Міністерство освіти і науки України

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «КИЄВО-МОГИЛЯНСЬКА АКАДЕМІЯ»

Кафедра інформатики

Застосування наївного байєсівського класифікатора для класифікації текстових повідомлень

Текстова частина до дипломної роботи за спеціальністю 121 «Інженерія програмного забезпечення»

Керівник кваліфікаційної роботи

доцент, кандидат наук Олецький О. В.

_____

(підпис)

“___” _____ 2025 року

Виконав студент БП ІПЗ-4

Андрусенко Анатолій Дмитрович

“___” _____ 2025 року

Київ 2025

ЗМІСТ

Міністерство освіти і науки України

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «КИЄВО-МОГИЛЯНСЬКА АКАДЕМІЯ»

Кафедра інформатики

ЗАТВЕРДЖУЮ

Завідувач кафедри інформатики,

доцент, кандидат наук С. С. Гороховський

_____

(Підпис)

“___” _____ 2024 року

ІНДИВІДУАЛЬНЕ ЗАВДАННЯ

для кваліфікаційної роботи

студенту 4 року БП ІПЗ факультету інформатики

Андрусенку Анатолію Дмитровичу

Тема: “Застосування наївного байєсівського класифікатора для класифікації текстових повідомлень”

Зміст:

Вступ

1. Теоретичні засади

2. Сучасні покращення наївного байєсівського класифікатора

3. Розробка застосунку та практичні результати

Висновок

Список літератури

Дата видачі 30 вересня 2024 р.

Керівник _____ (підпис)

Завдання отримав _____ (підпис)

КАЛЕНДАРНИЙ ПЛАН ВИКОНАННЯ РОБОТИ

	Етап роботи	Дата
	Початок роботи, обговорення теми та мети	1.10.2024-1.11.2024
	Пошук літератури	2.11.2024-1.2.2025
	Написання першого розділу роботи	2.2.2025-15.2.2025
	Написання другого розділу роботи	16.2.2025-1.3.2025
	Робота над практичною частиною роботи, збір експериментальних результатів	2.3.2025-1.4.2025
	Написання третього розділу роботи	2.4.2025-1.5.2025
	Написання висновків до роботи	2.5.2025-10.5.2025

АНОТАЦІЯ

У роботі досліджуються різні варіації наївного байєсівського класифікатора для задачі класифікації текстових повідомлень. Проведено огляд технік попередньої обробки тексту, методів векторизації, а також підходів до згладжування і вибору ознак. На основі отриманих висновків реалізовано десктоп-застосунок з графічним інтерфейсом на базі Tkinter, який дозволяє інтерактивно підбирати параметри попередньої обробки, векторизації, вибору ознак та типу байєсівської моделі, проводити навчання та отримувати метрики якості класифікації.

ВСТУП

В останні десятиліття текстові повідомлення стали основною формою комунікації у багатьох цифрових середовищах — від месенджерів і форумів до корпоративних чатів та мобільних додатків. Кожен день в інтернеті з’являються мільйони коротких повідомлень, коментарів, відгуків та постів, що віддзеркалюють думки, емоції та наміри їх авторів. Ці дані є цінним джерелом інформації для аналізу суспільної думки, маркетингових досліджень, виявлення фейків тощо.

Однією з ключових задач є класифікація повідомлень: визначення тематики, емоційного забарвлення, токсичності, належності до спаму. Враховуючи швидкість та об’єм появи нових повідомлень, ручна модерація та класифікація стає майже неможливою, що потребує використання автоматизованих методів обробки текстової інформації.

Одним з найвідоміших та простих методів класифікації текстів є наївний байєсівський класифікатор. Незважаючи на свою теоретичну простоту та наївне припущення про незалежність ознак, ця модель показує хороші результати у прикладних задачах, особливо за обмежених обчислювальних ресурсів та невеликих об’ємів навчальних даних. Класичний наївний байєс виділяється високою швидкістю та низькою складністю у реалізації, що робить його частим вибором для реалізації систем аналізу тексту.

Втім, класифікація коротких текстів — таких як повідомлення в месенджерах, форумах або коментарі користувачів — створює специфічні труднощі для традиційного NB. Невеликий об’єм тексту, висока розрідженість ознак, сленг, помилки, емодзі, хештеги та інші нестандартні елементи ускладнюють точну оцінку ймовірностей та погіршують якість класифікації. Тому в останні роки активно проводяться дослідження, що мають на меті модифікацію та розширення наївного байєсівського метода,

у тому числі через покращене представлення тексту, введення ваг ознак, використання семантичних вкладань слів та гібриди з іншими алгоритмами.

Таким чином, актуальність цієї роботи обумовлена як високою практичною значущістю задачі класифікації повідомлень, так і потребою в дослідженні та реалізації покращених підходів до наївного байєсівського класифікатора для коротких текстів.

Ця робота досліджує такі задачі:

- вивчити існуючі підходи до класифікації текстових даних;
- проаналізувати обмеження класичного NB для коротких повідомлень;
- дослідити сучасні методи покращення NB;
- реалізувати застосунок для класифікації коротких повідомлень;
- провести тестування та оцінку ефективності моделі на вибраних наборах даних.

РОЗДІЛ 1. ТЕОРЕТИЧНІ ЗАСАДИ.

1.1 Загальні теоретичні відомості

Наївний байєсівський класифікатор (NB) — це простий і водночас ефективний імовірнісний метод класифікації, що базується на застосуванні теореми Байєса:

$$P(C_k|\, x) \;=\; \frac{P(x \mid C_k)P(C_k)}{P(x)}$$

де:

- $P(C_k)$ – апріорна ймовірність класу C_k,
- $P(x \mid C_k)$ – ймовірність спостереження ознак x за умови, що об’єкт належить до класу C_k,
- $P(x)$ – ймовірність спостереження ознак x. Оскільки цей множник однаковий для всіх класів, то при порівнянні ймовірності класів його зазвичай опускають.

У класичному «наївному» байєсівському класифікаторі робиться сильне припущення незалежності ознак x в межах одного класу. Тоді:

$$P(x \mid C_k) \;=\; \prod\nolimits_{i=1}^{n} P(x_i \mid C_k)$$

Таким чином, кінцева формула, за якою приймається рішення, набуває вигляду:

$$\hat{k} \;=\; \underset{\mathrm{k}}{\operatorname{argmax}}\, P(C_k) \prod\nolimits_{i=1}^{n} P(x \mid C_k)$$

Зазвичай цю формулу логарифмують, оскільки множення великої кількості малих ймовірностей може призвести до числових похибок:

$$\hat{k} = \underset{k}{\operatorname{argmax}} \left(\log\big(\mathrm{P(C_k)}\big) + \sum\nolimits_{\mathrm{i=1}}^{\mathrm{n}} \log\big(\mathrm{P(x_i|C_k)}\big) \right)$$

Серед класифікаторів NB має найменшу обчислювальну складність, і незважаючи на припущення незалежності ознак, показує достатньо високу точність [1], [2]. Переваги цього методу у його швидкості у тренуванні, та ефективності навіть на невеликих об’ємах даних.

У залежності від типу даних та припущень про розподіл ознак виділяють три основні варіанти наївного байєсівського класифікатора: Multinomial NB, Bernoulli NB, Gaussian NB.

1.2 Multinomial NB

Multinomial NB робить припущення, що ймовірність вектору ознак x = ($x_1, \dots x_v$) у класі C_k обчислюється через мультиноміальний розподіл:

$$P(x \mid C_k) = \frac{(\sum_{i=1}^{V} x_i)!}{\prod_{i=1}^{V} x_i!} \prod_{i=1}^{V} P(x_i \mid C_k)^{x_i}$$

де $P(x_i \mid C_k)$ – ймовірність входження ознаки x_i у клас C_k, яка рахується як частота її входження у клас.

Цей варіант найбільш поширений у текстовій класифікації.

Якщо частота входження ознаки у клас рівна нулю, то ймовірність цього вектору ознак прирівнюється нулю, що негативно впливає на точність класифікації. У класичному підході ця проблема вирішується за допомогою згладжування:

$$P(x_i|C_k) = \frac{N_{ik} + a}{N_k + ad}$$

де N_{ik} – кількість разів, коли ознака x_i з’являється у класі C_k, d – кількість можливих ознак (словник), a – параметр згладжування (зазвичай має значення 1).

1.3 Bernoulli NB

Bernoulli NB припускає, що кожна ознака є бінарною. У контексті текстової класифікації, документ подається як вектор ознак x = ($x_1, \dots x_v$), де кожна ознака x_i – це значення 0 або 1, в залежності від того, входить термін у документ чи ні. Ймовірність обчислюється як:

$$P(x \mid C_k) = \prod_{i=1}^{V} \left[{p_{k,i}}^{x_i} (1 - p_{k,i})^{1-x_i} \right]$$

де $p_{k,i} = P(x_i = 1 | C_k)$ – відношення кількості документів класу C_k, що містять термін і, до загальної кількості документів класу C_k. Таким чином, цей алгоритм не враховує частоту появи ознаки у документі, а лише частку документів, у яких він з'являється. Загалом, працює трохи гірше у задачах текстової класифікації, ніж Multinomial NB, але має перевагу на коротких текстах та невеликих наборах даних [3].

1.4 Gaussian NB

Gaussian NB припускає, що для кожного класу значення ознаки x_i задане нормальним розподілом:

$$P(x_i \mid C_k) = \frac{1}{\sqrt{2\pi {\sigma^2}_{k,i}}} \exp\left(-\frac{(x_i - \mu_{k,i})^2}{2{\sigma^2}_{k,i}} \right)$$

де $\mu_{k,i}$ та ${\sigma^2}_{k,i}$ – оцінені середнє та дисперсія ознаки i у класі C_k. Підходить для безперервних числових даних, але припущення нормальності розподілу часто не виконується у текстових ознаках. У статті [4] зокрема проведено порівняння влучності (precision), повноти (recall) та F1 метрики класичних алгоритмів NB на текстових даних. Gaussian NB показує суттєво гірші результати, ніж інші моделі.

1.5 Complement NB (CNB)

CNB – це різновид наївного байєсівського класифікатора, що був розроблений з метою усунення проблеми дисбалансу класів та підвищення стабільності моделі у задачах текстової класифікації.

Замість оцінки ймовірності ознак для кожного класу, як це робить стандартний MNB, CNB обчислює ймовірність ознак на основі сукупності усіх інших класів.

У випадках, коли деякі класи мають значно більше даних, ніж інші, стандартний MNB має тенденцію переоцінювати вагу ознак для класів з більшою кількістю даних. Натомість, CNB обчислює ймовірності ознак для всіх інших класів і таким чином знижує вплив домінуючого класу.

1.6 Висновки до розділу

Наївний байєсівський класифікатор є одним із найпростіших і водночас ефективних методів машинного навчання, що базується на принципі умовної незалежності ознак. Незважаючи на спрощені припущення, він демонструє високу точність у задачах текстової класифікації завдяки обчислювальній простоті та здатності працювати з малими об’ємами даних.

У цьому розділі було розглянуто основні варіації наївного байєсівського класифікатора – Multinomial NB, Bernoulli NB, Gaussian NB, Complement NB.

- Multinomial NB орієнтований на врахування частот ознак та є найбільш поширеним у задачах текстової класифікації. Важливим аспектом його реалізації є застосування методів згладжування, що дозволяє уникнути впливу рідкісних ознак.
- Bernoulli NB базується на бінарних представленнях ознак і є корисним для коротких текстів та невеликих корпусів даних, оскільки фокусує увагу лише на наявності або відсутності термінів, а не на їх частоті.
- Gaussian NB застосовується до числових даних, моделюючи ймовірності за нормальним розподілом. Однак, у задачах текстової

класифікації його ефективність є обмеженою через відсутність чіткої інтерпретації числових характеристик текстів.

- Complement NB, що був розроблений для зменшення впливу домінуючих класів та корекції дисбалансу даних. Використовуючи ймовірності комплементарних класів, CNB підвищує стабільність моделі.

Таким чином, основною перевагою NB є простота та швидкість навчання, що дозволяє використовувати його як базову модель для задач текстової класифікації. У наступному розділі буде детально розглянуто сучасні модифікації байєсівських алгоритмів, спрямовані на покращення їх точності та стійкості до шуму в даних.

РОЗДІЛ 2. СУЧАСНІ ПОКРАЩЕННЯ НАЇВНОГО БАЙЄСІВСЬКОГО КЛАСИФІКАТОРА

Наївний байєс широко використовується у задачах класифікації текстів завдяки своїй простоті та високій точності, проте він має низку недоліків, як-от гіпотеза про незалежність ознак, а також проблема нульової частоти. У сучасних дослідженнях розроблено методи, що мають на меті вирішити ці проблеми та покращити точність класифікації.

2.1 Покращення на рівні представлення тексту.

Вибір методів представлення тексту ускладнюється тим, що важкі методи попередньої обробки тексту сильно впливають на швидкість роботи класифікатора, тим самим зводячи нінащо його основну перевагу. Далі наведено список технік, що використовуються для текстової класифікації.

2.1.1 Загальні методи видалення малоінформативних елементів

Одним з ключових кроків у попередній обробці текстових повідомлень є видалення елементів, що не несуть корисної інформації для класифікації. До таких елементів відносяться:

- Пунктуація та спеціальні символи
- Посилання
- Згадки користувачів (наприклад @username)
- HTML-теги

В залежності від задачі такі елементи можна видаляти повністю, або ж заміняти на стандартні терміни, наприклад, згадку користувача на <user>.

Видалення або заміна таких компонентів дозволяє знизити розмір словника ознак, зменшити кількість унікальних токенів.

2.1.2 Видалення шумових слів

Шумові слова – це такі слова, що часто зустрічаються та не несуть корисної інформації, наприклад «the», «is» в англійській мові. Їх видалення є частою практикою в попередній обробці тексту. Утім, для текстової класифікації, у більшості випадків їх використання не має позитивного ефекту [5]. Також, загальнодоступні словники шумових слів можуть містити слова, що несуть інформацію для конкретних задач класифікації.

2.1.3 Видалення рідкісних слів

Видалення рідкісних слів має високу ефективність для наївного байєса [6]. Суть техніки в тому, щоб видаляти слова, що зустрічаються занадто рідко (наприклад, друкарські помилки). Такі слова практично не несуть користі, і отримують занадто високу умовну ймовірність в одному класі.

2.1.4 Врахування n-грам

Розбиття тексту на окремі слова (уніграми) може бути недостатньо, щоб зберегти контекст у короткому повідомленні. Словосполучення різної довжини можуть допомогти його врахувати. Наприклад, словосполучення «not good» вказує на негативне забарвлення повідомлення. Автори статті [7] провели порівняння точності класифікації наївного байєса для n-грам різної довжини. Отримані дані вказують, що використання n-грам довжини 2 (біграм) та 3 (триграм) разом з уніграмами суттєво підвищує точність. Інше дослідження [6] показало, що уніграми мають найбільшу інформативність, і що використання тільки біграм сильно знижує точність.

2.1.5 Стемінг і лематизація

Стемінг та лематизація – це дві поширені техніки у передобробці тексту, котрі працюють схожим чином.

Стемінг – процес зведення слова до його основної або кореневої форми, яка може бути не обов’язково словниковою. Основна мета стемінгу – усунути префікси та суфікси, зберігаючи корінь слова. Стемінг не враховує контекст слова та не завжди забезпечує граматично правильний результат. Наприклад, він зводить слово studies до форми studi.

Лематизація – процес зведення слова до його словникової форми – леми, враховуючи частину мови та контекст. Лематизація використовує морфологічний аналіз, що дозволяє отримати граматично правильні форми слів. Приклад: studies – study.

Дослідження [8] аналізує вплив цих двох методів на точність класифікації емоцій з використанням наївного байєсівського класифікатора. Автори виявили, що стемінг підвищує точність класифікації, об’єднуючи різні форми слів в одну основу, що зменшує розмір словника та покращує здатність моделі до узагальнення. Лематизація, хоча й потребує більших обчислювальних ресурсів, не завжди призводить до значного покращення точності у порівнянні зі стемінгом.

2.1.6 TF-IDF та вплив частотних ваг

TF-IDF (Term Frequency – Inverse Document Frequency) – це метод надання ваг словам у текстах, котрий враховує як частоту появи слова у конкретному документі, так і його розповсюдженість у корпусі документів. Цей підхід дозволяє виділити найбільш інформативні терміни та знизити вплив слів, що часто зустрічаються, та мають менше значення.

TF-IDF складається з двох компонентів:

- TF (Term Frequency) – частота, з якою слово зустрічається в документі

- IDF (Inverse Document Frequency) – зменшує вагу слів, що часто зустрічаються у всіх документах, і збільшує вагу рідкісних слів.

Формула TF-IDF для слова t в документі d виглядає наступним чином:

$$\mathrm{TF-IDF}(t,d) \;=\; \mathrm{TF}(t,d) \times \log\left(\frac{N}{\mathrm{DF}(t)}\right)$$

де N – загальна кількість документів, DF(*t*) – кількість документів, у яких зустрічається слово t.

Наївний байєсівський класифікатор (NB) традиційно використовує частотне представлення слів, що може призводити по переоцінки слів, що часто зустрічаються. Використання TF-IDF дозволяє покращити точність класифікації за рахунок більш точного відображення ваги слів, знизити вплив слів загального вжитку.

Дослідження [9] показало, що використання TF-IDF з наївним байєсівським класифікатором досягло точності 96.2% для класифікації спам-повідомлень в SMS, що перевершує результат стандартного підходу Bag-of-Words.

2.2 Вбудовування семантичної інформації

Класичний NB представляє текст як мішок слів**,** ігноруючи порядок та семантичні зв’язки між словами. Це спрощення може бути недостатнім при обробці коротких і неформальних текстів. Для того, щоб обійти це обмеження використовуються вкладання слів (word embeddings), такі як Word2Vec, fastText та GloVe, що дозволяють враховувати семантичну близькість між словами.

Вкладання слів перетворюють слова на вектори фіксованої довжини, що відображають їх контекстне значення. Для інтеграції цих вкладань слів в NB необхідно зібрати вектори слів у представлення всього документа.

Один із розповсюджених методів – усереднення векторів усіх слів у документі.

Дослідження [10] демонструє, що використання Word2Vec разом з NB може підвищити точність класифікації текстів. Зокрема, при класифікації відгуків на фільми точність моделі з Word2Vec досягла 86%, що на 4% вище результатів базової моделі.

Кластеризація слів на основі вкладань слів дозволяє групувати семантично близькі слова, що може підвищити здібність моделі до узагальнення. Наприклад, слова «fun», «pleasure», «joy» можуть бути об’єднані в один кластер, що дозволяє NB враховувати їх схоже значення при класифікації.

Вкладання слів особливо корисні при обробці синонімів і сленгу, що часто зустрічаються у текстових повідомленнях. Дослідження [11] показує, що використання fastText з NB призводить до покращення точності класифікації текстів, особливо в умовах обмеженого об’єму навчальних даних.

2.3 Покращення наївного байєсівського класифікатора з використанням методів згладжування

У статті [12] представлено дослідження, що присвячено покращенню наївного байєсівського класифікатора для задач класифікації коротких текстів з використанням різних методів згладжування.

В задачах класифікації коротких текстів стандартні методи згладжування, такі як згладжування Лапласа та згладжування Ліндстоуна, можуть працювати недостатньо ефективно через високу розрідженість даних (data sparsity). Основною проблемою є відсутність спостережень для деяких слів, що призводить до присвоєння нульових ймовірностей. Запропоновані методи:

- Згладжування Лапласа (Laplace Smoothing): додавання одиниці до усіх частот термінів для того, щоб запобігти появі нульових ймовірностей.
- Згладжування Ліндстоуна (Lindstone Smoothing): узагальнена версія згладжування Лапласа, де додається параметр, що дозволяє гнучко регулювати ступінь згладжування.
- Jelinek-Mercer Smoothing: ідея у змішуванні емпіричної ймовірності з апріорною ймовірністю, обчисленою на всьому корпусі. Формула:

$$P(x_i|C_k) = (1-\lambda)\frac{N_{ik}}{\sum_{j=1}^{V} N_{jk}} + \lambda \cdot P(x_i)$$

Де λ – параметр згладжування в діапазоні від 0 до 1, N_{ik} – кількість появ слова x_i у класі C_k, $\sum_{j=1}^{V} N_{jk}$ – кількість слів у класі C_k, $P(x_i)$ – апріорна ймовірність слова x_i у всьому корпусі.

- Згладжування Діріхле (Dirichlet Smoothing). Його ідея у тому, щоб додавати до частот слів ймовірність слова по всьому корпусу:

$$P(x_i|C_k) = \frac{N_{ik} + \lambda \cdot P(x_i)}{\sum_{j=1}^{V} N_{jk} + \lambda}$$

- Абсолютне дисконтування (Absolute Discounting): зменшення ймовірності частих слів і перерозподіл ймовірностей на рідкісні терміни. Формула:

$$P(x_i|C_k) = \frac{max(N_{ik} - \lambda, 0)}{\sum_{j=1}^{V} N_{jk}} + \lambda \cdot \frac{|C_k|_u}{\sum_{j=1}^{V} N_{jk}} \cdot P(x_i)$$

де $|C_k|_u$ – кількість унікальних слів у класі C_k.

- Two-Stage Smoothing. Цей тип згладжування об’єднує у собі підходи Jelinek-Mercer і Dirichlet згладжування. Формула:

$$P(x_i|C_k) = (1-\lambda)\frac{N_{ik} + \mu \cdot P(x_i)}{\sum_{j=1}^{V} N_{jk} + \mu} + \lambda \cdot P(x_i)$$

У дослідженні були протестовані запропоновані методи згладжування на вибірці коротких текстів з SMS-повідомлень. Результати дослідження показали, що згладжування Лапласа збільшило точність класифікації на 2%, нале мало обмежений вплив на рідкісні слова. Згладжування Ліндстоуна показало кращий баланс між точністю та повнотою. Абсолютне дисконтування забезпечило найбільший приріст точності (до 4%) за рахунок перерозподілу ймовірностей серед рідкісних слів.

2.4 Зважені наївні байєсівські класифікатори

Наївний байєсівський класифікатор робить припущення про умовну незалежність ознак. Реальні дані, особливо тексти, зазвичай не дотримуються цього припущення, що впливає на точність класифікації. Для пом’якшення цієї проблеми були розроблені методи зважування ознак і класів.

Зважений наївний байєсівський класифікатор (Weighted Naïve Bayes, WNB) модифікує стандартний NB, вводячи ваги для ознак, що відображають їх значущість. Це дозволяє враховувати кореляції між ознаками та покращувати класифікацію. Дослідження показують, що WNB перевершує стандартний NB, особливо при порушенні припущення про незалежність ознак [13].

2.4.1 Класово-зважений наївний байєс (CAWNB)

У статті [14] пропонується новий підхід до зважування ознак для NB. Основна ідея полягає у тому, що замість використання однакової ваги для всіх класів (як це робиться у класичному NB або у General Attribute

Weighted Naive Bayes – GAWNB), запропоновано використовувати окремі ваги для кожного класу.

Для зважування атрибутів у CAWNB використовується формула:

$$\hat{k}(x) = \underset{k}{argmax}\left(P(C_k) \cdot \prod_{i=1}^{m} [P(x_i|C_k)^{w_{ik}}]\right)$$

Для кожного класу C_k та ознаки x_i визначається окрема вага w_{ik}.

Запропоновано два методи оптимізації ваг:

- CAWNB CLL – оптимізації ваг за допомогою максимізації умовної логарифмічної ймовірності (Conditional Log-Likelihood).
- CAWNB MSE – оптимізація шляхом мінімізації середньоквадратичної похибки (Mean Squared Error).

Алгоритм починається з ініціалізації ваг усіх атрибутів для кожного класу до 1. Далі використовуються методи оптимізації (L-BFGS-M) для знаходження оптимальних ваг. Обчислення прогнозу для тестових даних виконується з використанням оновлених ваг.

У дослідженні було проведено експерименти на 36 наборах даних. CAWNB показав кращі результати, ніж GAWNB та інші методи зважування, оглянуті у статті. CAWNB CLL та CAWNB MSE значно перевершують інші методи за точністю та середньою квадратичною похибкою.

Таким чином, CAWNB є більш детальним та гнучким підходом до зважування атрибутів, що дозволяє покращити продуктивність класифікатора за рахунок адаптивного підходу до зважування для кожного класу. Це особливо корисно у випадках, коли класи суттєво відрізняються за розподілом ознак.

2.4.2 Покращений мультиноміальний наївний Байєс із вагами ознак на основі кореляції відстаней **(IDCWMNB)**

У статті [16] запропоновано метод зваженої класифікації тексту на основі NB, який використовує покращений коефіцієнт кореляції відстаней.

Традиційні методи зважування ознак для текстової класифікації на основі NB покладаються на лінійні залежності між ознаками і класами. Однак, у випадку коли дані мають нелінійні залежності, ці методи стають менш ефективними. Автори пропонують використовувати покращений коефіцієнт кореляції відстаней, котрий більш ефективно враховує залежності між ознаками і класами.

У методі використовується міра Inverse Document Frequency (IDF) для врахування частоти слів у різних документах, що дозволяє посилити значущість рідкісних, але важливих термінів.

Основна формула класифікації з врахуванням ваг:

$$\hat{k}(x) = \underset{k}{argmax}\left(\ln\left(P(C_k)\right) + \sum_{i=1}^{m} w_{ik} \cdot f_i \cdot \ln\left(P(x_i|C_k)\right)\right)$$

де f_i – частота слова у документі. Вага w_{ik} рахується за формулою:

$$w_{ik} = \frac{D_{ik} \cdot m}{\sum_{j=1}^{m} D_{jk}} \cdot e^{-2 \cdot \frac{nd(x_i)}{n}}$$

де D_{ik} – кореляція ознаки x_i з класом C_k. Чим вище $D_{ik,}$ тим більше вплив ознаки на даний клас. $nd(x_i)$ – загальна ознаки x_i у документах.

IDCWMNB показав вищу точність класифікації (86.35%), коли класичний MNB – 82.44%.

2.5 Висновки до розділу

У цьому розділі було розглянуто сучасні методи покращення наївного байєсівського класифікатора (NB) з метою підвищення його

точності, стійкості до шуму та адаптивності до коротких текстів. Основними напрямками удосконалення стали методи попередньої обробки тексту, вбудовування семантичної інформації та використання зважених моделей. Також були оглянуті різні методи згладжування, гнучкіші за стандартні методи згладжування Лапласа та Ліндстоуна. Альтернативні методи згладжування дозволяють перерозподілити ймовірності більш збалансовано, що особливо актуально для текстів з високою розрідженістю даних.

У наступному розділі буде розглянуто практичну реалізацію деяких з описаних методів, їх інтеграцію у загальний конвеєр класифікації тексту та порівняння результатів на реальних наборах даних.

РОЗДІЛ 3. РОЗРОБКА ЗАСТОСУНКУ ТА ПРАКТИЧНІ РЕЗУЛЬТАТИ

У цьому розділі описується процес проєктування та реалізації десктоп-застосунку для класифікації текстових даних наївним байєсівським класифікатором. Основна мета – створити інструмент, що дозволяє досліднику чи практику швидко експериментувати з різними комбінаціями попередньої підготовки тексту, векторизації та моделей NB, при цьому зберігаючи високий рівень продуктивності і не жертвуючи прозорістю та відтворюваністю результатів. Оскільки першочергова перевага алгоритму перед іншими класифікаторами – його швидкість, то деякі методи попередньої підготовки тексту можуть бути неефективними через свою обчислювальну складність. Таким чином, при розробці акцент було поставлено на балансі між якістю класифікації та швидкістю виконання. З одного боку, для NB важливо мати важливо мати достатньо інформативні ознаки, а з іншого – надмірна попередня підготовка тексту чи занадто велика кількість ознак призводять до уповільнення алгоритму.

Для розробки було обрано мову Python, через її зручність та популярність у розробці для задач машинного навчання. Для створення інтерфейсу було використано бібліотеку Tkinter - популярний та простий програмний пакет для розробки десктопних застосунків.

3.1 Структура застосунку

У цьому підрозділі описані основні екрани програми, структуру елементів керування та принципи взаємодії користувача з додатком.

Застосунок складається з двох основних сторінок – експерименту та передбачення. Перехід між сторінками відбувається через панель навігації.

3.1.1 Сторінка «Experiment»

Сторінка експерименту - це центральний робочий простір, розбитий на три вкладки (ttk.Notebook):

1) **Params**. Ця вкладка відповідає за усі налаштування попередньої обробки, векторизації, відбору ознак та NB-моделі перед тренуванням.
2) **Data.** У цій вкладці користувач вибирає файл, з якого потрібно взяти дані для тренування, колонки у файлі, які відповідають за дані та мітки класів; режим тренування.
3) **Info.** Після тренування моделі у вкладці інформації відображаються метрики, що характеризують ефективність моделі. Також тут розташовані кнопки збереження моделі у файл для подальшого використання, та завантаження моделі з файлу.

3.1.2 Сторінка «Prediction»

На цій сторінці розташовується поле вводу власних даних для класифікації. Класифікатор повертає клас, що має найбільшу ймовірність. У випадку, якщо впевненість класифікатора низька (ймовірність класу менша за 60%), застосунок виводить три найбільш імовірні класи.

3.2 Використання бібліотеки scikit-learn.

Для створення конвеєру (pipeline) класифікації, було використано програмний пакет scikit-learn. API цієї бібліотеки дозволяє створювати власні алгоритми класифікації, побудовані на основі інтерфейсів, що забезпечує їх інтегрованість з іншими модулями бібліотеки. Pipeline у scikit – це механізм послідовного застосування кількох кроків трансформування (попередня обробка, векторизація, відбір ознак тощо) і класифікатора. Сирі тексти спочатку проходять через усі проміжні кроки, а потім використовуються для навчання NB-моделі. Завдяки цьому усі трансформації відбуваються однаково під час тренування і під час передбачення на нових текстах.

3.2.1 Розробка власних наївних байєсівських алгоритмів

Щоб створити власні класифікатори, сумісні з екосистемою scikit-learn, необхідно дотримуватися єдиного інтерфейсу, визначеного базовими класами BaseEstimator та ClassifierMixin. Методи, котрі повинен реалізовувати класифікатор:

- fit(self, X, y). Призначення – навчити модель на вибірці X та мітках y
- predict(self, X). Для кожного прикладу з вибірки X обчислити найімовірніший клас за внутрішніми параметрами моделі
- predict_proba(self, X). Для кожного прикладу з X обчислює ймовірності до всіх класів
- partial_fit(self, X, y, classes=None). Цей метод дозволяє тренувати модель на новій порції даних, не втрачаючи вже накопиченого стану.

3.2.1.1 Absolute Discounting NB

У класі AbsoluteDiscountingNB було реалізовано варіацію мультиноміального байєсівського класифікатора, у якій згладжування відбувається через формулу абсолютного дисконтування, за якого ймовірність перерозподіляється з частих термінів на більш рідкісні, замість класичного додавання абсолютної величини до частоти кожного терміна. Абсолютне дисконтування відбувається у методі _update_feature_log_prob (Рис. 1):

```
def _update_feature_log_prob(self, n_features):
    discounted_fc = np.maximum(self.feature_count_ - self.alpha, 0)
    total_discount = self.alpha * np.count_nonzero(self.feature_count_, axis=1)
    normalizer = discounted_fc.sum(axis=1) + total_discount

    self.feature_log_prob_ = np.log(
        (discounted_fc + (total_discount[:, np.newaxis] / n_features)) / (normalizer[:, np.newaxis] + 1e-10)
    )
```

Рисунок 1. Реалізація абсолютного дисконтування

Для кожного класу і кожної ознаки з матриці feature_count_ віднімаємо константу alpha. Усі від’ємні результати замінюємо на нуль.

Для кожного класу ми підраховуємо, скільки ознак мали ненульову оригінальну частоту (count_nonzero), а потім множимо цю кількість на alpha. Це дає загальний «об’єм» ймовірнісної маси, звільненої в результаті дисконтування.

Щоб сума ймовірностей була рівна 1, ми складаємо суму всіх «зрізаних» частот для класу (рядок discounted_fc.sum(axis=1)) з total_discount.

Далі кожній з можливих ознак додається частка (total_discount[:, np.newaxis] / n_features), і від ймовірності береться натуральний логарифм для стабільності.

3.2.1.2 CAWNB

У цій роботі реалізований алгоритм Class-Specific Attribute Weighted Naïve Bayes, описаний у теоретичній частині роботи. Конструктор класу окрім параметру згладжування приймає максимальну кількість ітерацій оптимізації max_iter для функції оптимізації ваг.

В основі реалізації використовується метод оптимізації L-BFGS-B, котрий працює за принципом градієнтного спуска. Він використовує обмежений об’єм пам’яті, що робить його ефективним для задач з великою кількістю параметрів [17].

Метод _neg_log_likelihood – функція логарифмічної правдоподібності, що використовується для пошуку ваг, що максимізують ймовірність правильної класифікації текстів. Для цього функція підраховує логарифм ймовірності для кожного класу, множить його на відповідну вагу, і підраховує суму для усіх ознак.

У методі fit викликається функція minimize з програмного пакету scipy.optimize (Рис. 2):

```
res = minimize(
    fun=self._neg_log_likelihood,
    x0=w0,
    args=(X, y_encoded),
    method='L-BFGS-B',
    jac=True,
    bounds=bounds,
    options={'maxiter': self.max_iter}
)
```

Рисунок 2. Функція minimize у CAWNB

Тут fun – функція для мінімізації; x0 – значення ваг, кожна з яких рівна одиниці на початку; args – аргументи, що передаються у функцію fun, а саме: X – матриця ознак, y_encoded – вектор міток класів; method – метод оптимізації; jac=True – вказує, що функція повертає як значення втрат (loss), так і градієнт; bounds – обмежує ваги значеннями у межах від 0 до 1; в options передається максимальна кількість ітерацій оптимізації.

Метод CAWNB модифікує класичний NB, додаючи кожній ознаці окрему вагу для кожного класу, що дозволяє врахувати різний ступінь важливості цих ознак для різних класів. Хоча швидкість тренування при використанні такого метода суттєво нижча, ніж у класичного відповідника, класифікація на основі натренованої моделі не має вищої обчислювальної складності.

3.2.2 Трансформери у scikit-learn

У процесі попередньої обробки текстових даних в конвеєрі бібліотеки scikit використовуються трансформери – класи, що реалізують методи fit(), transform(), fit_transform(). Їх ціль – перетворення вхідних

даних у зручний формат для подальшої обробки та класифікації. Це можуть бути як базові перетворення, так і більш складні операції, як токенізація, векторизація тексту або видалення шумових термінів

3.2.2.1 Vectorizers

Векторизація – перетворення тексту у числовий формат. У бібліотеці scikit реалізовано два основних векторизатори, що застосовуються у текстовій класифікації: CountVectorizer та TfIdfVectorizer.

CountVectorizer перетворює колекцію текстових документів на матрицю частот слів. Кожне слово у тексті стає окремою ознакою (стовпцем), а значенням є кількість разів, коли це слово зустрічається у відповідному документі.

TfIDfVectorizer є розширеною версією CountVectorizer, яка не тільки враховує частоту слова у документі, а й зменшує вагу слів, що зустрічаються часто у всіх документах. Це досягається за допомогою формули TF-IDF. Формат матриці TfIdfVectorizer аналогічний CountVectorizer, проте значення є не частотами, а вагами TF-IDF.

У розробленому застосунку реалізовано можливість вибору між двома типами векторизаторів – CountVectorizer та TfIdfVectorizer. Користувач може обрати відповідний метод на етапі налаштування пайплайну. Якщо текстова колекція містить багато коротких повідомлень (наприклад, твіти або коментарі), CountVectorizer може працювати краще. Якщо ж повідомлення довші або різняться за довжиною, TfIdfVectorizer допоможе компенсувати вплив надто частих слів та підвищити значимість унікальних слів.

Основні параметри, що використовуються при ініціалізації цих векторизаторів у коді застосунку:

- **analyzer –** об’єкт, що вказує, як саме текст розбивається на токени. Стандартне значення – word, тобто аналізатор розбиває текст на окремі слова. Замість стандартних аналізаторів у vectorizer можна вводити власні реалізації. У застосунку таким чином реалізовано StemTokenizer для стемінгу та LemmaTokenizer для лематизації відповідно.
- **lowercase -** вказує, чи потрібно перетворювати всі символи на нижній регістр перед токенізацією.
- **stop_words –** вказує, які шумові слова потрібно виключити з тексту перед побудовою вектору. Значення ‘english’ використовує стандартний список шумових слів англійської мови.
- **max_df –** визначає верхній поріг частоти слова. Слова, які зустрічаються у більш ніж max_df документів, будуть виключені з векторизації. Це дозволяє усунути занадто часті слова, що не несуть корисної інформації. Наприклад, 0.95 виключить слова, які зустрічаються більш ніж у 95% документів.
- **min_df –** визначає нижній поріг частоти слова. Слова, які зустрічаються менш ніж у min_df документах, будуть виключен з векторизації. Це дозволяє усунути рідкісні слова, що можуть бути шумом. Наприклад, значення 2 виключить усі слова, що з’являються лише в одному документі.
- **ngram_range –** визначає діапазон n-грам для створення ознак. Наприклад, значення (2, 3) включає лише біграми та триграми.

3.2.2.2 Очистка тексту

У розробленому застосунку реалізовано трансформер TextCleaner, що виконує комплексну чистку тексту. Він дозволяє очистити вхідні дані від зайвих символів, нормалізувати їх та зменшити кількість шумових ознак.

TextCleaner реалізовано як клас, що успадковує BaseEstimator та TransformerMixin. Це дозволяє використовувати його у складі конвеєру scikit-learn як звичайний трансформер.

Основні етапи очищення:

- нормалізація емодзі. За допоммогою функції tp.replace.emojis() з бібліотеки textacy всі емодзі замінюються на загальний токен “<emoji>”
- нормалізація посилань. Кожне посилання за регулярним виразом замінюється на токен “<url>”
- нормалізація адрес електронної почтової скриньки. Кожен email за регулярним виразом замінюється на токен “<email>”
- номери телефону замінюються на “<phone>”
- хештеги замінюються на “<hashtag>”
- згадки користувачів замінюються на “<user>”
- усі HTML-теги видаляються, оскільки зазвичай вони не несуть корисної інформації
- усі символи пунктуації видаляються
- усі цифри замінюються на токен “<num>”
- використовується функція tp.remove.accents() для заміни актентованих символів (é, ñ) на їх ASCII-аналоги (e, n).

3.2.2.3 Корекція орфографії

У застосунку реалізовано трансформер SpellCorrector, який здійснює корекцію орфографії на основі бібліотеки SymSpell.

В основі трансформера – алгоритм SymSpell, який використовує попередньо підготовлений частотний словник для швидкого знаходження найбільш ймовірних варіантів виправлення.

На етапі ініціалізації завантажується файл, що містить частотний словник англійської мови, що дозволяє алгоритму знаходити найбільш ймовірні варіанти для виправлення. Для зменшення кількості обчислень використовуються результати попередніх перевірок (self._cache). Якщо слово вже перевірено, його виправлена версія зберігається у кеші. Слова довжиною до двох символів не піддаються корекції. Для кожного токена виконується пошук найближчих варіантів за допомогою self.symspell.lookup(key, Verbosity.CLOSEST). Якщо знайдено кілька варіантів, вибирається найбільш ймовірний (best = suggestions[0].term). Якщо жодного варіанту не знайдено, слово залишається без змін.

3.3 Аналіз ефективності методів

У цьому підрозділі буде проведено експериментальну оцінку ефективності описаних методів, включаючи:

- Методи попередньої обробки тексту, такі як видалення шумових слів, стемінг та врахування n-грам.
- Використання різних типів векторизації.
- Застосування згладжування ймовірностей, таких як згладжування Лапласа, Ліндстоуна та Абсолютне дисконтування.
- Використання зваженого класифікатора CAWNB.

Основною метою цього розділу є визначення оптимальної комбінації методів для покращення роботи наївного байєсівського класифікатора на коротких текстах із текстових повідомлень. Для цього будуть проведені серії експериментів на обраному наборі даних з подальшим аналізом отриманих результатів.

Для оцінки ефективності класифікаторів у даній роботі використовується F1-метрика, оскільки вона є збалансованою мірою між точністю та повнотою.

3.3.1 Набори даних для тестування методів

Для проведення експериментальної оцінки ефективності методів класифікації було обрано три різні корпуси даних, що відрізняються між собою як за структурою даних, так і за типами завдань класифікації. Це дозволяє проаналізувати продуктивність розглянутих методів на текстах різної довжини та рівня шуму.

1. IMDB Sentiment Dataset.

- Кількість текстів: 50000.
- Класи: sentiment (positive, negative).
- Опис: набір даних складається з рецензій на фільми, позначених як позитивні або негативні. Він був обраний через достатньо високу кількість текстів та середню довжину кожного повідомлення. Використання цього набору даних дозволяє перевірити здатність класифікатора працювати з довгими повідомленнями та визначати загальну тональність тексту.

2. Tweets Dataset

- Кількість текстів: 50000
- Класи: 20 авторів
- Опис: Набір даних складається з повідомлень у соціальній мережі X, що були зібрані за певними авторами. Задача класифікації полягає у визначенні автора тексту на основі його стилістичних та лексичних особливостей. Цей набір був обраний через коротку довжину текстів, велику кількість класів та значний рівень шуму, що створює додатковий виклик для класифікації.

3. Reddit Posts Dataset

- Кількість текстів: 7000.
- Класи: 7 великих спільнот.

- Опис: набір даних було зібрано за допомогою бібліотеки praw з соціальної мережі Reddit. Він включає заголовки та тексти постів із семи обраних спільнот. Набір даних є помірно шумним, але тексти у ньому структуровані краще, ніж у Tweets Dataset. Це дозволяє протестувати методи класифікації на коротких, але більш формалізованих повідомленнях, що характерні для соціальних мереж.

Таким чином, використання трьох корпусів даних із різною структурою та довжиною текстів дозволяє оцінити ефективність класифікатора у різних умовах. Далі буде проведено аналіз запропонованих методів класифікації.

3.3.2 Використання різних моделей

Було проведено порівняння F1-метрик для моделей наївного байєсу, доступних у застосунку:

Dataset	**MNB**	**TF-IDF MNB**	**Bernoulli NB**	**NB з AD**	**CAWNB**
Tweets	0.5397	0.5348	0.5383	0.5419	0.4915
Reddit	0.7850	0.7376	0.7871	0.7916	0.7791
IMDB	0.8533	0.8682	0.8617	0.8536	0.8834

На основі отриманих даних можна зробити наступні висновки:

- Використання TF-IDF забезпечує найвищу точність у задачах з довгими текстами, але для коротких текстах його застосування може бути менш ефективним через надмірне зниження ваги слів, що часто зустрічаються.

- Bernoulli NB продемонстрував найкращі результати для Reddit, що свідчить про ефективність бінарного представлення для текстів невеликої довжини та з помірним рівнем шуму.
- MNB показав стабільні результати на усіх наборах даних, проте його ефективність є нижчою за TF-IDF у випадку довгих текстів та нижчою за Bernoulli NB у випадку коротких текстів.
- Використання NB з абсолютним дисконтуванням дозволило підвищити F1-метрику на малих корпусах завдяки адаптивному згладжуванню. З іншого боку, для великого корпусу він показує гірші результати, ніж аналогічні моделі.
- Використання CAWNB на малих корпусах даних показало гірший результат, ніж інші моделі. Скоріш за все, це пов’язано з тим, що у коротких текстах з високим рівнем шуму оптимізація ваг для кожного класу та ознаки може призвести до перенавчання або надмірної чутливості до рідкісних слів. На великому корпусі CAWNB навпаки, демонструє найкращий результат серед усіх методів. Метод оптимізації ваг за допомогою градієнтного спуску дозволяє підвищити точність моделі, особливо у задачах, де класи суттєво відрізняються за словниковим запасом.

Таким чином, вибір оптимальної моделі залежить від специфіки тексту: для коротких повідомлень краще використовувати Bernoulli NB, тоді як для довгих текстів доцільніше застосувати TF-IDF MNB.

3.3.3 Використання n-грам

Далі запропоновано порівняння ефективності у залежності від довжини n-грам для MNB.

Діапазон довжин n-грам	1-1	1-2	1-3	1-4	2-2	2-3
Tweets	0.5397	0.5830	0.5822	0.5779	0.5259	0.5220
Reddit	0.7850	0.7855	0.7768	0.7682	0.6863	0.6742
IDMB	0.8533	0.8865	0.8957	0.8972	0.8888	0.8965

Тут позначення, такі як 2-3 – це діапазон довжин n-грам, наприклад, 2-3 позначає використання n-грам довжини від 2 до 3.

З отриманих даних можна зробити такі висновки щодо впливу використання n-грам різної довжини на ефективність класифікації:

- Для невеликих наборів даних найкращий результат досягається при використанні уніграм та біграм. Це вказує на те, що їх додавання може допомогти краще врахувати контекст. 4-грами знижують показник F1, що вказує на надмірне збільшення розмірності ознак та появи шуму.
- Використання лише біграм також демонструє суттєве зниження точності, що підтверджує важливість уніграм для текстів середньої довжини.
- У випадку текстів IMDB найвищий показник F1 досягається при використанні 4-грам. Це вказує на те, що довгі тексти містять більше зв’язних фраз та послідовностей слів, що дозволяє n-грамам вищої довжини краще враховувати контекст. Проте у цьому випадку така довжина n-грам суттєво впливає на швидкість виконання програми, що робить їх використання практично марним.

3.3.4 Використання стемінгу та лематизації

У цьому підрозділі буде порівняна ефективність методів нормалізації тексту, таких як стемінг і лематизація для MNB.

Набір даних	MNB	MNB+стемінг	MNB+лематизація
Tweets	0.5397	0.5409	0.5426
Reddit	0.7850	0.7980	0.8021
IMDB	0.8533	0.8431	0.8463

На основі отриманих даних можна зробити наступні висновки щодо впливу методів нормалізації:

- На коротких текстах з високою кількістю шуму стемінг незначно покращує ефективність класифікації, зменшуючи розмір словника за рахунок зведення слів до їх кореневих форм. Лематизація демонструє дещо кращий результат.
- У випадку великого набору даних спостерігається дещо інша тенденція. Використання стемінгу та лематизації навіть знижує показник F1-метрики. Це може бути пов’язано з тим, що у великих наборах даних втрата закінчень може призводити до втрати частини семантичного значення.

Загалом, використання стемінгу або лематизації разом з NB доречне тільки у випадках, коли об’єм тренувальних даних недостатній для отримання достатнього словникового запасу. Також методи значно впливають на швидкодію, що є ще одною причиною використовувати їх обачно.

3.3.5 Видалення рідкісних слів

У цьому підрозділі оглянуто доцільність занадто рідкісних слів, з використанням MNB.

Набір даних	1	2	3	4	5	6
Tweets	0.5397	0.5342	0.5306	0.5247	0.5212	0.5164
Reddit	0.7850	0.7940	0.7782	0.7705	0.7673	0.7478
IMDB	0.8533	0.8516	0.8539	0.8520	0.8513	0.8503

Тут цифри у колонках – це мінімальна частота слова у корпусі, за якої слово не видаляється. З результатів видно, що у коротких текстах видалення рідкісних слів може бути небезпечним, оскільки навіть одиничні слова можуть містити важливу інформацію для класифікації. Оптимальним значенням min_df є 2, що дозволяє позбутися випадкових помилок, не втрачаючи інформативних термінів.

3.3.6 Видалення шумових слів

Тут виконано перевірки впливу видалення шумових слів на F1-метрику для MNB.

Набір даних	Без видалення	З видаленням
Tweets	0.5397	0.5303
Reddit	0.7850	0.7231
IMDB	0.8533	0.8598

Дані показують, що видалення шумових слів доцільне лише на великому корпусі даних. У дрібних корпусах їх видалення навпаки, знижує F1-метрику. Для Tweets це може означати, що навіть часті слова можуть бути важливими для класифікації, оскільки вони можуть містити ключову інформацію про стиль автора чи контекст повідомлення.

Видалення таких слів може призвести до втрати інформації та погіршення якості класифікації.

3.3.7 Перевірка правопису

Результати використання перевірки правопису з MNB наступні:

Набір даних	Без перевірки	З перевіркою
Tweets	0.5397	0.5324
Reddit	0.7850	0.7884
IMDB	0.8533	0.8521

Таким чином, з отриманої інформації можна зробити висновок, що використання SymSpell. Використання більш складних алгоритмів на великих корпусах даних недоцільне, оскільки вони мають високу обчислювальну складність, і таким чином, суттєво сповільнюють алгоритм наївного байєса.

3.3.8 Підбір оптимальних параметрів

Дані, отримані з минулих підрозділів, свідчать, що використання різних методів та технік сильно залежить від задачі та тренувального корпусу. Далі запропоновані комбінації параметрів, що пропонують найкращий результат для різних задач:

- Комбінація стемінгу, n-грам довжини 1-2, видалення слів, що зустрічаються лише один раз, та MNB дає хороший результат на коротких корпусах, а саме, 0.5745 в F1 метриці для Tweets та 0.8111 для Reddit. Absolute Discounting за тих самих параметрів сягає значення 0.8335 для Reddit та 0.5716 для Tweets.
- Для великого корпусу добре підходить MNB з TF-IDF, котрий разом з n-грамами від 1 до 3, видаленням рідкісних слів сягає метрики 0.9050 на корпусі даних IMDB. Альтернативно, за умови, коли

швидкість тренування не є критичною, можна використати CAWNB з n-грамами від 1 до 3, F1-метрика якого сягає 0.9169. Тим не менш, оскільки обчислювальна складність такого методу набагато вища за класичний наївний байєс, доцільність використання такого методу є сумнівною. Якщо потрібна вища точність, а жертва часом виконання допустима, краще використовувати більш складні методи, як-от згорткові мережі.

ВИСНОВОК

Робота була спрямована на дослідження ефективності наївного байєсівського класифікатора (NB) для задачі класифікації коротких текстових повідомлень. Основні цілі включали:

- аналіз теоретичних основ та існуючих варіацій NB;
- вивчення сучасних підходів до покращення класифікації текстів, зокрема методів попередньої обробки, згладжування, векторизації, використання ембедінгів та вагових моделей;
- розробку десктопного застосунку з графічним інтерфейсом для експериментів з NB;
- проведення порівняльного тестування на трьох наборах даних.
- У процесі дослідження було реалізовано застосунок на Python із використанням бібліотеки scikit-learn, що дозволяє користувачу гнучко конфігурувати конвеєр обробки текстів. Реалізовано власні варіанти NB, зокрема Absolute Discounting NB та Class-Specific Attribute Weighted NB (CAWNB).

Було отримано такі основні результати:

- Multinomial NB із TF-IDF показав найвищу точність на довших текстах, тоді як Bernoulli NB — на коротких текстах.
- Використання Absolute Discounting покращує результати для малих корпусів, але менш ефективне для великих.
- CAWNB доцільний при класифікації довгих текстів з великою кількістю класів, але чутливий до перенавчання у коротких текстах.
- Врахування біграм та триграм покращує точність класифікації завдяки послабленню гіпотези про незалежність ознак.
- Стемінг і лематизація мають незначний ефект і можуть погіршити якість на великих корпусах.

- Видалення шумових слів і перевірка орфографії не дають стабільного приросту якості.

Отже, оптимальний вибір параметрів NB значною мірою залежить від природи тексту та об’єму даних. Розроблений інструмент дозволяє гнучко підбирати налаштування для досягнення найкращих результатів у конкретних задачах класифікації коротких повідомлень.

СПИСОК ЛІТЕРАТУРИ

1. Bhavani, A., & Kumar, B. S. (2021, April). A review of state art of text classification algorithms. In *2021 5th international conference on computing methodologies and communication (ICCMC)* (pp. 1484-1490). IEEE.
2. Amancio, D. R., Comin, C. H., Casanova, D., Travieso, G., Bruno, O. M., Rodrigues, F. A., & da Fontoura Costa, L. (2014). A systematic comparison of supervised classifiers. *PloS one*, *9*(4), e94137.
3. Singh, G., Kumar, B., Gaur, L., & Tyagi, A. (2019, April). Comparison between multinomial and Bernoulli naïve Bayes for text classification. In *2019 International conference on automation, computational and technology management (ICACTM)* (pp. 593-596). IEEE.
4. Xu, S. (2018). Bayesian Naïve Bayes classifiers to text classification. *Journal of Information Science*, *44*(1), 48-59.
5. Jurafsky, D., & Martin, J. H. (2023). *Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition* (3rd ed., Chapter 4).
6. Zhu, Z., Hiemstra, D., Apers, P., & Wombacher, A. (2013, June). Ut-db: An experimental study on sentiment analysis in twitter. In *Second Joint Conference on Lexical and Computational Semantics (* SEM), Volume 2: Proceedings of the Seventh International Workshop on Semantic Evaluation (SemEval 2013)* (pp. 384-389).
7. Ogada, K., Mwangi, W., & Cheruiyot, W. (2015). N-gram based text categorization method for improved data mining. *Journal of Information Engineering and Applications*, *5*(8), 35-43.
8. Senders, Y. (2021). *The impact of stemming and lemmatization applied to word vector based models in sentiment analysis: A comparison of stemming and lemmatization methods* (Master's thesis). Tilburg

University, School of Humanities and Digital Sciences, Department of Cognitive Science & Artificial Intelligence, Tilburg, The Netherlands.

9. Ahmadi, M., Khajavi, M., Varmaghani, A., Ala, A., Danesh, K., & Javaheri, D. (2025). Leveraging Large Language Models for Cybersecurity: Enhancing SMS Spam Detection with Robust and Context-Aware Text Classification. *arXiv preprint arXiv:2502.11014*.
10. Hashmi F. How to classify text using Word2Vec [Електронний ресурс]. Режим доступу: https://thinkingneuron.com/how-to-classify-text-using-word2vec/ (Дата звернення: 09.05.2025).
11. Oancea, B. Text classification using machine learning methods. In *KNOWCON 2023: Knowledge on Economics and Management: Conference Proceedings* (p. 137). Palacký University Olomouc.
12. Yuan, Q., Cong, G., & Thalmann, N. M. (2012, April). Enhancing naive bayes with various smoothing methods for short text classification. In *Proceedings of the 21st international conference on world wide web* (pp. 645-646).
13. Zhang, H., & Sheng, S. (2004, November). Learning weighted naive Bayes with accurate ranking. In *Fourth IEEE International Conference on Data Mining (ICDM'04)* (pp. 567-570). IEEE.
14. Jiang, L., Zhang, L., Yu, L., & Wang, D. (2019). Class-specific attribute weighted naive Bayes. *Pattern recognition*, *88*, 321-330.
15. Zhou, X., Wu, D., You, Z., Wu, D., Ye, N., & Zhang, L. (2022). Adaptive Two-Index Fusion Attribute-Weighted Naive Bayes. *Electronics*, *11*(19), 3126.
16. Ruan, S., Chen, B., Song, K., & Li, H. (2022). Weighted naïve Bayes text classification algorithm based on improved distance correlation coefficient. *Neural Computing and Applications*, 1-10.
17. Zhu, C., Byrd, R. H., Lu, P., & Nocedal, J. (1997). Algorithm 778: L-BFGS-B: Fortran subroutines for large-scale bound-constrained

optimization. *ACM Transactions on mathematical software (TOMS)*, *23*(4), 550-560.