DOI 10.34185/1562-9945-5-154-2024-13
УДК 004.912:519.25

Д.І. Павлюк, О.Г. Байбуз

ОГЛЯД МЕТОДІВ СЕМАНТИЧНОЇ КЛАСИФІКАЦІЇ ТЕКСТУ

Аннотація: У статті проведено аналіз методів класифікації тексту, таких як наївний байєс, логістична регресія, метод опорних векторів (SVM) з акцентом на сучасні методи глибокого навчання, включаючи штучні нейронні мережі (ANN), згорткові нейронні мережі (CNN), рекурентні нейронні мережі (RNN) та трансформери. Для оцінки використовується датасет відгуків про фільми IMDB. Дослідження порівнює ефективність цих методів у класифікації тексту за тональністю, враховуючи точність та обчислювальні ресурси. Мета роботи - визначити найкращий підхід для автоматизованої класифікації тексту та надати рекомендації для подальших досліджень.
Ключові слова: Класифікація тексту, Наївний Байєс, Логістична регресія, Метод опорних векторів (SVM), Штучні нейронні мережі (АНМ), Згорткові нейронні мережі (ЗНМ), Рекурентні нейронні мережі (РНМ), Трансформери, Аналіз тональності, обробка природної мови).

Постановка проблеми. Класифікація тексту є важливою складовою задач, таких як аналіз настроїв, фільтрація спаму та організація документів. Традиційні методи, як наївний байєс, логістична регресія та метод опорних векторів (SVM), швидкі та ефективні, але їх продуктивність залежить від якості ознак. Методи машинного навчання, такі як дерева рішень та градієнтний бустинг, враховують складні закономірності, але потребують ретельного підбору параметрів.

Глибоке навчання, включаючи штучні нейронні мережі (ANN), згорткові нейронні мережі (CNN) та рекурентні нейронні мережі (RNN), автоматично виявляють складні патерни в текстових даних. Передові методи NLP, такі як трансформери, забезпечують високу точність і продуктивність у класифікації тексту. При цьому важливо зазначити, що сучасні підходи вимагають велику кількість обчислювальних ресурсів для донавчання та не завжди є оптимальним рішенням лоя певних задач.

Дослідження порівнює ефективність цих методів, визначаючи найкращий підхід для класифікації тексту. Використання сучасних методів глибокого навчання показує переваги в точності та продуктивності, що сприяє покращенню систем обробки текстів та надає напрямки для майбутніх досліджень.

Мета дослідження. Метою роботи є огляд методів автоматизованої класифікації тексту за ознакою тональності. Необхідно визначили точність передбачення тональності для різних моделей та проаналізувати отримані результати. Під час дослідження були розглянуті такі методи класифікації як Наївний Байес, Support Vector Machines, Логістична Регресія та різні

ISSN 1562-9945 (Print)
ISSN 2707-7977 (Online)

імплементації нейронних мереж. Такі методи як транформери тексту не розглялися через високі системні вимоги (не вдалося провести навчання BERT та ELMo на локальному середовищі та у хмарному середовищі Google Colab).

Дані для дослідження. Для навчальної та тестувальної вибірки було використано датасет “IMDB Dataset of 50K Movie Reviews”[2]. Він включає у себе два стовпці:

- *Review* (огляд) - огляд фільму, за яким необхідно необхідно передбачити враження від перегляду. Представляє з себе текстові дані
- *Sentiment* (настрій, враження) - емоційне забарвлення відгуку, яке необхідно передбачити. Представляє з себе поле, що може мати два допустимі значення: *positive* та *negative.*

Таблиця 1

Перші 4 записи у датасеті “IMDB Dataset of 50K Movie Reviews”

review	sentiment
One of the other reviewers has mentioned that after watching just 1 Oz episode you'll be hooked. The...	positive
A wonderful little production. The filming technique is very unassuming- very old-time-B...	positive
I thought this was a wonderful way to spend time on a too hot summer weekend, sitting in the air con...	positive
Basically there's a family where a little boy (Jake) thinks there's a zombie in his closet & his par...	negative

Розробка програмного забезпечення. Було використано мову програмування Python для імплементації роботи моделей для машинного навчання.

Для візуалізації результатів дослідження роботу було виконано у формі документу Jupyter Notebook. Формат дозволяє суміщати HTML-розмітку на код на Python. Звіти з візуалізацією результатів та програмним кодом можна знайти за посиланням на GitHub-репозиторій[1].

Завантаження даних. Імпортуємо необхідні бібліотеки. Наступним кроком слід завантажити zip-архів с csv-датасетом за спеціальним посиланням та конвертувати у датафрейм (pandas.DataFrame) для подальшої роботи з даними[1].

Підготовка та аналіз датасету.

HTML-розмітка

Якщо поглянемо на табл. 1, то можемо побачити HTML-теги, а саме

, у другому огляді фільму. HTML-розмітка не відіграє жодної ролі у семантичному аналізі, тому для моделей ця інформація буде шумовою. У такому випадку, видалимо HTML-теги з оглядів.

Регістр

Технічно між словом з маленької літери та словом з великої літери немає різниці з точки зору семантичного забарвлення, тому має сенс привести текст до одного регістру. Приведемо текст до нижнього регістру.

Пунктуація

Існують моделі, які здатні враховувати пунктуацію[3], але більш прості векторизатори, які ми розглянемо з початку. Враховуючи цей факт, можемо видалити пунктуацію з вхідних даних.

Стоп-слова

Для моделей, які не здатні розпізнавати граматичні зв'язки у реченнях, бажано видаляти так звані стоп-слова, які можуть вносити шум у вхідні дані. Часто у тексті зустрічаються слова без лексичного значення, такі як, наприклад, артиклі ("the", "a", "an"), сполучники ("and", "but", "or") або прийменники ("of", "to", "in"), тому їх видалення може сприяти більш високій точності класифікації[4].

Стеммінг та лемматизація

Як правило, стеммінг зводить слова до їх базової форми, знімаючи суфікси відповідно до набору правил, тоді як лемматизація розглядає граматичний контекст, щоб знайти словникову форму (лему) слова. Таким чином, два методи дозволяють захопити різні форми одного і того ж слова: стемінг перетворить всі слова у "run"; наприклад, при застосуванні цієї техніки можна отримати "run" від "running" або "ran" від "runs" при використанні лематизації. Тим не менш, стеммінг може бути швидшим, ніж лематизація, хоча це може дати менш точні результати. У дослідженні розглянемо лематизацію, хоча у випадку, коли вхідних даних більше, стеммінг може бути більш придатним з огляду на обчислювальну ефективність[5].

Бінарні ознаки

Поле *Sentiment* приймає два значення, тому можемо трансформувати його у булеве, що й було зроблено.

Таблиця 2

Перші 4 записи у оновленому датасеті

review	sentiment
one reviewer mentioned watching 1 oz episode y...	1
wonderful little production filming technique ...	1
thought wonderful way spend time hot summer we...	1
basically there family little boy jake think t...	0

Аналіз датасету

Проведемо аналіз датасету за ознакою *Sentiment.*

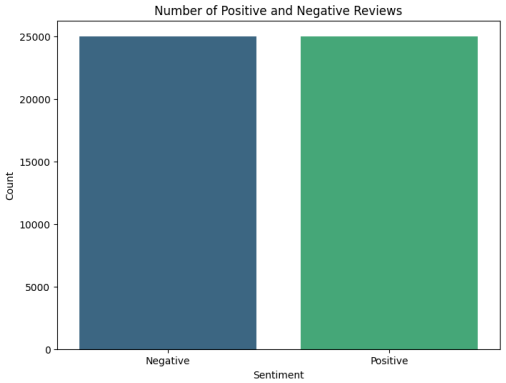

Рисунок 1 - Розподіл позитивних та негативних відгуків

ISSN 1562-9945 (Print)
ISSN 2707-7977 (Online)

Всього у датасеті 50000 записів. Як можна побачити на Рисунку 1, позитивні та негативні відгуки розподілені рівномірно, інаше кажучи, маємо по 25000 позитивних та негативних рецензій, тобто датасет є збалансованим.

Векторизація. Алгоритми машинного навчання не можуть працювати безпосередньо з сирим текстом. Проблема роботи з текстом у контексті обробки природної мови полягає в тому, що він є безладним з точки зору машини, і такі методи, як алгоритми машинного навчання, віддають перевагу добре визначеним вхідним даним фіксованої довжини (як, наприклад, числові вектори). Процес приведення тексту у зрозумілий машині вигляд називається *виведенням ознак* (англ. feature extraction). Допомогти з цією задачею можуть різноманітні алгоритми векторизації та більш передові трансформери тексту.

Bag-of-Words (BoW)

Bag-of-Words - це метод представлення тексту, який перетворює документ у вектор фіксованої довжини. У цьому методі текст розбивається на окремі слова, і кожне слово розглядається незалежно від інших. Вектор представляє кількість входжень кожного слова у документі, ігноруючи граматичні структури та порядок слів[6].

TF-IDF (Term Frequency-Inverse Document Frequency)

TF-IDF - це метод зважування слів у документі, який використовується для оцінки важливості слова в контексті всього корпусу документів. TF (Term Frequency) вимірює частоту слова у документі, а IDF (Inverse Document Frequency) оцінює рідкість слова в наборі документів. Зважування TF-IDF зменшує вплив часто зустрічаються слів, підкреслюючи унікальні слова, які мають більше значення для конкретного документа[6].

Word2Vec

Word2Vec – це алгоритм для створення векторних уявлень слів, розроблений дослідниками Google у 2013 році. Він використовує нейронні мережі для навчання векторів слів на великих текстових корпусах, де слова з подібними контекстами мають схожі вектори. Існують дві основні моделі: Continuous Bag of Words (CBOW), яка передбачає слово на основі його контексту, та Skip-Gram, яка передбачає контекстні слова на основі цільового слова. Word2Vec значно покращив обробку природної мови (NLP), дозволяючи розпізнавати семантичні та синтаксичні зв'язки між словами[7]. Існує велика кількість попередньо тренованих моделей, які можуть збільшити точність прогнозування[12].

Огляд статистичних методів. Для роботи з моделями текст попередньо обробляється (з використанням вищевказаних кроків; на останньому кроці надамо перевагу лемматизації, так як обсяг даних є відносно невеликим та різниця у часі у порівнянні зі стеммінгом не буде відчутною. Для роботи з моделями Наївний Байес, SCM та Логістичної Регресії вхідні дані векторизуємо з використанням TF-IDF. Також трансформуємо текст з використанням Word2Vec для SVM та Логістичної Регресії. Мультиноміальний наївний Байес не призначений для роботи з від’ємними значеннями, а нормалізація даних може шум у тестову вибірку. Також застосуємо Word2Vec, що був попередньо тренований на датасеті Google News (близько 100 мільярдів слів)[12]. Використання попередньо може вирішити проблему коли у тренувальній немає відповідного токена для тестової вибірки та збільшити точність прогнозування загалом за рахунок більш точних ваг для токенів.

Наївний Байес

Наївний Байес - група алгоритмів класифікації даних за допомогою машинного навчання заснованих на теоремі Байеса. Включає у себе різні варіації алгоритми для класифікації тексту. Мультиноміальний Байес (Multinomial Naive Bayes реалізує алгоритм Наївний Байес для мультиноміально розподілених даних. Враховуючи те, що взятий у роботу датасет включає у себе дві групи по 25000 зразків, можемо вважати дані збалансованими, тому розглянемо варіант використання Мультиноміального Наївного Байеса.

Алгоритм вирізняється простотою реалізації та високою ефективністю обчислень. Хоча, враховуючи факт, що алгоритм базується на припущенні про незалежність ознак, він є не зовсім підходящим для випадків коли є залежні ознаки. Висока ефективність обчислень дає можливість працювати з великими обсягами даних.

Розглянемо параметри для класифікатора:

***alpha*:** Цей параметр задає величину згладжування Лапласа. Додає невелике фіктивне число до кожного підрахунку ознак у кожному класі. Допомагає запобігти перенавчанню, особливо коли кількість ознак велика, а кількість навчальних прикладів - мала.

Спробуємо перебрати значення: 0.1, 0.5, 1.0, 2.0, 5.0 (стандартне значення: 1.0).

fit_prior: Цей параметр вказує, чи потрібно обчислювати апріорні ймовірності класів з навчальних даних.

Значення параметра:

True: Рекомендується, так як модель автоматично підлаштовується під розподіл класів у даних.

False: Може призвести до зміщення в бік поширеного класу.

Для збалансованих даних не відіграє великого значення.

Очікується, що для обох параметрів найкращі результати покажуть параметри за замовчуванням зважаючи на те, що датасет є збалансованим[8].

Розмір навчальної вибірки: Враховуючи, високу ефективність обчислень для даного алгоритму можемо використовувати досить велику навчальну вибірку. Розглянемо такі розбиття (0.95 означає, що 95% відводиться на навчальну вибірку, а решта - на тестову): 0.95, 0.9, 0.8, 0.5, 0.2, 0.1, 0.05, 0.01, 0.005, 0.001, 0.0005, 0.0001.

Результати перевірки:

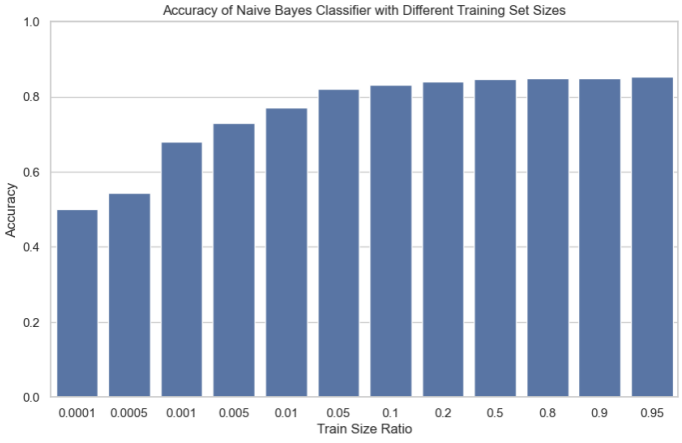

Рисунок 2 - Результати класифікації відгуків для алгоритму Наївний Байес

ISSN 1562-9945 (Print)
ISSN 2707-7977 (Online)

Як можемо побачити на Рисунку 2, починаючи з 5000-10000 зразків навчальної вибірки швидкість росту точності в залежності від розміру тренувальної вибірки зменшується і графік точності виходить на своєрідне плато. При значеннях навчальної вибірки менше 25 зразків точність дорівнює близько 50%, що фактично є вгадуванням наосліп.

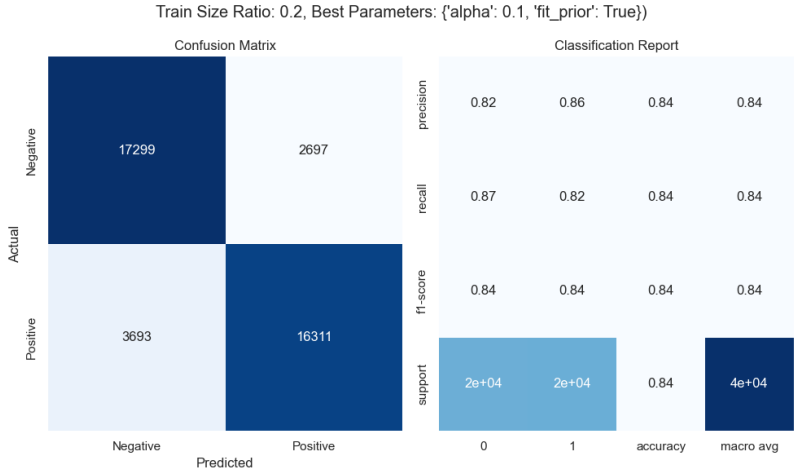

Рисунок 3 - Більш детальний звіт для алгоритму Наївний Байес

Як можемо побачити на Рисунку 3, точність передбачення дорівнює близько 84%. Найкращі результати показали параметри за замовчуванням (alpha: 0.1, fit_prior: true).

Продуктивність: Підбір параметрів, який включав у себе кросс-валідацію з різними наборами параметрів на вказаних вибірках зайняв близько 5 хвилин. Враховуючи відносно високу точність класифікації починаючи з тренувальної вибірки у 5000-10000 тисяч зразків, у реальних умовах застосування час навчання моделі є несуттєвим, особливо у порівнянні з багатьма іншими моделями.

Попередні висновки**:** При навчанні на вибірці з 10000 зразків точність склала 84%. При використанні 47500 зразків точність скалала 85%, що не є надто суттєвим ростом. Існує не надто багато способів для налаштування алгоритму, хоча він дозволяє запобігти перенавчанню та працювати з незбалансованими даними при використанні налаштовуваних параметрів. В цілому, алгоритм придатний для класифікації тексту, особливо якщо навчальна вибірка має великий обсяг, а обчислювальні потужності обмежені.

Support Vector Machines

Support Vector Machines (SVM) використовується для класифікації текстових даних шляхом знаходження оптимальної гіперплощини, яка розділяє дані різних класів з максимальним проміжком. SVM може бути застосований для задач класифікації тексту, таких як визначення тональності тексту, завдяки здатності працювати у високорозмірних просторах ознак.

C: Параметр регуляризації, що контролює баланс між максимізацією зазору та мінімізацією помилки класифікації. Низькі значення C роблять модель більш гнучкою (висока регуляризація), тоді як високі значення C зменшують регуляризацію, роблячи модель більш суворою. Моделі з високим значенням цього параметру можуть мати схильність до перенавчання. Приклади значень: [0.01, 0.1, 1, 10, 100].

gamma: Коефіцієнт для ядер, що визначає "радіус впливу" однієї навчальної точки. Актуальний для RBF, поліноміальних та сигмоїдальних ядер. Високі значення gamma призводять до моделі з високим перенавчанням, а низькі значення роблять модель більш гладкою і узагальненою. Приклади значень: ['scale', 'auto', 1, 0.1, 0.01, 0.001, 0.0001].

kernel: Функція ядра визначає спосіб перетворення вхідних даних у більш високий простір ознак. Основні типи ядер:

- 'linear': Лінійне ядро, підходить для лінійно роздільних даних.
- 'poly': Поліноміальне ядро, корисне для складних нелінійних взаємозв'язків.
- 'rbf': Ядро радіальної базисної функції, добре моделює складні нелінійні межі.
- 'sigmoid': Сигмоїдальне ядро, аналог нейронних мереж із сигмоїдальною функцією активації.

Серед недоліків зазвичай згадується вразливість до шумових даних та вибагливість до обчислювальний ресурсів. У випадку роботи з десятками тисяч зразків зразків рекомендується використовувати лінійне ядро[9].

Розмір навчальної вибірки: Враховуючи, вибагливість обчислень для даного алгоритму, маємо певні обмеження по розміру вибірки. Розглянемо такі розбиття: 0.5, 0.1, 0.05, 0.01, 0.005, 0.001, 0.0005.

Результати перевірки:

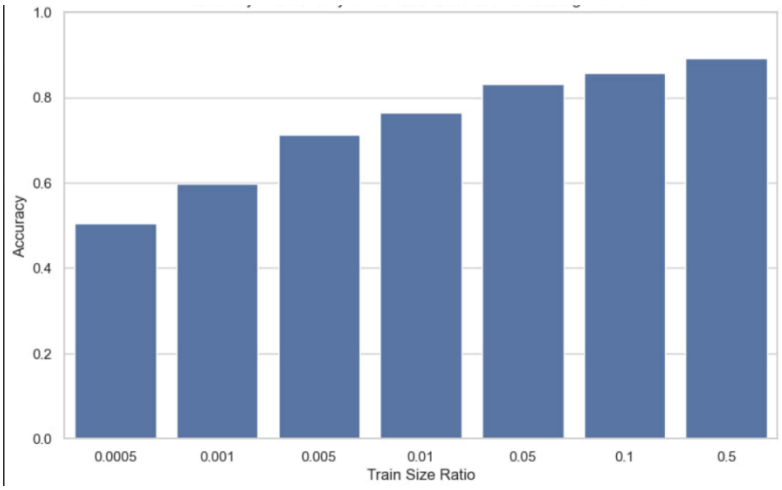

Рисунок 4 - Результати класифікації відгуків для алгоритму SVM

На Рисунку 4 можемо побачити, що точність суттєво збільшується зі збільшенням навчальної вибірки (25 зразків - 50% (фактично, вгадування наосліп), 50 зразків - 60%, 250 зразків - 71%, 500 зразків - 76%, 2500 зразків - 83%, 5000 зразків - 86%, 25000 зразків - 89%).

ISSN 1562-9945 (Print)
ISSN 2707-7977 (Online)

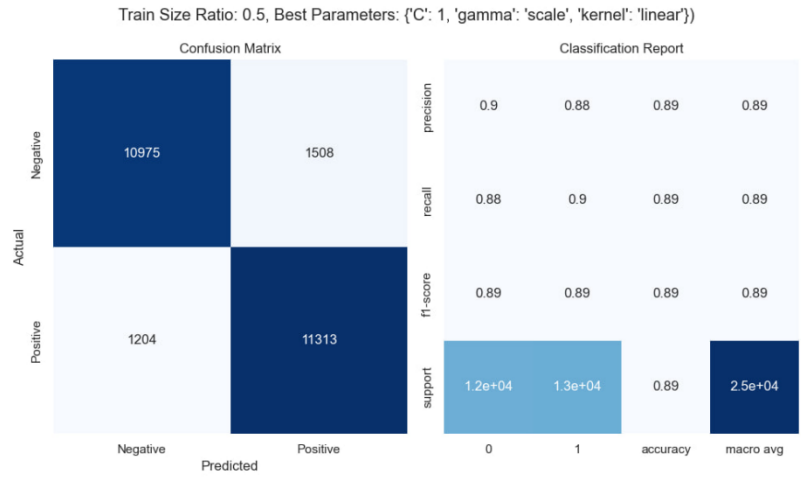

Рисунок 5 - SVM з TF-IDF та оптимальний набір параметрів моделі

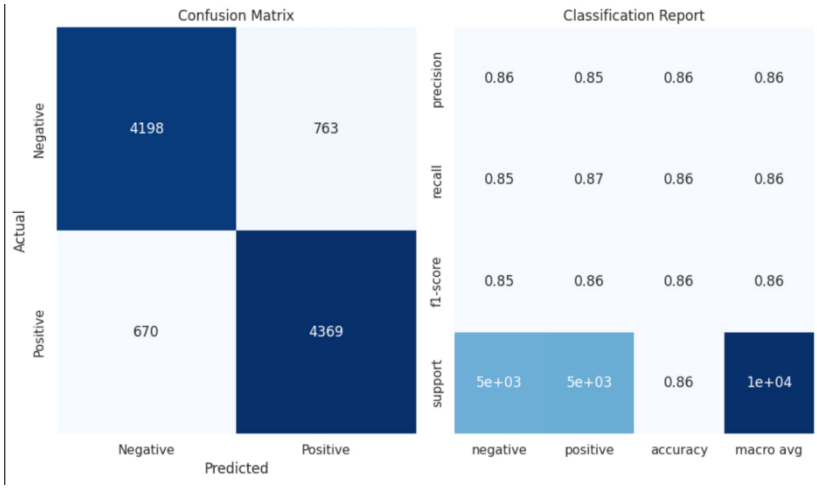

Рисунок 6 - SVM з Word2Vec

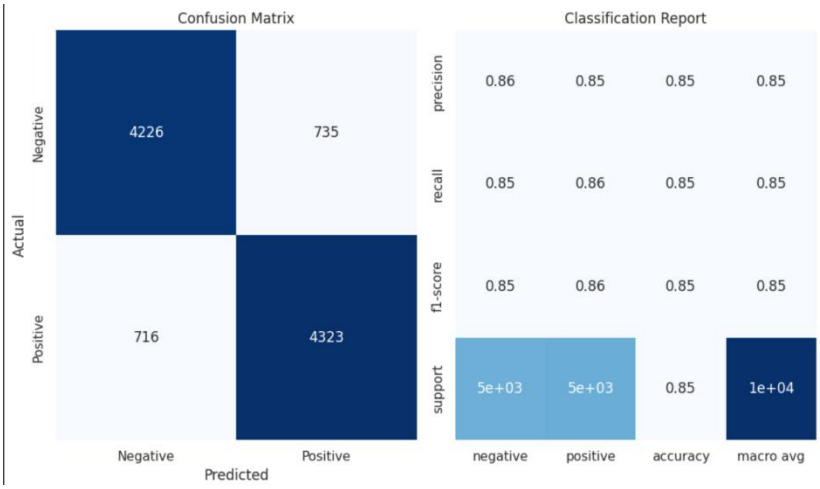

Рисунок 7 - SVM з попередньо натренованим Word2Vec

На Рисунку 5 можемо побачити оптимальний набір параметрів для заданої вибірки. Результати можуть відрізнятися від характеру текстових даних, але загалом рекомендується

використовувати лінійне ядро у випадку великої кількості даних задля економії часу та обчислювальних потужностей.

Висновки: При використанні TF-IDF вдалося досягти точності прогнозування у 89% (Рисунок 5); 86% для Word2Vec (Рисунок 6) та 85% для попередньо натренованого Word2Vec (Рисунок 7); підбір параметрів для навчальних вибірок різного розміру зайняв близько 160 хвилин.

Логістична регресія

Логістична регресія є статистичним методом для аналізу набору даних, у якому одна або кілька незалежних змінних визначають результат. Вона використовується для задач класифікації, де цільова змінна є бінарною (наприклад, позитивна або негативна тональність тексту). Логістична регресія моделює ймовірність того, що конкретний зразок належить до певного класу, використовуючи логістичну функцію (сигмоїдну).

C: параметр регуляризації, який визначає силу регуляризації. Низькі значення C вказують на сильну регуляризацію, тоді як високі значення зменшують регуляризацію. Приклади значень: [0.01, 0.1, 1, 10, 100].

penalty: тип регуляризації, яка застосовується до моделі для запобігання перенавчанню. Приклади значень: ['l1', 'l2', 'elasticnet'(комбінація l1 та l2)[11].

solver: алгоритм оптимізації, який використовується для пошуку оптимальних параметрів моделі. Приклади значень: ['newton-cg', 'lbfgs', 'liblinear', 'sag', 'saga'].

max_iter: максимальна кількість ітерацій, що використовуються алгоритмом для збіжності. Приклади значень: [100, 200, 500].

l1_ratio: параметр для балансування між L1 та L2 регуляризаторів для ElasticNet[10][11].

Розмір навчальної вибірки: Алгоритм є менш витратним ніж SVM[10]. Розглянемо такі розбиття: 0.5, 0.2, 0.1, 0.05, 0.01, 0.005, 0.001, 0.0005, 0.0001.

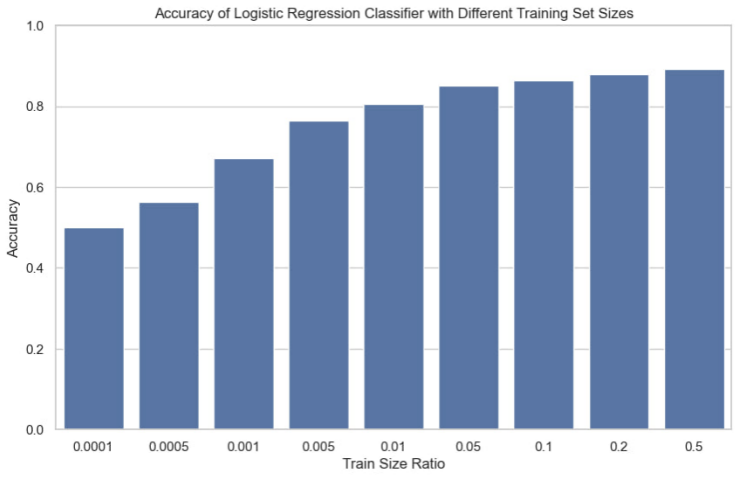

Рисунок 9 - Результати класифікації для Логістичної регресії

На Рисунку 7 можемо побачити, що точність суттєво збільшується зі збільшенням навчальної вибірки (5 зразків - 50%, 25 зразків - 58%, 50 зразків - 67%, 250 зразків - 76%, 500

ISSN 1562-9945 (Print)
ISSN 2707-7977 (Online)

зразків - 81%, 2500 зразків - 85%, 5000 зразків - 86%, 10000 зразків - 88%, 25000 зразків - 89%).

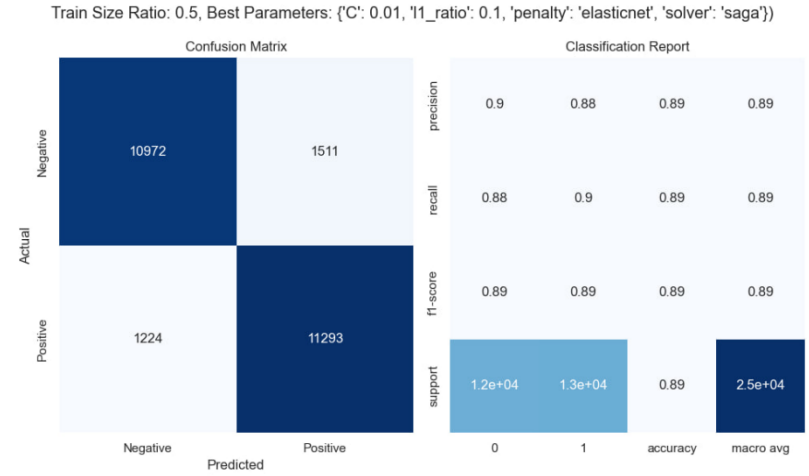

Рисунок 10 - Результати для Логістичної Регресії з ElasticNet регуляризацією

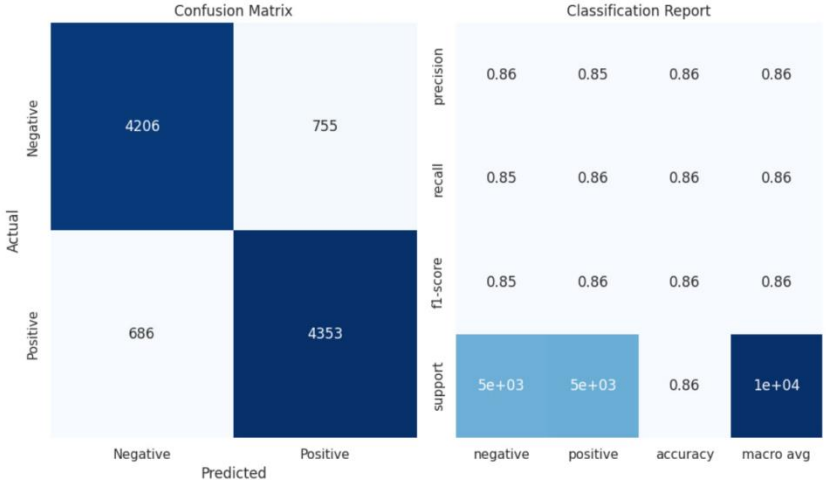

Рисунок 11 - Результати для Логістичної Регресії з Word2Vec

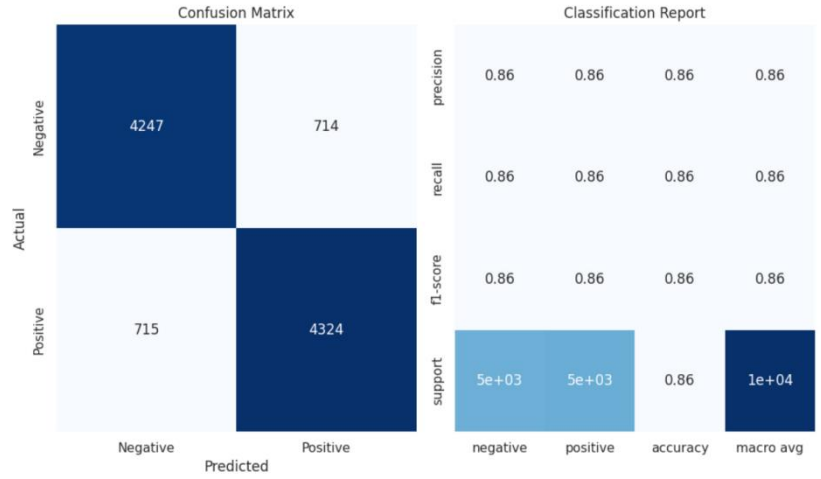

Рисунок 12 - Результати для логістичної регресії з попередньо натренованим Word2Vec

Висновки: При використанні TF-IDF вдалося досягти точності прогнозування у 89% (Рисунок 10); 86% для Word2Vec (Рисунок 11) та 86% для попередньо натренованого Word2Vec (Рисунок 12); підбір параметрів для навчальних вибірок різного розміру зайняв близько 20 хвилин.

Загальні висновки

При класифікації тексту з використанням TF-IDF та моделей Naive Bayes, SVM та Logistic Regression найкраща точність була досягнута з використанням моделей SVM та Logistic Regression (89% при розбиття навпіл на навчальну та тестову вибірки). Модель Naive Bayes продемонструвала теж прийнятні результати, але показника точності вище 84-85% досягти не вдалося. При використанні моделі Word2Vec точність передбачення при задіяних 80% вибірки для навчання склала 86%. Для SVM та Logistic Regression метрики виявилися дуже схожими. Подібні результати були продемонстровані Word2Vec попередньо тренованому на вибірці з Google News (точність для Логістичної Регресії: 86%, для SVM: 85%). Вірогідно, проблема полягає у не стільки у обмеженому об’ємі словника, скільки у необхідності роботи з контекстом та розпізнаванні текстових патернів.

Нейронні мережі. Розглядаємо три типи нейронних мереж для задачі класифікації тексту: штучні нейронні мережі (ANN), згорткові нейронні мережі (CNN) та рекурентні нейронні мережі (RNN) з використанням LSTM. Для кожної моделі ми описали використані параметри та налаштування, пояснили причини їх вибору та провели короткий огляд.

Короткий огляд

Artificial Neural Networks (ANN): ANN складаються з шарів нейронів, кожен з яких з'єднаний з нейронами наступного шару. Вони підходять для базових задач класифікації тексту, але можуть не враховувати порядок слів у тексті, що обмежує їх ефективність у складніших задачах .

Convolutional Neural Networks (CNN)**:** CNN використовують згорткові шари для виділення локальних ознак тексту, таких як фрази або н-грами. Вони добре підходять для задач, де важливі локальні залежності між словами, але можуть не враховувати далекі залежності у тексті .

Recurrent Neural Networks (RNN) з LSTM: RNN з LSTM-шарами дозволяють моделі враховувати послідовність та залежності між словами у тексті. Вони ефективні для задач, де важливий контекст і порядок слів, що робить їх найкращими для обробки послідовних даних.

Підготовка даних

При класифікації тексту з використанням методів глибокого навчання видалення стопслів може негативно вплинути на якість класифікації через втрату граматичних зв’язків у реченнях.

Використаємо токенізатор для роботи з даними. Його принцип роботи полягає у перетворенні слів у числові токени, а тексти, в свою чергу, у вектори з цих чисел. Попередньо текст приводиться до нижнього регістру, видаляються символи пунктуації та спеціальні символи [13].

Крім того, слід додати значення «0» у кінець векторів (або у початок) для того щоб привести усі вектори до однакової довжини (довжини найдовшого вектора)[13].

ISSN 1562-9945 (Print)
ISSN 2707-7977 (Online)

Методика

Artificial Neural Network (ANN):

Модель ANN була побудована зі наступними шарами:

Embedding Layer: Використано шар embedding з розмірністю 256, щоб перетворити текстові дані у вектори фіксованої розмірності. Це дозволяє моделі ефективно працювати з текстом.

Flatten Layer: Було додано шар Flatten для перетворення багатовимірного виходу шару embedding в одновимірний вхід для повнозв'язних шарів.

Dense Layers: Використано два повнозв'язні шари з 128 та 64 нейронами відповідно та функцією активації ReLU для навчання складним нелінійним взаємозв'язкам у даних.

Dropout Layers: Було додано два шари Dropout з параметром 0.5 для запобігання перенавчанню шляхом випадкового вимикання нейронів під час навчання.

Output Layer: Використано вихідний шар з 1 нейроном та функцією активації sigmoid для бінарної класифікації.

Convolutional Neural Network (CNN)

Модель CNN складалася з наступних шарів:

Embedding Layer: Використано шар embedding з розмірністю 128 для перетворення тексту у вектори фіксованої розмірності.

Conv1D Layer: Було додано згортковий шар з 128 фільтрами та розміром фільтра 5 для виділення локальних ознак тексту. Функція активації ReLU забезпечує нелінійність.

GlobalMaxPooling1D Layer: Використано глобальне підсумовування для зменшення розмірності виходу з згорткового шару, вибираючи максимальне значення з кожного фільтра.

Dense Layer: Додано повнозв'язний шар з 64 нейронами та функцією активації ReLU для класифікації на основі виділених ознак.

Output Layer: Використано вихідний шар з 1 нейроном та функцією активації sigmoid для бінарної класифікації.

Recurrent Neural Network (RNN) з LSTM:

Модель RNN з LSTM включала такі шари:

Embedding Layer: Використано шар embedding з розмірністю 256 для перетворення текстових даних у вектори фіксованої розмірності.

LSTM Layer: Було додано LSTM-шар з 256 нейронами для обробки послідовностей та врахування залежностей між елементами тексту.

Dense Layers: Використано два повнозв'язні шари з 128 та 64 нейронами відповідно та функцією активації ReLU для навчання складним нелінійним взаємозв'язкам у даних.

Dropout Layers: Було додано два шари Dropout з параметром 0.5 для запобігання перенавчанню.

Output Layer: Використано вихідний шар з 1 нейроном та функцією активації sigmoid для бінарної класифікації.

Для кожної моделі було проведено навчання та оцінку на тестовій вибірці з наступними налаштуваннями:

Було обрано оптимізатор Adam та функцію втрат binary_crossentropy для швидкої і стабільної конвергенції.

EarlyStopping: Використано callback EarlyStopping з моніторингом показника 'val_accuracy', patience=2 та mode='max' для зупинки навчання, коли покращення точності на валідаційній вибірці зупиняється, щоб уникнути перенавчання.

Результати:

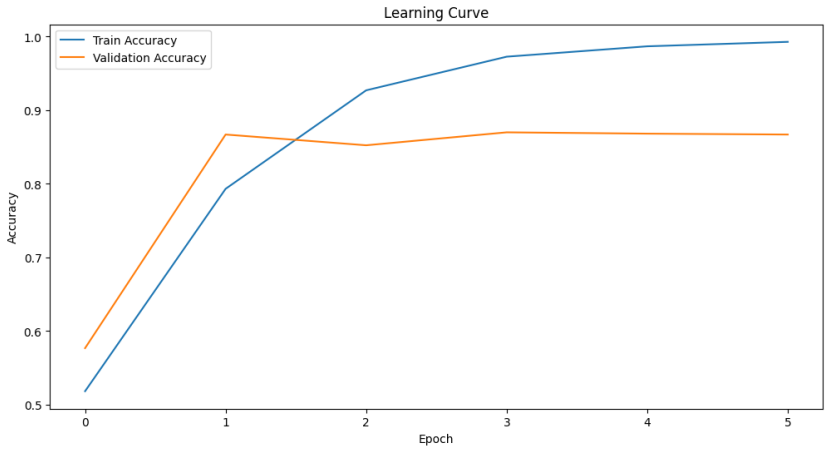

Рисунок 13 - Крива навчання для ANN

Таблиця 3

Звіт для ANN

Class	Precision	Recall	F1-score	Support
0	0.85	0.86	0.85	12500
1	0.86	0.85	0.85	12500
accuracy	0.85	0.85	0.85	12500
macro avg	0.85	0.85	0.85	12500
weighted avg	0.85	0.85	0.85	12500

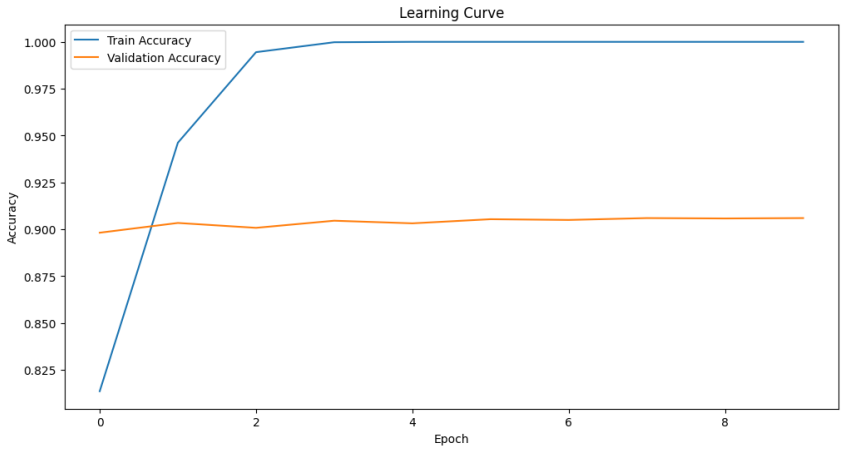

Рисунок 15 - Крива навчання для CNN

ISSN 1562-9945 (Print)
ISSN 2707-7977 (Online)

Таблиця 4

Звіт для CNN

Class	Precision	Recall	F1-score	Support
0	0.90	0.90	0.90	12500
1	0.90	0.90	0.90	12500
accuracy	0.90	0.90	0.90	12500
macro avg	0.90	0.90	0.90	12500
weighted avg	0.90	0.90	0.90	12500

На Рисунок 13 можемо побачити схожу тенденцію зменшення похибки на тренувальних і тестових наборах. Починаючи з 6 епохи модель почала демонструвати ознаки перенавчання, тому на епосі 8 навчання було зупинено. Точність прогнозування склала 90%.

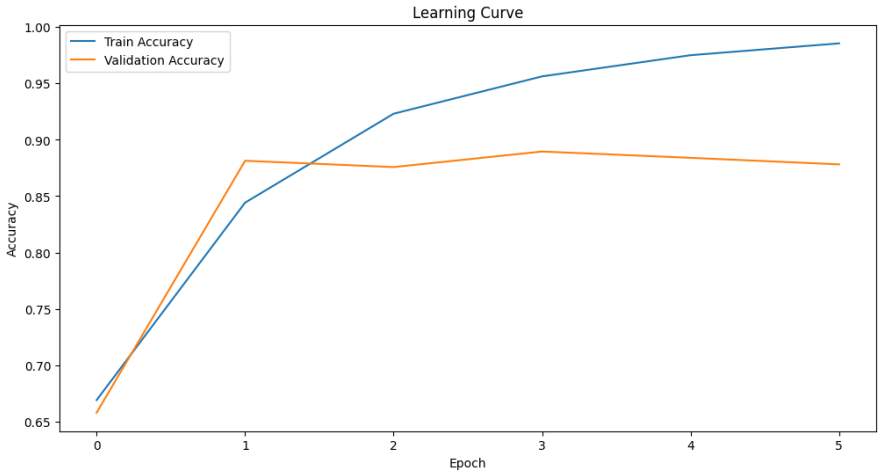

Рисунок 17 - Крива навчання для RNN

Таблиця 5

Звіт для RNN

Class	Precision	Recall	F1-score	Support
0	0.87	0.86	0.86	12500
1	0.86	0.87	0.87	12500
accuracy	0.87	0.87	0.87	12500
macro avg	0.87	0.87	0.87	12500
weighted avg	0.87	0.87	0.87	12500

На кривих навчання (Рисунок 13, Рисунок 15, Рисунок 17) можемо побачити схожу тенденцію зменшення похибки на тренувальних і тестових наборах. Починаючи з певної епохи (ANN: 3, CNN: 6, RNN: 3) моделі почали демонструвати ознаки перенавчання, тому двома епохами пізніше навчання було зупинено. Точність прогнозування для ANN склала 90%, для CNN - 90% та 87% для RNN, відповідно.

Висновки:

- CNN показала найкращі результати за всіма показниками, що вказує на високу здатність розпізнавати особливості вхідних даних, де важлива локальна структура.
- RNN показує кращі результати ніж ANN, що може бути корисним у випадках з послідовними даними, такими як текст у тому числі.
- ANN має найнижчу точність, але все ще є ефективною у випадках з менш складними даними.

Висновки. З дослідження можемо зробити висновок, що за допомогою статичних методам та нейронним мережам можливо досягти точності прогнозування близько 90 відсотків без значних обчислювальних витрат та витрат часу на навчання класифікатора. Такі методи як Логістична Ресресія, CNN та RNN мают ширових простір для налаштування параметрів та можуть бути підлаштовані для різноманітних задач. Якщо є бізнес-вимога у вищій точності прогнозування, то, можливо, є сенс у використанні трансформерів та LLM (Large Language Model). При цьому має місце аспект наявності необхідних обчислювальних потужностей та економічної доцільності використання Open API від LMM (GPT-3.5 Turbo, GPT-4 та інші).

ЛІТЕРАТУРА/REFERENCES

1.Source Code for the Article. URL: https://github.com/w3t4nu5/NLP-Article

2. IMDB Dataset of 50K Movie Reviews.
URL: https://www.kaggle.com/datasets/lakshmi25npathi/imdb-dataset-of-50k-movie-reviews

3. HuggingFace: Transformers. URL: https://huggingface.co/docs/transformers/index

4. Stopwords [NLP, Python]. URL: https://medium.com/@yashj302/stopwords-nlp-python-4aa57dc492af

5. Pavliuk, D. I., Baibuz, O. H., and Honcharova, Y. S. "Text Preparation for Natural Language Processing." 'XIX International Scientific and Practical Conference “Creative Business Management and Implementation of New Ideas”', 14-17 May 2024, Tallinn, Estonia, pp. 223-225.

6. Feature extraction. URL: https://scikit-learn.org/stable/modules/feature_extraction.html#text-feature-extraction

7. Mikolov T., Chen K., Corrado G., Dean J. Efficient Estimation of Word Representations in Vector Space. 2013. URL: https://arxiv.org/pdf/1301.3781

8. MultinomialNB.
URL: https://scikit-learn.org/stable/modules/generated/sklearn.naive_bayes.MultinomialNB.html

9. Support Vector Machines. URL: https://scikit-learn.org/stable/modules/svm.html

10. LogisticRegression. URL: https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LogisticRegression.html

ISSN 1562-9945 (Print)
ISSN 2707-7977 (Online)

11. Elastic Net Regression —Combined Features of L1 and L2 regularization. URL: https://medium.com/@abhishekjainindore24/elastic-net-regression-combined-features-of-l1-and-l2-regularization-6181a660c3a5
12. Google Code: word2vec. URL: https://code.google.com/archive/p/word2vec/
13. Natural Language Processing in TensorFlow. URL: https://www.coursera.org/learn/natural-language-processing-tensorflow/home/week/1

Received 17.06.2024.
Accepted 21.06.2024.

Review of methods for semantic text classification

Recent advancements in text classification have focused on the application of machine learning and deep learning techniques. Traditional methods such as Naive Bayes, Logistic Regression, and Support Vector Machines (SVM) have been widely utilized due to their efficiency and simplicity. However, the advent of deep learning has introduced more complex models like Artificial Neural Networks (ANN), Convolutional Neural Networks (CNN), and Recurrent Neural Networks (RNN), which can automatically extract features and detect intricate patterns in textual data. Additionally, transformer-based models such as BERT have set new benchmarks in text classification tasks. Despite their high accuracy, these models require substantial computational resources and are not always practical for every application. The ongoing research aims to balance accuracy and computational efficiency.

Purpose of Research. The primary objective of this study is to review and compare various methods for automated text classification based on sentiment analysis. This research aims to evaluate the prediction accuracy of different models, including traditional machine learning algorithms and modern deep learning approaches, and to provide insights into their practical applications and limitations.

Presentation of the Main Research Material. This study utilizes the "IMDB Dataset of 50K Movie Reviews" to train and test various text classification models. The dataset comprises movie reviews and their associated sentiment labels, either positive or negative. The research employs several preprocessing steps. For feature extraction, methods such as Bag-of-Words (BoW), TF-IDF (Term Frequency-Inverse Document Frequency), and Word2Vec are used. These features are then fed into various classifiers: Naive Bayes, Support Vector Machines (SVM), Logistic Regression, Deep Learning Models.

Conclusions. The comparative analysis reveals that while traditional machine learning methods like Naive Bayes, SVM and Logistic Regression are efficient and easy to implement, deep learning models offer superior accuracy by capturing more complex patterns in the data. However, the computational demands of deep learning models, particularly transformers, limit their applicability in resource-constrained environments. Future research should focus on optimizing these models to balance accuracy and computational efficiency, making advanced text classification accessible for a broader range of applications.

Recent advancements in text classification have focused on the application of machine learning and deep learning techniques. Traditional methods such as Naive Bayes, Logistic Regression, and Support Vector Machines (SVM) have been widely utilized due to their efficiency and simplicity. However, the advent of deep learning has introduced more complex models like Artificial Neural Networks (ANN), Convolutional Neural Networks (CNN), and Recurrent Neural Networks (RNN),

which can automatically extract features and detect intricate patterns in textual data. Additionally, transformer-based models such as BERT have set new benchmarks in text classification tasks. Despite their high accuracy, these models require substantial computational resources and are not always practical for every application. The ongoing research aims to balance accuracy and computational efficiency.

Павлюк Дмитро Іванович – аспірант 1-го року навчання Дніпровського національного університету імені Олеся Гончара.
Байбуз Олег Григорович - доктор технічних наук, професор, завідувач кафедри математичного забезпечення ЕОМ Дніпровського національного університету імені Олеся Гончара.

Pavliuk Dmytro - postgraduate student of the 1st year of study at the Dnipro National University named after Oles Honchar.
Baibuz Oleh - doctor of technical sciences, professor, head of the department of computer mathematical support of Dnipro National University named after Oles Honchar.

ISSN 1562-9945 (Print)
ISSN 2707-7977 (Online)