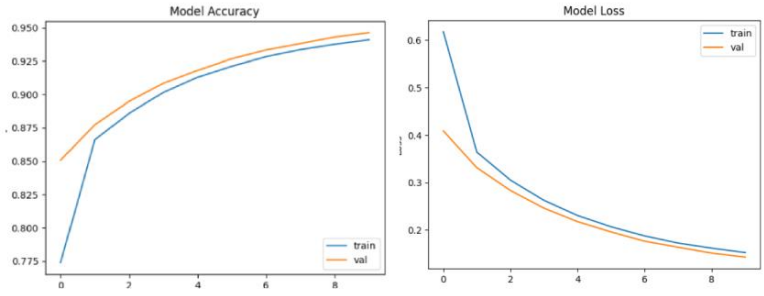


Рисунок 4.4.4 – Результати навчання GRU

4.5. Висновки до розділу 4 Обчислювальний експеримент

У результаті проведення обчислювального експерименту було протестовано чотири популярних моделі класичного навчання для багато класової класифікації а також проведено роботу з моделями глибокого навчання шляхом створення і перевірки 2 моделей типу LSTM та GRU.

Всі дослідження проводились з різними типами вбудовування даних. З метою віднайти найкращий. Під час дослідження було використано методи Мішок слів та TF-IDF з двома типами n-грам: 1-грами та комбінація 1-грам і 2-грам. Навчання моделей проводилось на створеному. корпусі даних українських авторів, який має в собі. збірку творів 6 українських авторів. Дослідження показали, що використання як класичних таких глибоких моделей може дати достатньо хороший результат.. Усі дані моделі можна використовувати для вирішення задачі ідентифікації авторства. Якщо дивитись на класичні моделі, то найкраще себе показали моделі даній яких пройшли через TF-IDF В середньому такі моделі мають на 2% вищу точність, ніж аналогічні їм моделі але кодовані мішком слів. А використання комбінацій n-грам разом з TF-IDF допомогла підвищити точність від 1-4%. Але у випадку використання мішка слів на цьому наборі даних моделі показали навіть меншу точність вона може знизитися від 2% до більше 10%.

Найкращі результати показали наступні моделі:Decision Tree – 95.80% точності (Мішок слів + 1-грами) Multi-layer Perceptron 88.58% точності (TF-IDF + комбінація n-грам) Що стосується моделей глибокого навчання, вони показали доволі схожий результат. За однаковий період свого навчання їхня точність коливаєтеся в районі 94 %. GRU 94.63% LSTM 94.59% . Одна епоха навчання GRU йшла в середньому 800 секунд в той час як Одна епоха навчання LSTM в середньому 900 секунд. GRU показала себе ефективнішою.

ВИСНОВКИ

У першому розділі було дано визначення поняття ідентифікації авторства тексту також розглянуто можливі сфери її застосування, пов’язані з цим проблеми та актуальність самого процесу будо пропрацювано та проаналізовано деякі цікаві дослідження в сфері ідентифікації авторства оглянуло можливі готові рішення та досліджено основні підходи для ідентифікації авторства тексту а саме: підхід використанням статистичних методів, підхід з використанням експертного аналізу та підхід використанням машинного навчання під час процесу дослідження було виявлено сильні та слабкі сторони кожного з підходів. У другому розділі було розглянуто повний алгоритм вирішення задачі в сфері обробки природної мови. Будо розібрано етап підготовки текстових даних, етап вбудовування даних, етап підготовки та навчання моделей та етап оцінки результатів. Під час розбору цих етапів було розглянуто такі поняття як нормалізація, тонізація, лематизація,стоп слова. Розібрано поняття машинного навчання його основні типи та моделі які можна застосувати в контексті вирішення задачі ідентифікації авторства тексту.

У третьому розділі було розроблено програмний додаток для ідентифікації авторства тексту, який базується на результатах досліджень отриманих в процесі виконання роботи. Для своєї роботи додаток використовує наступні технології: Python, Keras , simplemma , Pandas , Streamlit та langdetect. У четвертому розділі було проведено обчислювальний експеримент ефективності класичних та глибоких моделей машинного навчання для вирішення задачі ідентифікації авторства тексту таких як Logistic Regression, Decision Tree, Multinominal NB, Multi-layer Perceptron з класичних і GRU т LSTM з глибоких. Під час оцього експерименту вдалося досягти високої точності ідентифікації авторства яка знаходиться в районі 94-95%

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Ying Zhao, Justin Zobel Searching with Style: Authorship Attribution in Classic Literature 2007. № 148. С. 89-111.
2. Yülüce, İ., Dalkılıç, F. (2022). Author Identification with Machine Learning Algorithms. International Journal of Multidisciplinary Studies and Innovative Technologies, 6(1): 45-50
3. М. І. Лупей Визначення авторської належності українськомовного тексту за допомогою нейросистеми для ідентифікації авторства
4. Подшиваленко Б. О. Застосування методів статистичного аналізу для розв’язання задачі ідентифікації текстів.
5. Shriya TP Gupta, Jajati Keshari Sahoo, Rajendra Kumar Roul: Authorship Identification using Recurrent Neural Networks 2019.
6. Статистичний аналіз. Електронний ресурс: https://stud.com.ua/49878/marketing/statistichniy_analiz 2016
7. Що таке машинне навчання. Електронний ресурс: https://www.ibm.com/topics/machine-learning 2020
8. Словник NLP. Електронний ресурс: https://medium.com/stinopys/%D1%81%D0%BB%D0%BE%D0%B2%D0%BD%D0%B8%D0%BA-nlp-b0fab1027551 2021
9. Моделі машинного навчання. Електронний ресурс: https://learn.microsoft.com/en-us/windows/ai/windows-ml/what-is-a-machine-learning-model 2023
10. Класичне та глибоке навчання. Електронний ресурс: https://lamiae-hana.medium.com/classical-ml-vs-deep-learning-f8e28a52132d 2020

11. Scikit-learn User Guide Електронний ресурс: https://scikit-learn.org/stable/user_guide.html
12. LSTM проти GRU у рекурентній нейронній мережі: порівняльне дослідження Електронний ресурс: https://analyticsindiamag-com.translate.goog/lstm-vs-gru-in-recurrent-neural-network-a-comparative-study/?_x_tr_sl=en&_x_tr_tl=uk&_x_tr_hl=ru&_x_tr_pto=wapp 2023

ДОДАТОК А

Код додатку LAV:

Файл lav.py

```
class Program:
    def __init__(self, gru_model_file, lstm_model_file, stop_words_file, tokenizer_file,
encoder_file):
        self.gru_model_file = gru_model_file
        self.lstm_model_file = lstm_model_file
        self.stop_words_file = stop_words_file
        self.tokenizer_file = tokenizer_file
        self.encoder_file = encoder_file
        self.gru_textClassifier = TextClassifier(model_file=self.gru_model_file,
tokenizer_file=self.tokenizer_file,
                                                 encoder_file=self.encoder_file,
stop_words_file=self.stop_words_file)
        self.lstm_textClassifier = TextClassifier(model_file=self.lstm_model_file,
tokenizer_file=self.tokenizer_file,
                                                  encoder_file=self.encoder_file,
stop_words_file=self.stop_words_file)

    @staticmethod
    def templeate_style():
        hide_streamlit_style = """
            <style>
            #MainMenu {visibility: hidden;}
            footer {visibility: hidden;}
            header {visibility: hidden;}
            div.stButton > button:first-child {
                display block;
                width: 100%;
                background-color: rgb(204, 49, 49);
                color: #fff;
```

```
            }
            </style>
            """
        st.markdown(hide_streamlit_style, unsafe_allow_html=True)

    @staticmethod
    def is_valid(text):
        return detect(text) == 'en' or detect(text) == 'it'

    def main(self):
        self.templeate_style()

        models = {'GRU модель': self.gru_textClassifier, 'LSTM модель':
self.lstm_textClassifier}
        st.title('Розпізнавання автора тексту')
        text_input = st.text_area('Введіть текст українською мовою:', height=100)
        texts = text_input.split('\n')
        model_choice = st.selectbox('Виберіть модель:', list(models.keys()))

        if st.button('Розпізнати'):
            new_texts = []
            indices = []
            for index, element in enumerate(texts):
                if len(element.split(' ')) < 3:
                    st.error(f'Помилка! рядок {index + 1} [Введіть принаймні 3 слова]')
                elif self.is_valid(element):
                    st.error(f'Помилка! рядок {index + 1} [Текст повинен бути написаний
українською мовою]')
                else:
                    new_texts.append(element)
                    indices.append(index + 1)
```

```
            texts = new_texts
            model = models[model_choice]
            prediction = model.predict(texts)
            authors = [item for sublist in prediction for item in sublist]
            result = pd.DataFrame({'Автор': authors, 'Його текст': texts}, index=indices)
            st.write(result)

        if st.button('Доступні автори'):
            result = pd.DataFrame({'Автор': self.gru_textClassifier.get_class_names()},
                                  index=range(1, len(self.gru_textClassifier.get_class_names()) + 1))
            st.write(result)


if __name__ == '__main__':
    gru_model_file = './files/GRU.h5'
    lstm_model_file = './files/LSTM.h5'
    stop_words_file = './files/stopwords_ua_list.txt'
    tokenizer_file = './files/tokenizer.pickle'
    encoder_file = './files/encoder.pickle'
    program = Program(gru_model_file, lstm_model_file, stop_words_file, tokenizer_file,
encoder_file)
    program.main()
```