**ДЕРЖАВНИЙ УНІВЕРСИТЕТ**
**ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ**
**НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ**
**КАФЕДРА ШТУЧНОГО ІНТЕЛЕКТУ**

# КВАЛІФІКАЦІЙНА РОБОТА

на тему: «Система автоматичного виявлення небажаних повідомлень у соціальних мережах з використанням технік машинного навчання»

на здобуття освітнього ступеня бакалавра
зі спеціальності 122 Комп'ютерні науки

освітньо-професійної програми Штучний інтелект

*Кваліфікаційна робота містить результати власних досліджень. Використання ідей, результатів і текстів інших авторів мають посилання на відповідне джерело*

_____ Євгеній ЗЛЕНКО
*(підпис)*

Виконав: здобувач вищої освіти група ШІД-41
Євгеній ЗЛЕНКО

Керівник: Антон ШАНТИР
к.т.н. доцент

Рецензент:
*науковий ступінь, вчене звання* _____
Ім'я, ПРІЗВИЩЕ

**Київ 2025**

**ДЕРЖАВНИЙ УНІВЕРСИТЕТ**
**ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ**

**Навчально-науковий інститут інформаційних технологій**

Кафедра Штучного інтелекту

Ступінь вищої освіти Бакалавр

Спеціальність Комп'ютерні науки

Освітньо-професійна програма Штучний інтелект

**ЗАТВЕРДЖУЮ**
Завідувач кафедрою Штучного інтелекту

_____ Ольга ЗІНЧЕНКО
«_____» _____ 2025 р.

**ЗАВДАННЯ**
**НА КВАЛІФІКАЦІЙНУ РОБОТУ**

Зленко Євгенію Сергійовичу

1. Тема кваліфікаційної роботи: Система автоматичного виявлення небажаних повідомлень у соціальних мережах з використанням технік машинного навчання
керівник кваліфікаційної роботи Антон ШАНТИР, к.т.н. доцент,
затверджені наказом Державного університету інформаційно-комунікаційних технологій від «24» лютого 2025 р. № 56
2. Строк подання кваліфікаційної роботи «23» травня 2025 р.
3. Вихідні дані до кваліфікаційної роботи: приклади реалізації систем виявлення спаму у соціальних мережах, науково-технічна література.
4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)

Аналіз видів небажаних повідомлень у соціальних мережах

Дослідження алгоритмів машинного навчання для виявлення спаму

Розроблення рекомендацій щодо можливостей впровадження у платформи соціальних мереж

5. Перелік графічного матеріалу: *презентація.*
6. Дата видачі завдання «25» лютого 2025 р.

# КАЛЕНДАРНИЙ ПЛАН

| № з/п | Назва етапів кваліфікаційної роботи | Строк виконання етапів роботи | Примітка |
|---|---|---|---|
| 1 | Визначення типів небажаних повідомлень у соціальних мережах | 15.03 - 31.03.2025 | Виконано |
| 2 | Аналіз наукової та технічної літератури з машинного навчання та NLP | 31.03 - 10.04.2025 | Виконано |
| 3 | Вибір алгоритмів та інструментів для побудови моделі класифікації | 10.04 - 25.04.2025 | Виконано |
| 4 | Формування датасету та підготовка навчальної вибірки | 25.04 - 10.05.2025 | Виконано |
| 5 | Розробка та тренування моделі виявлення небажаних повідомлень | 10.05 - 25.05.2025 | Виконано |
| 6 | Тестування системи та аналіз результатів | 25.05 - 28.05.2025 | Виконано |
| 7 | Підготовка звіту та доповіді до захисту | 28.05 - 31.05.2025 | Виконано |

Здобувач вищої освіти _____ Євгеній ЗЛЕНКО
*(підпис)*

Керівник
кваліфікаційної роботи _____ Антон ШАНТИР, к.т.н. доцент
*(підпис)*

## РЕФЕРАТ

Текстова частина бакалаврської роботи: 63 сторінка, 40 рисунків, 39 джерел.

*Об'єкт дослідження* – процеси розпізнавання та фільтрації небажаних текстових повідомлень у соціальних мережах.

*Предмет дослідження* – алгоритми машинного навчання для виявлення небажаного контенту, побудова моделей класифікації повідомлень, інтеграція в системи моніторингу соціальних платформ.

*Мета роботи* – зробити та реалізувати ефективну систему виявлення небажаних повідомлень у соціальних мережах з використанням сучасних технік машинного навчання, а також провести оцінку її ефективності на основі експериментального тестування.

*Методи дослідження*:

1. Аналіз підходів до автоматичного розпізнавання текстів – вивчення нейронних мереж, LSTM-моделей, рекурентних архітектур.
2. Формування навчального датасету – обробка вхідних повідомлень, розмітка класів (спам/не спам).
3. Побудова моделі виявлення спаму – проєктування та тренування класифікатора з використанням Python та бібліотек машинного навчання.
4. Тестування моделі – оцінка точності, recall, precision, побудова кривих ROC та аналіз результатів на практичних прикладах.

*Галузь використання* – інформаційна безпека, автоматизація модерації контенту, цифрова фільтрація в соціальних платформах.

КЛЮЧОВІ СЛОВА: МАШИННЕ НАВЧАННЯ, НЕБАЖАНІ ПОВІДОМЛЕННЯ, СОЦІАЛЬНІ МЕРЕЖІ, LSTM, КЛАСИФІКАЦІЯ ТЕКСТУ, СПАМ-ФІЛЬТР, АВТОМАТИЧНЕ ВИЯВЛЕННЯ, НЕЙРОННА МЕРЕЖА.

**ЗМІСТ**

# ВСТУП

*Актуальність дослідження.* У сучасному цифровому суспільстві соціальні мережі відіграють важливу роль у комунікації, поширенні інформації та формуванні громадської думки. Разом із ростом популярності таких платформ як Facebook, Twitter, Instagram та TikTok зростає й кількість небажаних повідомлень – спаму, фейкових новин, агресивного контенту, шахрайських схем. Це створює серйозні загрози для користувачів, знижує довіру до платформ та ускладнює модерацію контенту вручну.

Традиційні методи фільтрації, засновані на словниках і правилах, виявляються недостатньо ефективними в умовах великої кількості даних та швидкої еволюції мовних конструкцій. У зв'язку з цим зростає потреба у використанні сучасних технік машинного навчання, зокрема глибоких нейронних мереж та методів обробки природної мови (NLP), які дозволяють автоматично і точно виявляти небажані повідомлення в режимі реального часу.

Дослідження в цій галузі є актуальним як з точки зору підвищення інформаційної безпеки, так і для створення ефективних інструментів модерації, що можуть бути інтегровані в інфраструктуру соціальних мереж. Це дозволяє покращити якість онлайн-комунікації, захистити користувачів від шкідливого контенту та зменшити навантаження на служби підтримки платформ.

*Мета роботи* – озробити та реалізувати ефективну систему виявлення небажаних повідомлень у соціальних мережах з використанням сучасних технік машинного навчання, а також провести оцінку її ефективності на основі експериментального тестування.

*Об'єкт дослідження* – процеси розпізнавання та фільтрації небажаних текстових повідомлень у соціальних мережах.

*Предмет дослідження* – алгоритми машинного навчання для виявлення небажаного контенту, побудова моделей класифікації повідомлень, інтеграція в системи моніторингу соціальних платформ.

# 1 ТЕХНІКИ МАШИННОГО НАВЧАННЯ ДЛЯ ОБРОБКИ ТЕКСТОВИХ ПОВІДОМЛЕНЬ

## 1.1 Рекурентні нейронні мережі як інструмент аналізу тексту

Нейронна мережа – це математична модель, яка імітує принципи роботи людського мозку для розв’язання різноманітних задач, зокрема класифікації, прогнозування та розпізнавання образів. Вона складається з великої кількості взаємопов’язаних елементів – нейронів, які об'єднані в шари: вхідний, приховані та вихідний [1].

Кожен нейрон отримує на вході певні дані, зважує їх (множить на коефіцієнти – ваги), підсумовує та передає далі через активаційну функцію, яка визначає, чи варто сигнал передавати на наступний рівень. Завдяки цим вагам і здатності змінювати їх під час навчання, нейронна мережа може «вивчати» закономірності в даних та адаптуватися до нової інформації.

Нейронні мережі є основою багатьох сучасних методів машинного навчання, зокрема глибокого навчання (Deep Learning), і застосовуються для:

- обробки зображень і відео,
- розпізнавання мови,
- аналізу текстів (NLP),
- рекомендаційних систем,
- виявлення шахрайства та кіберзагроз.

Однією з архітектур у галузі машинного навчання, що активно використовується для обробки послідовних даних, зокрема текстової інформації, є рекурентні нейронні мережі (Recurrent Neural Networks, RNN). На відміну від класичних багатошарових перцептронів, які передають інформацію лише в одному напрямку – від входу до виходу, рекурентні нейронні мережі мають зворотні зв'язки, що дозволяє їм зберігати інформацію про попередні стани нейронів. Завдяки цьому RNN здатні ефективно враховувати контекст попередніх елементів при обробці поточних вхідних даних.

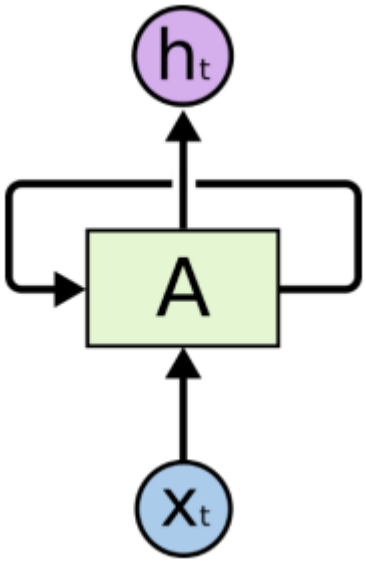



Рисунок 1.1 Схематичне зображення рекурентної нейронної мережі (RNN)

Основною особливістю RNN є наявність прихованого стану (hidden state), який акумулює інформацію про оброблену послідовність на кожному кроці. Кожен елемент вхідної послідовності впливає не лише на поточний вихід мережі, але й на стан, що передається до наступного етапу обробки. Це забезпечує можливість моделі аналізувати залежності між елементами послідовності, що є критично важливим для задач, де порядок слів або символів відіграє значну роль [1].

Рекурентні нейронні мережі широко застосовуються для вирішення таких задач, як:

- класифікація текстових повідомлень, зокрема виявлення спаму та токсичного контенту;
- автоматичне розпізнавання та синтез мовлення;
- машинний переклад;
- аналіз часових рядів;
- генерація тексту, музики або відеопослідовностей.

Проте класичні RNN мають низку обмежень, зокрема схильність до проблеми затухаючих та вибухаючих градієнтів, що ускладнює навчання моделей на довгих послідовностях. Для подолання цих недоліків було розроблено модифікації архітектури, серед яких найбільш відомими є мережі довготривалої

короткочасної пам’яті (Long Short-Term Memory, LSTM) та блоки з керованими рекурентними одиницями (Gated Recurrent Units, GRU). Ці моделі дозволяють ефективніше зберігати інформацію про важливі елементи послідовності протягом більш тривалого часу [2].

Рекурентні нейронні мережі є їх здатність опрацьовувати послідовні дані, де порядок елементів має принципове значення. Така властивість дозволяє використовувати RNN для аналізу текстів, мовлення, відеопослідовностей, часових рядів та інших типів інформації, що подаються у вигляді впорядкованих наборів елементів.

Елементом архітектури RNN є циклічний механізм передачі інформації між кроками обробки. На кожному кроці нейронна мережа отримує на вхід не лише поточний елемент послідовності, але й прихований стан, що зберігає інформацію про попередні елементи. Цей прихований стан діє як своєрідна пам’ять, яка дає змогу мережі враховувати контекст, накопичений під час обробки попередніх елементів [3 ].

Рекурентні нейронні мережі функціонують шляхом поетапної обробки кожного елемента вхідної послідовності, зберігаючи інформацію про попередні стани за допомогою механізму прихованого стану. На відміну від звичайних нейронних мереж, які працюють із фіксованими наборами вхідних даних, RNN мають можливість передавати накопичену інформацію від одного кроку до іншого, що дозволяє моделі враховувати контекст попередніх елементів під час прийняття рішень щодо поточного елемента.

Процес обробки послідовності в RNN можна уявити у вигляді розгортання мережі в часовому просторі. Як зображено на рис. 1.2, початковий вхідний елемент обробляється першим блоком, який передає прихований стан до наступного блоку, що отримує наступний елемент, і так далі. Такий підхід дозволяє нейронній мережі накопичувати інформацію про попередні кроки, що є важливим при роботі з текстами, де значення слова визначається не лише самим словом, а й його оточенням у реченні.

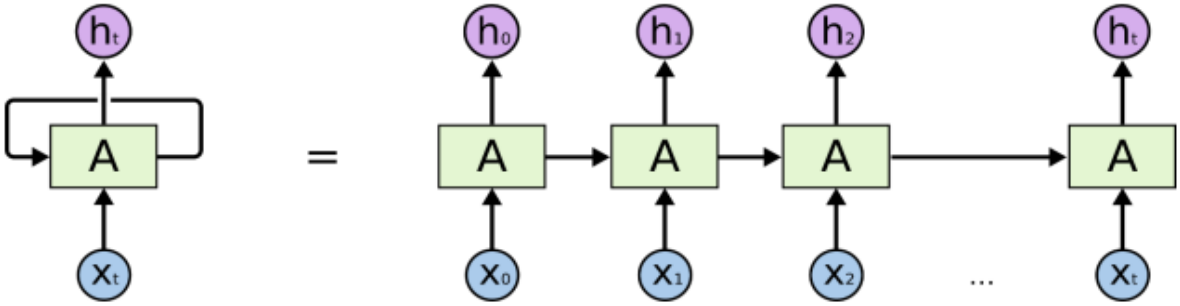



Рисунок 1.2 – Розгортання рекурентної нейронної мережі у часовому просторі

У випадку роботи з текстовими повідомленнями така властивість має принципове значення, оскільки зміст речення або фрази залежить не лише від окремих слів, але й від їхнього порядку та взаємозв'язку. Наприклад, різниця між твердженнями *«Не схвалюю повідомлення»* та *«Схвалюю повідомлення»* визначається одним словом, і правильне розуміння значення вимагає обліку контексту всієї послідовності[4].

При поданні тексту на вхід RNN, кожне слово або символ обробляється поетапно, із збереженням проміжної інформації про попередні кроки. Це дозволяє моделі:

– визначати, які саме попередні елементи мають вагоме значення для поточного рішення;
– враховувати граматичні та семантичні залежності;
– формувати більш точні класифікаційні висновки, ґрунтуючись на повному контексті.

Для забезпечення якісної роботи з текстовими даними часто застосовуються додаткові техніки підготовки вхідної інформації, такі як токенізація (розбиття тексту на окремі слова або символи), лематизація (приведення слів до початкової форми), а також векторизація (перетворення слів у числові вектори, які можна обробляти нейронною мережею).

Завдяки здатності зберігати та обробляти контекст послідовностей, рекурентні нейронні мережі широко застосовуються для реалізації задач аналізу природної мови, включаючи класифікацію повідомлень, розпізнавання спаму,

виявлення токсичного контенту, що обґрунтовує їх вибір для створення системи автоматичного виявлення небажаних повідомлень у соціальних мережах.

### 1.2 Мережі LSTM у задачах класифікації повідомлень

Одним із найбільш ефективних рішень для роботи з послідовними даними є мережі з довготривалою короткочасною пам'яттю (*Long Short-Term Memory*, LSTM), що належать до класу рекурентних нейронних мереж. LSTM-мережі були запропоновані у 1997 році Зеппом Гохрайтером (*Sepp Hochreiter*) та Юргеном Шмідхубером (*Jürgen Schmidhuber*) як вдосконалення класичних RNN для подолання їхніх основних недоліків, пов'язаних із втратою інформації на великих відстанях у послідовностях [5].

Основна проблема стандартних RNN полягає в затуханні або вибуханні градієнтів під час зворотного поширення похибки при навчанні на довгих послідовностях. Це призводить до того, що мережа не здатна ефективно запам'ятовувати інформацію, яка знаходиться на значній відстані від поточного кроку обробки. Як наслідок, стандартні RNN мають обмежені можливості для моделювання довготривалих залежностей у даних.

На відміну від звичайних рекурентних мереж, архітектура LSTM включає спеціальні елементи – комірки пам'яті (memory cells), які здатні зберігати інформацію протягом тривалого часу. Управління потоком інформації в LSTM здійснюється за допомогою тристадійного механізму керуючих "воріт" (gates):

- вхідні ворота (input gate) – регулюють, яка частина нової інформації потрапить до комірки пам'яті;
- забуваючі ворота (forget gate) – визначають, яка частина попередньої інформації буде збережена, а яка – видалена;
- вихідні ворота (output gate) – контролюють, яка частина інформації з комірки пам'яті буде передана на вихід мережі [6].

Такий підхід дозволяє LSTM-моделям ефективно навчатися як на коротких, так і на довгих послідовностях, забезпечуючи здатність зберігати

контекст навіть на віддалених кроках, що є критично важливим для обробки природної мови, зокрема в задачах класифікації текстових повідомлень.

Основні переваги LSTM над класичними RNN:

- можливість зберігати довготривалі залежності завдяки механізму воріт;
- стійкість до проблеми затухання градієнтів;
- покращена здатність до навчання на складних та великих текстових корпусах;
- висока ефективність у задачах генерації та класифікації текстів, розпізнавання мовлення, машинного перекладу [7].

У контексті розв'язання задачі автоматичного виявлення небажаних повідомлень у соціальних мережах застосування LSTM є доцільним вибором, оскільки ця архітектура здатна враховувати складні залежності між словами та фразами в повідомленнях, що підвищує точність класифікації контенту.

Для покращення здатності зберігати інформацію на довгих часових відрізках у послідовностях рекурентні нейронні мережі використовують спеціальні архітектури, такі як LSTM. Основною відмінністю цих мереж від класичних RNN є наявність механізму пам’яті та воріт, які контролюють потік інформації всередині кожної комірки. При цьому важливим етапом є передача прихованого стану та активація за допомогою нелінійної функції, що забезпечує стабільність навчання та запобігає затуханню градієнтів [8]. Схематичне зображення принципу роботи такої моделі наведено на рис. 1.3.

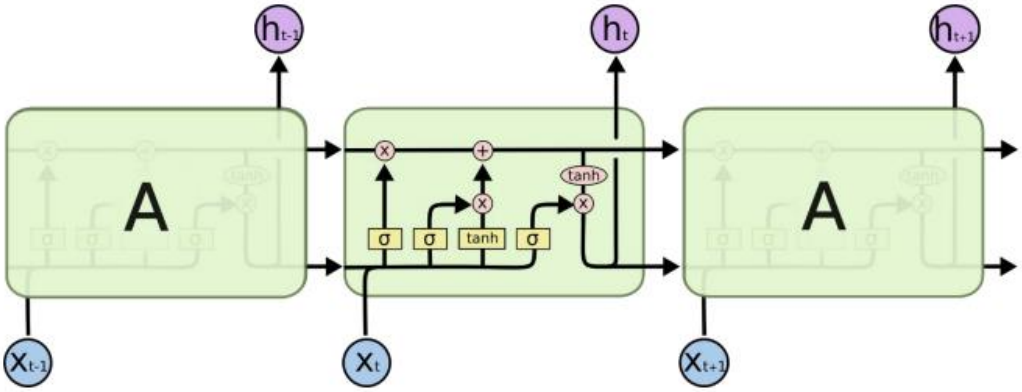



Рисунок 1.3 Схематичне представлення принципу роботи рекурентної нейронної мережі з використанням механізму пам’яті

Як було зазначено раніше, мережі LSTM мають специфічну архітектуру, яка відрізняється від класичних рекурентних нейронних мереж тим, що повторюваний модуль складається не з одного елемента, а з декількох взаємопов’язаних підкомпонентів. У рамках цього модуля реалізується складний механізм обробки інформації, що дає змогу ефективно контролювати потік даних через нейронну мережу.

На рис. 1.4 зображено принцип роботи такої структури. Вектор, що передається від одного етапу обробки до наступного, проходить через низку логічних операцій, які відповідають за фільтрацію та трансформацію інформації. Кожен блок має свою функцію: збереження, оновлення чи передача інформації далі. Елементи у вигляді кіл символізують виконання математичних дій (наприклад, множення або додавання), тоді як прямокутники вказують на застосування активаційних функцій (наприклад, сигмоїдальної функції) [9].

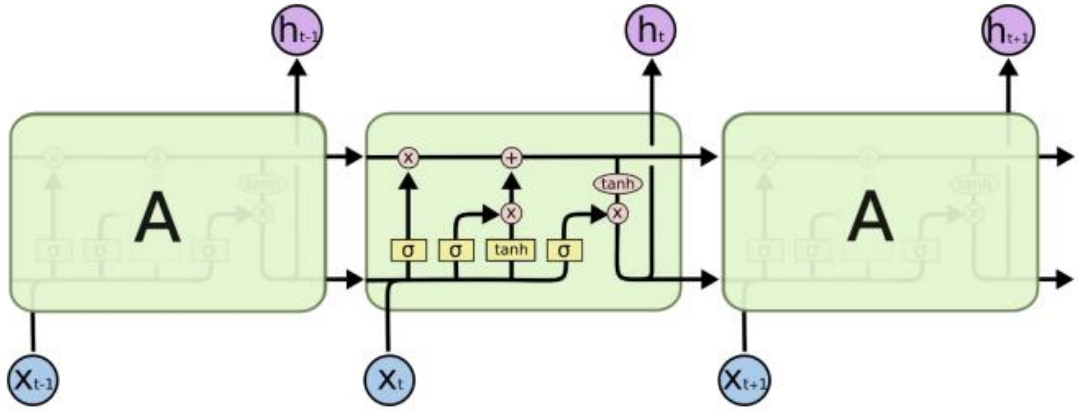



Рисунок 1.4 – Структура LSTM-мережі з використанням механізму керування пам’яттю

Стрілки, що розгалужуються та зливаються, відображають маршрути передачі інформації між різними частинами модуля. Ці зв’язки визначають, яка саме інформація буде передана на вихід, а яка – відфільтрована або модифікована на попередніх етапах. Таким чином, така архітектура дозволяє мережі гнучко управляти пам’яттю та підтримувати необхідний контекст для ефективного аналізу послідовностей.

Архітектури LSTM це стан комірки пам’яті, який виконує роль основного каналу передачі інформації протягом обробки всієї послідовності. Цей стан представлений у вигляді горизонтальної лінії, що проходить через верхню частину схеми та забезпечує збереження інформації від одного такту до іншого. Завдяки спеціальним механізмам, таким як забуваючі та вхідні ворота, відбувається регулювання обсягу інформації, яка зберігається або відкидається на кожному етапі. Дані можуть транспортуватися через усю мережу практично без змін або із незначними лінійними модифікаціями [9]. Така структура забезпечує можливість зберігати важливі відомості на довгих часових інтервалах, що дозволяє ефективно працювати з послідовностями, де присутні далекі залежності між елементами.

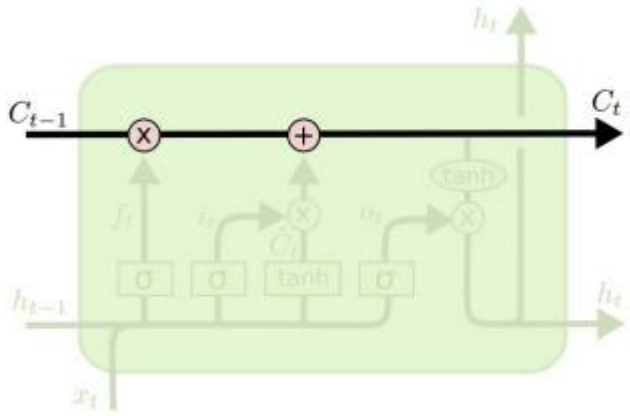



Рисунок 1.5 – Стан комірки пам’яті в архітектурі LSTM

Застосування архітектури Long Short-Term Memory (LSTM) у задачах обробки природної мови (Natural Language Processing, NLP) демонструє високу ефективність при вирішенні завдань класифікації текстів, зокрема для виявлення спаму, фішингових атак, агресивних та токсичних повідомлень. Завдяки здатності зберігати довготривалі залежності та враховувати контекст попередніх елементів послідовності, LSTM-моделі забезпечують значно кращі результати порівняно з традиційними методами класифікації, які базуються лише на статичних ознаках (наприклад, словникових підходах або наївному баєсівському класифікаторі [10]).

У задачах розпізнавання спаму LSTM використовується для аналізу послідовності слів у текстових повідомленнях, що дозволяє виявити приховані закономірності, які характерні для спамових листів, навіть якщо вони маскуються

під звичайне повідомлення. Наприклад, система може розпізнати повторювані шаблони закликів до дії, використання нав’язливих фраз на кшталт «безкоштовно», «виграй приз» або нехарактерних для живої мови конструкцій.

У випадку виявлення токсичних повідомлень (hate speech, abusive language), LSTM-моделі дозволяють враховувати не лише окремі образливі слова, а й загальну інтонацію повідомлення, що формується в результаті взаємодії слів у реченні. Наприклад, повідомлення може не містити відвертих образ, однак виявляти агресію через певні словосполучення або контекст, що важко ідентифікувати за допомогою простих методів фільтрації.

Практичні приклади застосування LSTM для цих цілей:

– Фільтрація спаму в електронній пошті: компанії, які займаються обслуговуванням електронної пошти (наприклад, Gmail), використовують моделі LSTM для вдосконалення точності розпізнавання спаму, зменшуючи кількість хибнопозитивних і хибнонегативних спрацьовувань.
– Автоматична модерація коментарів на платформах соціальних мереж: Facebook, YouTube та інші сервіси інтегрують LSTM у системи модерації для блокування образливого чи токсичного контенту [11].
– Захист месенджерів і чат-ботів: системи, які інтегрують LSTM, здатні в режимі реального часу аналізувати повідомлення користувачів і попереджати про порушення політик комунікації.

Завдяки високій здатності до узагальнення та адаптивності до різних мовних конструкцій, моделі на основі LSTM є ефективним рішенням для створення систем автоматичного виявлення небажаного контенту, що підтверджується численними дослідженнями та практичними реалізаціями в галузі обробки текстів.

### 1.3 Поняття небажаних повідомлень у соціальних мережах

Небажані повідомлення – це текстові, мультимедійні або змішані інформаційні матеріали, які надсилаються користувачам без їхнього запиту чи

згоди та мають на меті нав'язування інформації, маніпуляцію свідомістю, поширення дезінформації або здійснення шахрайських дій. До таких повідомлень відносять, зокрема, спам, фішинг, шахрайські пропозиції, токсичний контент, а також агресивну або образливу мову, що може завдавати шкоди емоційному стану користувачів або порушувати етичні та правові норми.

Особливістю небажаних повідомлень є те, що вони зазвичай розповсюджуються масово або навмисно спрямовуються на певну цільову аудиторію з використанням автоматизованих інструментів, ботів чи спеціальних скриптів. Такий контент може бути призначений для:

– реклами товарів і послуг без дозволу користувачів;
– введення в оману з метою отримання персональних даних (фішинг);
– поширення неправдивої інформації або маніпуляційної пропаганди;
– ображання, цькування або дискримінації окремих осіб чи груп.

У сучасних соціальних мережах і месенджерах небажані повідомлення становлять серйозну проблему, оскільки ускладнюють користувачам отримання достовірної інформації, підвищують ризик кібершахрайства та негативно впливають на рівень інформаційної безпеки загалом. Саме тому актуальним є завдання створення систем автоматичного виявлення такого контенту із застосуванням методів машинного навчання та обробки природної мови [11].

У межах інформаційного простору небажані повідомлення можуть набувати різних форм залежно від мети їх поширення, способів впливу на одержувачів та характеру вмісту. Серед основних типів небажаного контенту, що є предметом дослідження у сфері автоматичного виявлення повідомлень, виділяють такі категорії:

Спам – це масова розсилка небажаних повідомлень, яка зазвичай має рекламний характер та здійснюється без попередньої згоди отримувачів. Такі повідомлення часто містять посилання на зовнішні ресурси, що мають на меті просування товарів, послуг або інтернет-ресурсів. Характерною ознакою спаму є його нав'язливість та використання автоматизованих систем для доставки великої кількості однотипних повідомлень.

Приклади:

– Реклама сумнівних фінансових схем;
– Масові запрошення взяти участь у виграшах або акціях;
– Розсилки комерційного контенту без підписки користувача.

Токсичний контент включає повідомлення, що містять образливі висловлювання, дискримінаційні твердження, погрози, мову ворожнечі або приниження людської гідності. Такий контент може бути спрямований як на окремих осіб, так і на соціальні, етнічні, релігійні групи.

Приклади:

– Образи, приниження гідності;
– Дискримінаційні або сексистські висловлювання;
– Мова ворожнечі щодо расових, національних чи релігійних спільнот.

Шахрайські повідомлення (fraudulent messages) – цей тип охоплює повідомлення, що мають на меті обман одержувачів з метою отримання конфіденційної інформації, фінансових коштів або інших вигод. До цієї категорії відносяться фішингові атаки, повідомлення про вигадані виграші або прохання надати персональні дані.

Приклади:

– Фальшиві повідомлення від нібито офіційних установ (банків, поштових служб);
– Запити про передачу паролів або даних банківських карток;
– Пропозиції інвестувати у неіснуючі фінансові проєкти.

Фейкові новини – це дезінформаційні повідомлення, що поширюються з метою свідомого викривлення фактів, маніпуляції громадською думкою або створення соціальної напруги. Часто такі повідомлення мають сенсаційний характер, ґрунтуються на вигаданих подіях або свідомо спотвореній інформації.

Приклади:

– Поширення неправдивої інформації про політичні події;
– Маніпулятивні статті, що дискредитують публічних осіб;
– Неправдиві відомості про надзвичайні ситуації, епідемії.

Таблиця 1.1

Приклади класифікації повідомлень моделлю LSTM

| № | Текст повідомлення | Клас |
|---|---|---|
| 1 | Вітаємо! Ви виграли айфон. Перейдіть за посиланням для отримання призу | Спам |
| 2 | Ти абсолютно нікчемна людина, краще б тебе тут не було | Токсичне повідомлення |
| 3 | Шановний клієнте, ваше замовлення було відправлене та скоро надійде | Нормальне |
| 4 | Терміново! Поповніть рахунок, щоб уникнути блокування вашої картки | Шахрайське повідомлення |
| 5 | Привіт! Як справи? Чи зручно зараз говорити? | Нормальне |
| 6 | Не пропусти шанс! Інвестуй у криптовалюту та зароби 1000% за тиждень | Спам |
| 7 | Усі люди твоєї нації – просто сміття | Токсичне повідомлення |
| 8 | Доброго дня! Ваш кабінет успішно активовано. Дякуємо, що обрали нас | Нормальне |

### 1.4 Методи виявлення та протидії небажаному контенту

Sender Policy Framework (SPF) – це механізм автентифікації електронної пошти, який дозволяє отримувачу перевірити, чи має сервер, що надсилає повідомлення, дозвіл здійснювати розсилку від імені певного домену. Основна мета SPF – запобігти підробці адрес відправника (спуфінгу) та зменшити ймовірність поширення спаму, фішингових листів і шахрайських повідомлень.

Принцип роботи SPF полягає в тому, що власник доменного імені визначає в DNS-записах перелік IP-адрес, яким дозволено надсилати електронну пошту з цього домену. Коли поштовий сервер отримує повідомлення, він перевіряє IP-адресу відправника та співставляє її з переліком, вказаним у SPF-записі домену. Якщо адреса не входить до дозволеного списку, повідомлення може бути позначене як підозріле або відхилене [12].

Основні етапи роботи SPF:

– відправник надсилає електронний лист із певного IP-адресу;
– сервер-отримувач перевіряє SPF-запис у DNS відповідного домену;

- визначається, чи має відправник право надсилати листи від імені цього домену;
- у разі невідповідності лист може бути відхилено або марковано як спам.

Переваги використання SPF:

- зниження ризику розповсюдження фішингових листів і підроблених повідомлень;
- підвищення довіри до електронної пошти від офіційних доменів;
- полегшення фільтрації небажаного контенту на рівні поштових серверів.

SPF має певні обмеження, зокрема:

- він не захищає вміст листа, а лише підтверджує джерело відправлення;
- у разі пересилання повідомлень (forwarding) може виникати некоректна оцінка SPF;
- для підвищення ефективності SPF часто використовується разом із іншими технологіями автентифікації, такими як DKIM (DomainKeys Identified Mail) та DMARC (Domain-based Message Authentication, Reporting and Conformance).

Отже, SPF є одним із базових інструментів у системі протидії небажаному контенту, забезпечує перевірку достовірності відправника та зменшує ймовірність розповсюдження шкідливих або небажаних повідомлень [13].

На рис. 1.6 зображено принцип роботи механізму автентифікації електронної пошти за допомогою Sender Policy Framework (SPF). Процес перевірки складається з кількох основних етапів:

1. Надсилання повідомлення: відправник (Sender) надсилає електронний лист на поштовий сервер одержувача (Inbound Mail Server).
2. Отримання повідомлення сервером одержувача: вхідний поштовий сервер приймає повідомлення та ініціює перевірку SPF.
3. Перевірка SPF-запису через DNS: сервер звертається до системи доменних імен (DNS) для отримання SPF-запису домену відправника, де вказано перелік IP-адрес, яким дозволено надсилати пошту від імені цього домену.

4. Прийняття рішення щодо автентичності повідомлення: якщо IP-адреса відправника співпадає з IP-адресами в SPF-записі, повідомлення вважається автентичним (Authenticated) і доставляється одержувачу. Якщо IP-адреса не входить до дозволеного списку, лист позначається як неавтентичний (Not Authenticated) та може бути заблокований або переміщений до спам-папки [14].

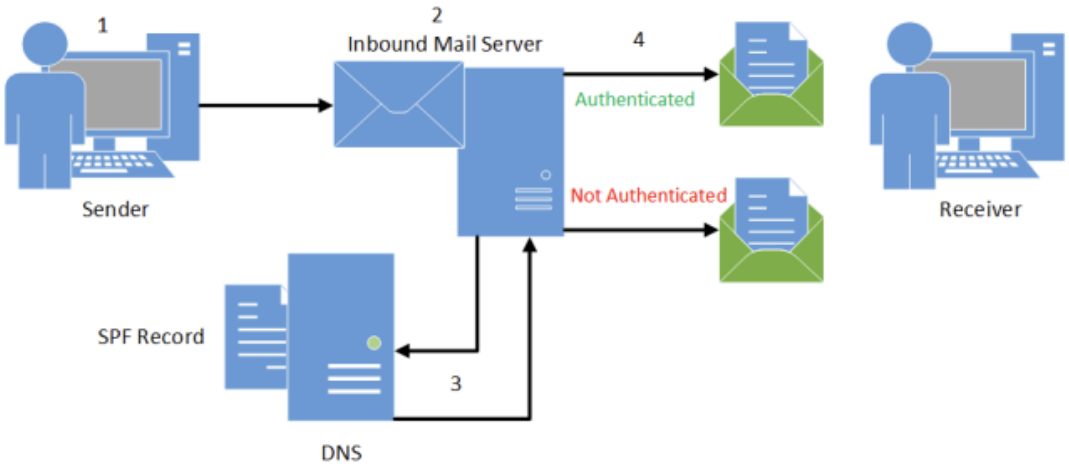



Рис. 1.6 Принцип роботи механізму автентифікації електронної пошти за SPF

SPF механізм допомагає ефективно виявляти підроблені листи, сприяючи зниженню кількості фішингових атак і небажаних повідомлень.

Окрім SPF, до сучасних засобів автентифікації електронної пошти, які відіграють важливу роль у виявленні та протидії небажаному контенту, належать також технології DKIM та DMARC. Їх використання дозволяє значно підвищити рівень захищеності електронного листування, мінімізуючи ризики підробки адрес відправника, розповсюдження фішингових листів та іншого шахрайського контенту.

DomainKeys Identified Mail (DKIM) є технологією цифрового підпису повідомлень, яка дозволяє отримувачу перевірити, що електронний лист дійсно був надісланий з авторизованого серверу від імені вказаного домену та що його вміст не було змінено під час передачі. Відправник створює пару криптографічних ключів: відкритий ключ розміщується в DNS-записах домену, а закритий використовується для генерації цифрового підпису, що додається до заголовка кожного електронного листа. При отриманні повідомлення поштовий

сервер-отримувач може, використовуючи відкритий ключ, перевірити дійсність підпису та цілісність повідомлення. Якщо підпис не відповідає, це свідчить про можливу підробку або зміну листа в процесі доставки. DKIM забезпечує високий рівень довіри до повідомлень, оскільки дозволяє підтвердити не лише факт надсилання від імені певного домену, а й достовірність самого вмісту листа.

Для узгодження роботи SPF та DKIM, а також для забезпечення єдиної політики автентифікації електронної пошти використовується технологія Domain-based Message Authentication, Reporting and Conformance (DMARC). DMARC є надбудовою над SPF та DKIM і дозволяє власнику домену задати чіткі правила того, як слід обробляти повідомлення, які не проходять автентифікацію за цими механізмами. DMARC також забезпечує можливість зворотного зв'язку через спеціальні звіти, які надсилаються власнику домену про те, які листи пройшли або не пройшли перевірку автентичності. Це дозволяє власникам доменів оперативно реагувати на спроби підробки їхнього домену та налаштовувати політики відповідно до реальної ситуації. Політика DMARC може передбачати одну з кількох дій: дозволити доставку листа, помістити його в спам, або відхилити повідомлення повністю, якщо воно не відповідає вимогам SPF чи DKIM.

Інтеграція SPF, DKIM та DMARC дозволяє створити багаторівневу систему захисту від підроблених електронних повідомлень, де кожен з компонентів виконує свою функцію: SPF перевіряє, чи має право сервер надсилати листи від імені домену, DKIM гарантує цілісність та автентичність вмісту повідомлення, а DMARC визначає політику дій у разі невідповідності автентифікаційних перевірок та забезпечує контроль за процесом автентифікації. Такий комплексний підхід значно знижує ймовірність потрапляння шкідливого або небажаного контенту до одержувачів, що є надзвичайно важливим для підтримки безпеки інформаційного простору [15].

## 1.5 Аналіз сучасних фільтрів та алгоритмів класифікації

Для вирішення завдань автоматичного виявлення небажаних повідомлень та класифікації текстового контенту використовуються різноманітні алгоритми машинного навчання. Ефективність цих методів залежить від специфіки даних, складності структури тексту, а також від необхідного рівня точності та швидкості обробки. До найбільш розповсюджених підходів належать: наївний баєсівський класифікатор (Naive Bayes), дерева рішень (Decision Trees), метод опорних векторів (Support Vector Machines, SVM) та нейронні мережі (Neural Networks), включно з їх глибокими архітектурами [16].

Одним із найпростіших та найшвидших підходів до класифікації текстових повідомлень є наївний баєсівський класифікатор (Naive Bayes). Цей алгоритм ґрунтується на застосуванні теореми Байєса для обчислення ймовірності того, що певне повідомлення належить до того чи іншого класу (рис. 1.7). Основне припущення моделі полягає в незалежності ознак між собою, що спрощує обчислення, але в деяких випадках може знижувати точність класифікації. Незважаючи на простоту, Naive Bayes показує непогані результати для задач виявлення спаму завдяки високій швидкості навчання та передбачення.

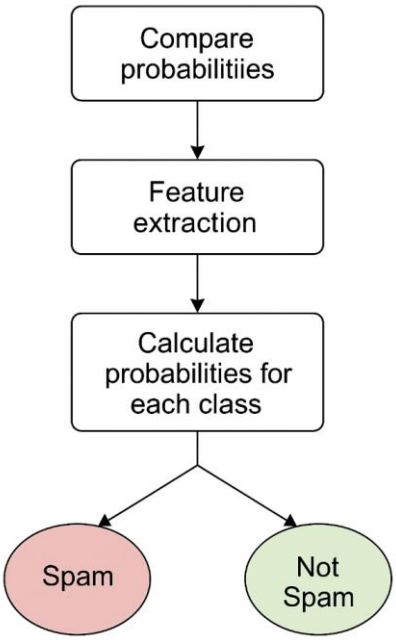



Рисунок 1.7 – Схема роботи наївного баєсівського класифікатора для класифікації повідомлень

Дерева рішень (Decision Trees) є ще одним популярним методом класифікації, який будує модель у вигляді дерева, де кожен вузол відповідає за прийняття рішення на основі певної ознаки, а гілки представляють можливі результати перевірки цієї ознаки (рис. 1.8). Метод дозволяє візуалізувати логіку прийняття рішень та легко інтерпретується. Проте для великих обсягів даних дерева рішень схильні до переобучення (overfitting), що може негативно вплинути на якість класифікації [17].

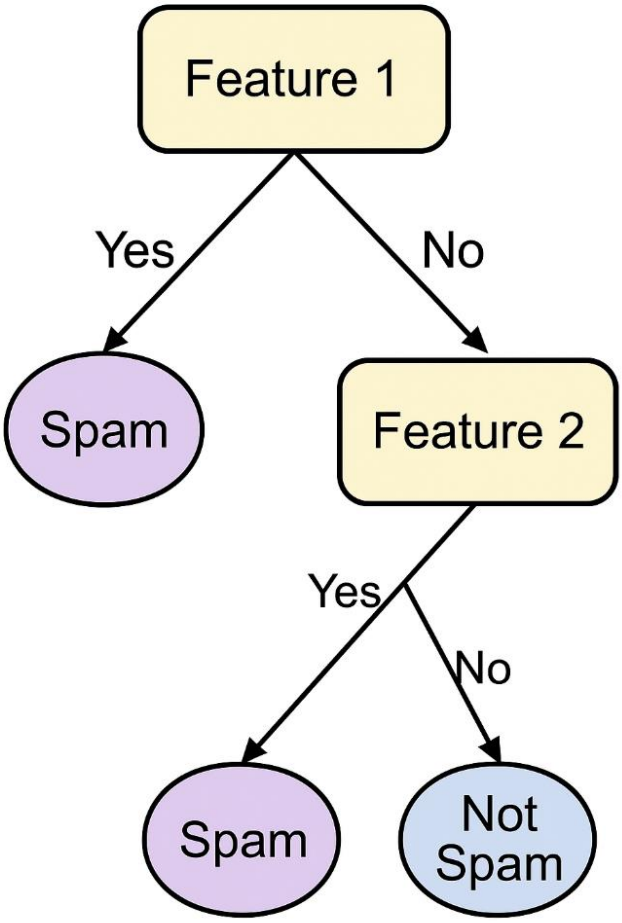



Рисунок 1.8 – Схема роботи дерева рішень для класифікації повідомлень

Більш складним, але потужним підходом є метод опорних векторів (Support Vector Machines, SVM), який будує гіперплощини для поділу даних різних класів у багатовимірному просторі ознак. SVM добре працює у випадках, коли необхідно знайти оптимальні межі між класами навіть при їх складній просторовій конфігурації. Алгоритм демонструє високу точність при класифікації текстових повідомлень, однак потребує значних обчислювальних ресурсів при роботі з великими наборами даних [17]. На рис. 1.9 схематично зображено

принцип роботи алгоритму SVM, де оптимальна гіперплощина розділяє два класи з максимальним відступом між найближчими точками (опорними векторами), що забезпечує ефективне розмежування даних.

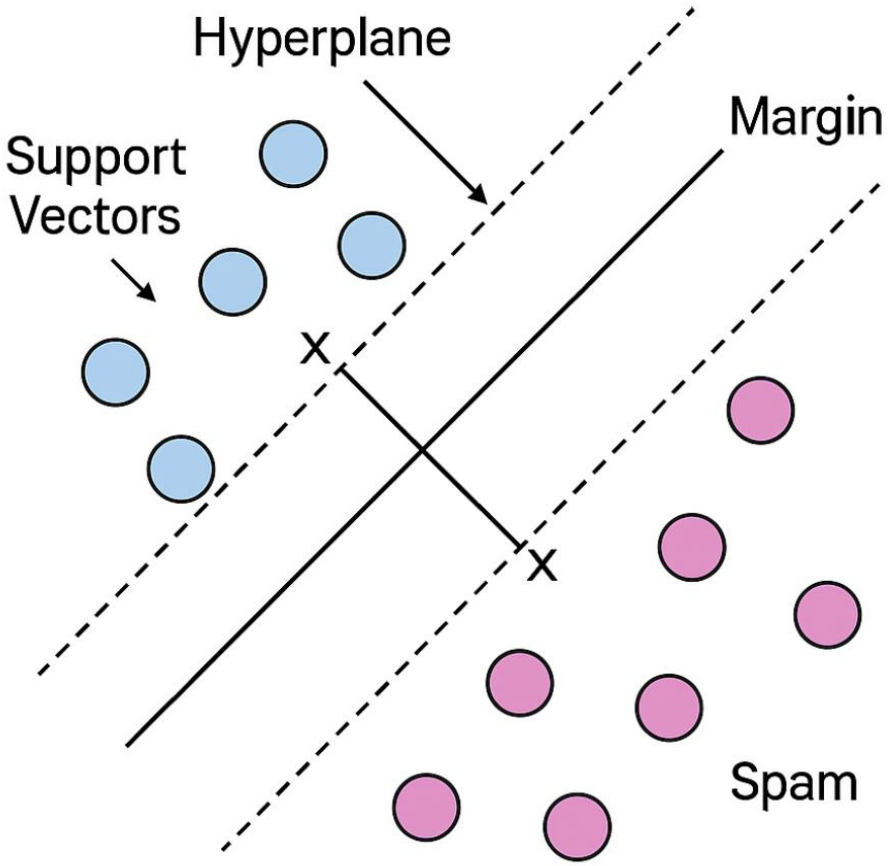



Рисунок 1.9 – Побудова гіперплощини методом опорних векторів (SVM)

Останнім часом значної популярності набули нейронні мережі (Neural Networks), зокрема їх глибокі архітектури, такі як багатошарові перцептрони (MLP), рекурентні нейронні мережі (RNN), довготривала короткочасна пам’ять (LSTM) та трансформери (наприклад, BERT). Завдяки здатності моделювати складні нелінійні залежності та зберігати інформацію про контекст, нейронні мережі демонструють високу ефективність у задачах обробки природної мови, зокрема для виявлення спаму, токсичних повідомлень та іншого небажаного контенту [17]. Вони забезпечують найвищу точність класифікації, але вимагають більше часу на тренування та наявності значних обчислювальних потужностей.

На рис. 1.10 представлено схематичне порівняння різних типів нейронних мереж: багатошаровий перцептрон (MLP), рекурентна нейронна мережа (RNN),

мережа з довготривалою короткочасною пам’яттю (LSTM) та трансформерна архітектура (Transformer). Схема демонструє відмінності у способі передачі та збереження інформації між шарами кожної архітектури.

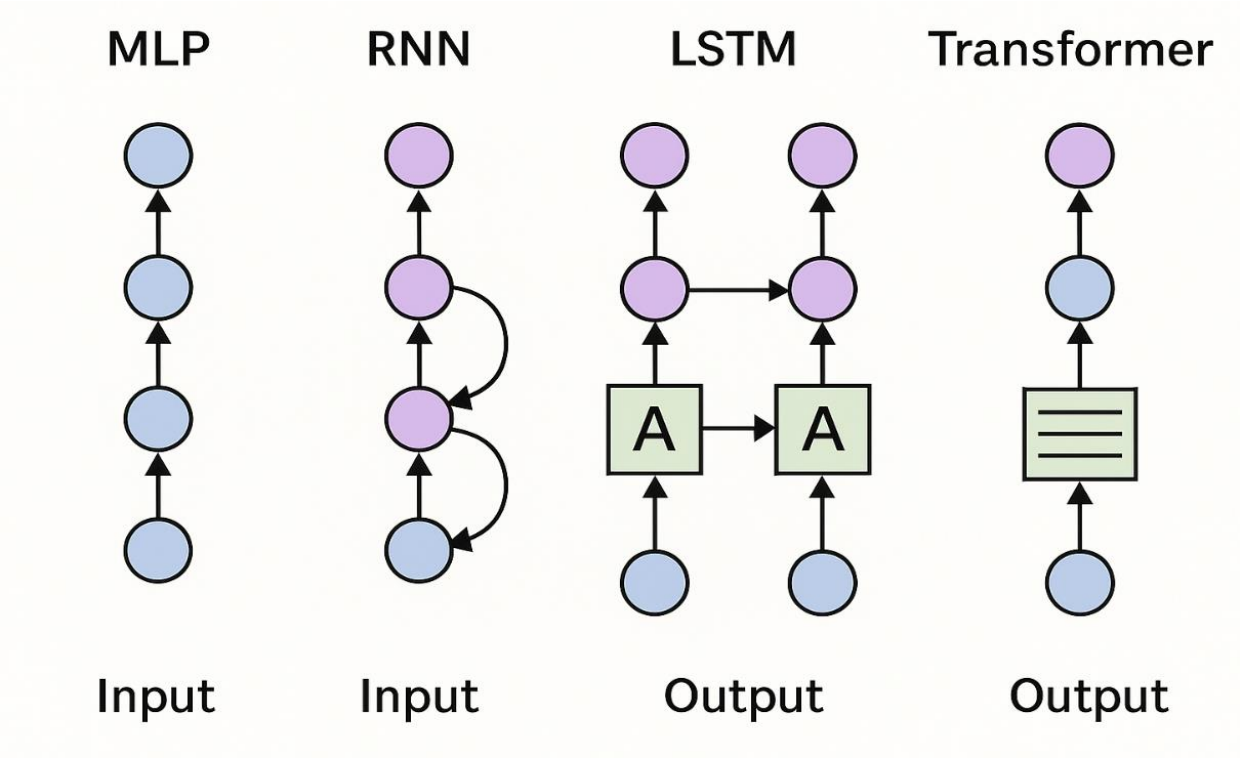



Рисунок 1.10 – Основні архітектури нейронних мереж для класифікації текстових повідомлень

Перед побудовою системи автоматичного виявлення небажаних повідомлень доцільно провести порівняльний аналіз існуючих алгоритмів класифікації, які застосовуються для обробки текстових даних. Різні методи машинного навчання мають свої переваги та обмеження залежно від складності задачі, розміру навчальної вибірки, потреб у швидкості обробки та доступності обчислювальних ресурсів. Простішими у впровадженні є традиційні підходи, такі як наївний баєсівський класифікатор та дерева рішень, тоді як більш сучасні моделі на основі нейронних мереж, зокрема рекурентних архітектур (LSTM) або трансформерів (BERT), забезпечують високу точність за рахунок складнішої обробки контексту повідомлень [18]. Для обґрунтування вибору оптимального підходу до розв'язання задачі класифікації небажаного контенту в соціальних мережах доцільно розглянути порівняльну характеристику зазначених методів (табл. 1.2).

Таблиця 1.2

Порівняння ефективності алгоритмів класифікації небажаного контенту

| **Алгоритм** | **Точність класифікації** | **Швидкість навчання** | **Стійкість до переобучення** | **Вимоги до ресурсів** | **Урахування контексту** |
|---|---|---|---|---|---|
| Naive Bayes | середня | висока | висока | низькі | не враховує |
| Decision Trees | середня | висока | схильність до переобучення | низькі–середні | частково (через структуру дерева) |
| SVM (Support Vector Machine) | висока | середня–низька | висока | середні–високі | не враховує |
| Neural Networks (MLP) | висока | середня | залежить від налаштувань | середні–високі | не враховує |
| RNN / LSTM | дуже висока | середня–низька | висока | високі | враховує довготривалі залежності |
| Transformer (BERT та ін.) | дуже висока | низька (потрібне тривале навчання) | висока | дуже високі | враховує контекст на глобальному рівні |

Порівнюючи ефективність зазначених алгоритмів, можна зробити висновок, що прості моделі, такі як Naive Bayes та Decision Trees, є оптимальними для початкового аналізу або коли потрібно досягти високої швидкості обробки при обмежених ресурсах. У той час як SVM та нейронні мережі показують кращі результати за точністю, особливо при обробці великих і складних текстових корпусів. Нейронні мережі, особливо моделі LSTM та BERT, є найбільш адаптованими до завдань класифікації природної мови, оскільки здатні враховувати контекст і взаємозв'язок між словами, що робить їх пріоритетним вибором для побудови систем виявлення небажаного контенту в соціальних мережах [19].

## Висновок до першого розділу

У першому розділі було проведено аналіз сучасних технік машинного навчання, які застосовуються для обробки текстових повідомлень з метою автоматичного виявлення небажаного контенту. Розглянуто теоретичні основи роботи рекурентних нейронних мереж (RNN), а також їх удосконалену архітектуру – довготривалу короткочасну пам’ять (LSTM), яка завдяки механізму збереження контексту та контролю інформаційного потоку дозволяє досягати високої точності класифікації повідомлень.

Надані визначення небажаних повідомлень дозволили структурувати розуміння основних типів такого контенту, серед яких виокремлено спам, токсичні повідомлення, шахрайські листи та фейкові новини. Це дало змогу сформувати критерії для подальшого моделювання та розробки системи виявлення небажаного контенту в соціальних мережах.

Також було проаналізовано ключові методи протидії небажаному контенту, включно з механізмами автентифікації електронної пошти SPF, DKIM та DMARC, що забезпечують перевірку достовірності відправника та сприяють зниженню ризику розповсюдження підроблених повідомлень.

Проведене теоретичне дослідження створює необхідне підґрунтя для реалізації практичної частини роботи, яка передбачає побудову та тестування моделі автоматичного виявлення небажаних повідомлень на основі методів машинного навчання.

# 2 ПОБУДОВА МОДЕЛІ ВИЯВЛЕННЯ НЕБАЖАНИХ ПОВІДОМЛЕНЬ

## 2.1 Архітектура моделі на основі машинного навчання

Для реалізації системи автоматичного виявлення небажаних повідомлень у соціальних мережах у даній роботі запропоновано побудову моделі на основі методів машинного навчання з використанням нейронної мережі типу LSTM (Long Short-Term Memory). Вибір такої архітектури зумовлений її здатністю ефективно опрацьовувати послідовні дані, зокрема текстові повідомлення, та зберігати інформацію про попередні елементи вхідної послідовності, що є критично важливим для врахування контексту при класифікації. Загальна структура моделі включає кілька основних етапів: підготовка навчальної вибірки, попередня обробка текстових даних, векторизація вхідної інформації, побудова та навчання LSTM-класифікатора, а також тестування та оцінка ефективності побудованої моделі. Запропонована архітектура дозволяє здійснювати класифікацію вхідних повідомлень за різними класами, такими як «спам», «токсичне повідомлення», «шахрайське повідомлення» або «нейтральне повідомлення», що забезпечує можливість автоматизованого моніторингу та фільтрації небажаного контенту в інформаційному просторі [20].

У представленій моделі (рис. 2.1) показано загальну архітектуру побудови системи, яка використовує нейронну мережу для обробки та генерації текстових повідомлень. На першому етапі формується нейронна мережа, для якої визначаються основні параметри, необхідні для налаштування процесу навчання. Далі відбувається підготовка навчальної вибірки, яка є базовим елементом для коректного функціонування моделі. Після цього нейронна мережа проходить процес навчання на підготовлених даних, що дозволяє їй адаптуватися до розв'язання поставленої задачі.


Рисунок 2.1 Архітектура моделі побудови нейронної мережі для обробки та генерації тексту

### 2.2 Формування навчальної вибірки та підготовка даних

Особливістю цієї архітектури є можливість використання зовнішніх API-сервісів, таких як Google API та Microsoft API, для автоматизованого перекладу текстів у разі потреби. Це забезпечує гнучкість системи в роботі з багатомовними повідомленнями. Підсумковим етапом є генерація тексту на основі натренованої моделі, що може використовуватися як для створення нового контенту, так і для аналізу або класифікації вхідних повідомлень [21].

Конфігураційні параметри для побудови та навчання нейронної мережі, які визначають основні характеристики моделі та процесу її тренування (рис. 2.2). Параметри моделі (model_cfg) задають архітектурні особливості нейронної мережі, зокрема розмір рекурентного шару (rnn_size), кількість шарів (rnn_layers), тип рекурентної архітектури (односпрямована або двонаправлена), максимальну довжину вхідної послідовності (max_length) та розмір словникового запасу (max_words). У даній конфігурації використовується тришарова односпрямована рекурентна нейронна мережа з розміром шару 128 нейронів та максимальною довжиною послідовності 30 слів.

```
[ ]  model_cfg = {
         'word_level': False,
         'rnn_size': 128,
         'rnn_layers': 3,
         'rnn_bidirectional': False,
         'max_length': 30,
         'max_words': 10000,
     }

     train_cfg = {
         'line_delimited': False,
         'num_epochs': 20,
         'gen_epochs': 5,
         'train_size': 0.8,
         'dropout': 0.0,
         'validation': False,
         'is_csv': False
     }
```

Рисунок 2.2 Конфігураційні параметри моделі та навчання нейронної мережі

Налаштування тренувального процесу (train_cfg) містить параметри, які регулюють кількість епох навчання (num_epochs), розмір навчальної вибірки (train_size), використання відсотка випадкового відключення нейронів для запобігання переобученню (dropout), а також можливість валідації та формат вхідних даних. Таке конфігурування забезпечує гнучкість та адаптивність нейронної мережі до конкретної задачі класифікації текстових повідомлень [22].

У процесі налаштування нейронної мережі для реалізації поставленої задачі було прийнято рішення використовувати переважно базові конфігураційні параметри, за винятком значення параметра «gen_epochs», який було свідомо збільшено. Це обумовлено тим, що збільшення кількості епох дозволяє забезпечити більш тривалий процес навчання моделі, що позитивно впливає на якість сформованого тексту.

Вибір стандартних параметрів ґрунтується на проведених попередніх експериментах із навчанням цієї нейронної мережі. У ході тестування було встановлено, що надмірне ускладнення архітектури моделі призводить до значного збільшення часу тренування, при цьому результати не покращуються, а навпаки – генерується текст низької якості, який є важким для сприйняття. Оптимальним часом для навчання моделі було визначено період до 12 годин. За цей проміжок часу при збереженні стандартних параметрів мережа встигала

якісно опрацювати навчальну вибірку та демонструвала здатність формувати текст, придатний для сприйняття людиною.

Для побудови моделі залишено стандартні значення максимальної кількості слів, які враховуються під час тренування, а також обсяг попереднього контексту – кількість символів або слів, що використовуються для прогнозування наступного елемента послідовності. Проведені дослідження підтвердили, що такі налаштування є оптимальними для досягнення балансу між часом навчання та якістю отриманого результату.

Після етапу налаштування параметрів нейронної мережі одним із ключових кроків є підготовка навчального датасету, який використовується як для процесу тренування моделі, так і для формування прикладів небажаних повідомлень, зокрема спаму. У межах цієї роботи було обрано один із найбільш відомих і широко використовуваних відкритих наборів електронної кореспонденції – Enron Email Dataset, що є придатним для задач аналізу текстів електронних листів [23].

Даний набір даних містить значну кількість електронних повідомлень різного характеру – як службових, так і особистих. У використаній версії датасету присутні понад 650 000 листів, які належать приблизно 170 співробітникам компанії, а сама структура листування представлена у вигляді більш ніж 4000 поштових папок. Важливо відзначити, що цей набір не включає вкладень, що дозволяє зосередитись саме на текстовому вмісті повідомлень. Таке наповнення дає змогу ефективно використовувати ці дані для навчання нейронної мережі в контексті завдання класифікації та автоматичного виявлення небажаного контенту.

Кожен лист у цьому наборі містить базову структуровану інформацію: адресу відправника та одержувача, дату та час надсилання, тему повідомлення, текст листа, а також технічні відомості, пов’язані з процесом передачі повідомлень. Саме на основі цього текстового контенту здійснюється формування навчальної вибірки для моделі, яка в подальшому здатна не лише класифікувати вхідні повідомлення як «спам» або «не спам», але й за необхідності генерувати

приклади спамових текстів для додаткового тестування або моделювання типових шаблонів небажаних листів.

Використання Enron Email Dataset дозволяє забезпечити достатню різноманітність навчальних прикладів та створює необхідне підґрунтя для підвищення точності класифікації, а також для відпрацювання механізмів розпізнавання потенційно небажаного контенту в умовах реального інформаційного середовища [24].

На рис. 2.3 зображено структуру списку користувачів у наборі даних Enron Email Dataset, де кожна папка відповідає окремому співробітнику компанії. Для кожного з цих користувачів у датасеті зберігаються електронні листи, які класифіковано за окремими підпапками, що дає змогу чітко ідентифікувати відправників та одержувачів повідомлень, а також структурувати кореспонденцію відповідно до організаційної ієрархії.

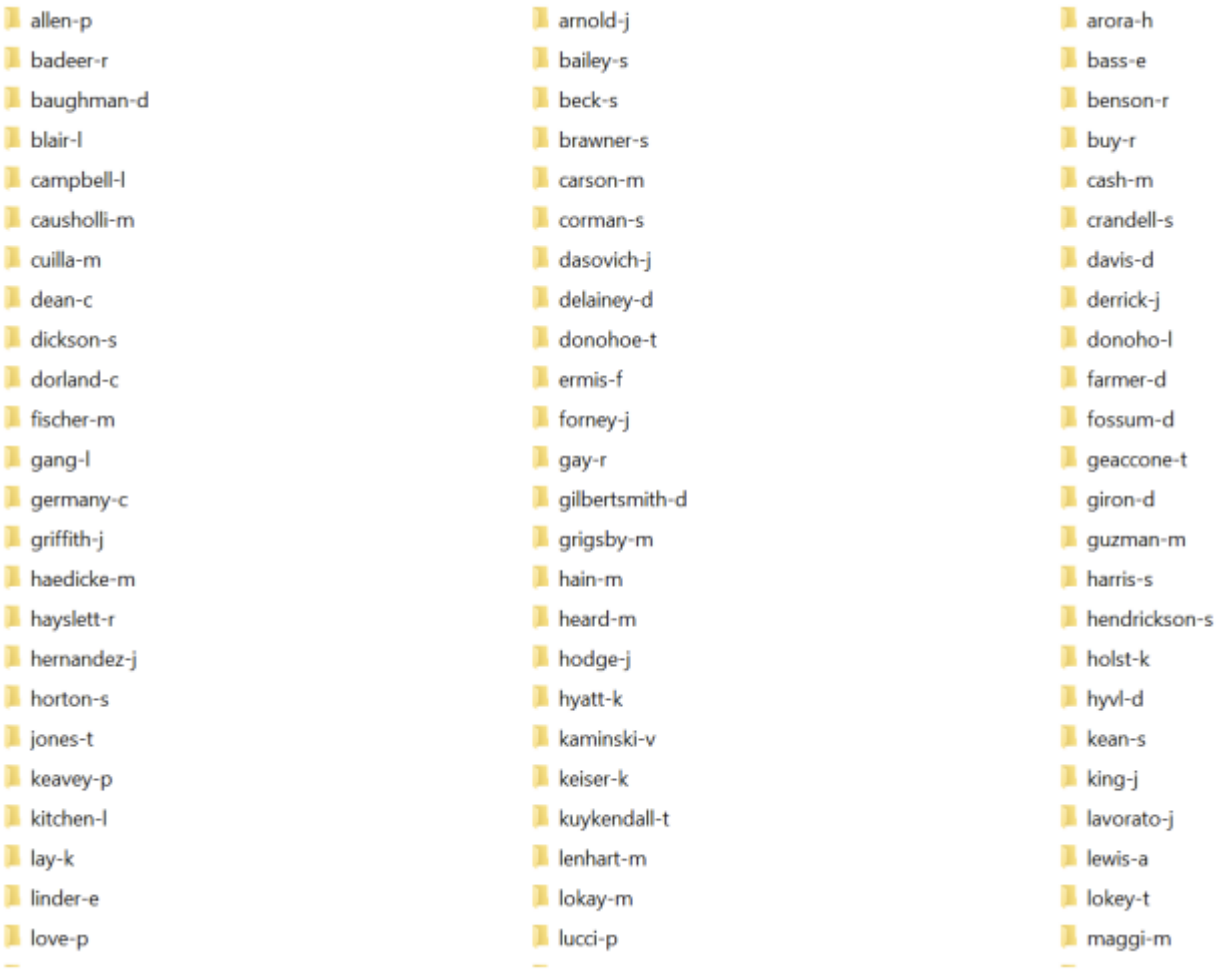
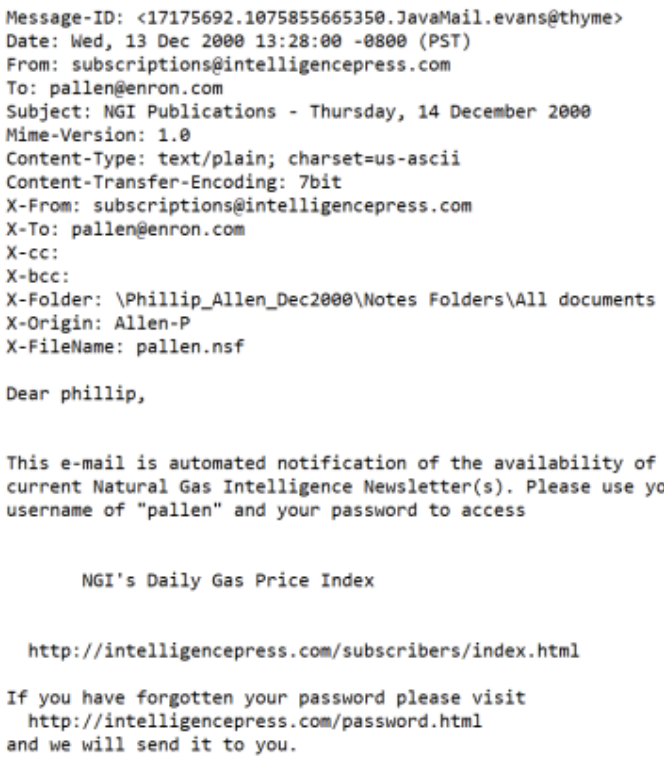



Рисунок 2.3 Користувачі

На прикладі фрагмента одного з листів (рис. 2.4) можна побачити структуру електронного повідомлення в цьому датасеті. Кожен лист містить службову інформацію: ідентифікатор повідомлення, дату та час відправлення, електронні адреси відправника та отримувача, тему повідомлення, а також основний текст листа [25]. Крім цього, у заголовку листа зазначено додаткові метадані, зокрема ім’я файлу, до якого прив’язане повідомлення, а також дані про розташування листа в папках відповідного користувача.



```
Message-ID: <17175692.1075855665350.JavaMail.evans@thyme>
Date: Wed, 13 Dec 2000 13:28:00 -0800 (PST)
From: subscriptions@intelligencepress.com
To: pallen@enron.com
Subject: NGI Publications - Thursday, 14 December 2000
Mime-Version: 1.0
Content-Type: text/plain; charset=us-ascii
Content-Transfer-Encoding: 7bit
X-From: subscriptions@intelligencepress.com
X-To: pallen@enron.com
X-cc:
X-bcc:
X-Folder: \Phillip_Allen_Dec2000\Notes Folders\All documents
X-Origin: Allen-P
X-FileName: pallen.nsf

Dear phillip,


This e-mail is automated notification of the availability of your
current Natural Gas Intelligence Newsletter(s). Please use your
username of "pallen" and your password to access


      NGI's Daily Gas Price Index


  http://intelligencepress.com/subscribers/index.html

If you have forgotten your password please visit
  http://intelligencepress.com/password.html
and we will send it to you.
```

Рисунок 2.4 – Приклад електронного листа з набору даних Enron Email Dataset

Структура дозволяє формувати навчальну вибірку для тренування нейронної мережі та забезпечує можливість коректної розмітки повідомлень для подальшої класифікації за ознакою «спам» або «не спам», а також для генерації шаблонів небажаних текстів.

**Висновок до другого розділу**

У другому розділі було розглянуто основні етапи побудови моделі для автоматичного виявлення небажаних повідомлень із використанням методів машинного навчання. Обґрунтовано вибір архітектури нейронної мережі, яка базується на використанні рекурентної структури з елементами довготривалої

короткочасної пам’яті (LSTM), що забезпечує здатність моделі зберігати контекст та ефективно працювати з послідовними текстовими даними. Така архітектура дозволяє підвищити якість класифікації повідомлень, особливо у випадках складних мовних конструкцій та великої різноманітності текстового наповнення.

Описано процес підготовки навчальної вибірки, який включає збір, обробку та розмітку даних на основі відомого датасету Enron Email Dataset, що містить велику кількість електронних листів різного характеру. Здійснено попередню обробку текстів, яка передбачає очищення даних від службових символів, видалення зайвих елементів форматування, а також нормалізацію тексту з метою підвищення якості навчання моделі.

Проведені дії на цьому етапі створили необхідне підґрунтя для реалізації наступних кроків, пов’язаних із навчанням моделі, її оптимізацією та тестуванням на контрольних вибірках. Запропонована структура моделі та сформована навчальна база є основою для подальшого підвищення ефективності системи автоматичного виявлення небажаного контенту в електронних повідомленнях.

# 3 ПРАКТИЧНА РЕАЛІЗАЦІЯ ТА ТЕСТУВАННЯ СИСТЕМИ

## 3.1 Програмна реалізація моделі виявлення

Експериментальне тестування моделі було виконано у середовищі операційної системи Ubuntu версії 20.04.2 LTS.

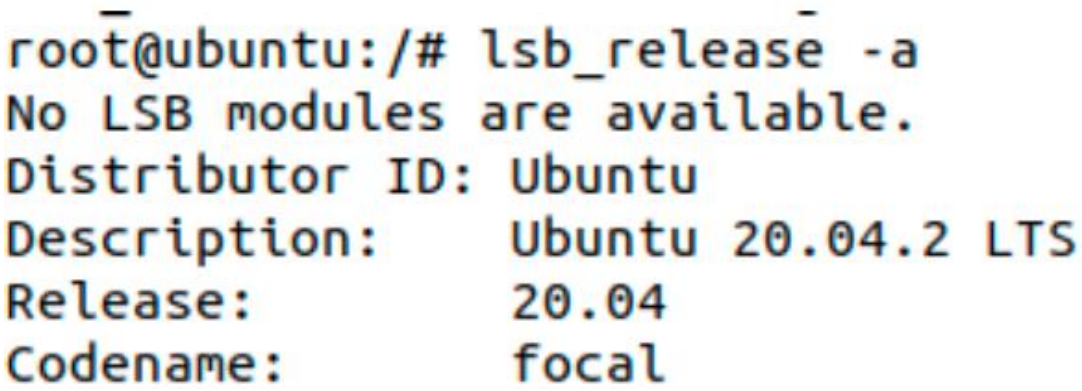



Рисунок 3.1 Інформація про версію використаної операційної системи Ubuntu 20.04.2 LTS.

У межах реалізації моделі виявлення було виконано встановлення пакета фільтрації електронної пошти SpamAssassin на базі операційної системи Ubuntu 20.04.2 LTS. Для інсталяції актуальної версії SpamAssassin застосовувалася команда системного менеджера пакетів [26]:

*apt install spamassassin*

Після завершення інсталяції виконувалося налаштування параметрів роботи спам-фільтра. Основна конфігурація здійснювалася через редагування файлу налаштувань, розташованого за адресою:

*/etc/spamassassin/local.cf*

До цього файлу було додано необхідні правила та параметри, які регламентують логіку аналізу вхідної пошти та визначають критерії класифікації повідомлень як спам.

```
report_safe 1

# Set the threshold at which a message is considered spam (default: 5.0)
required_score 2.0

# Use Bayesian classifier (default: 1)
use_bayes 1

# Bayesian classifier auto-learning (default: 1)
bayes_auto_learn 1

include /usr/share/spamassassin/99_wentor.cf
include /usr/share/spamassassin/99_russian_common_re.cf

score FUZZY_XPILL 1.0
score BAYES_60 3.5
score BAYES_95 5.5
score BAYES_99 6.0
score BAYES_999 3.0

# Set headers which may provide inappropriate cues to the Bayesian classifier
# bayes_ignore_header X-Bogosity
# bayes_ignore_header X-Spam-Flag
# bayes_ignore_header X-Spam-Status
```

Рисунок 3.2 Налаштування параметрів аналізу повідомлень у SpamAssassin

Після внесення змін до конфігураційного файлу local.cf в системі фільтрації спаму SpamAssassin необхідно виконати перевірку коректності синтаксису налаштувань. Для цього застосовується команда [26, 27]:

*spamassassin –lint*

Ця команда дозволяє виявити можливі помилки в структурі конфігураційного файлу ще до запуску системи в робочому середовищі, що сприяє підвищенню надійності роботи фільтра та запобігає некоректній обробці поштового трафіку.

```
root@ubuntu:/# spamassassin --lint
root@ubuntu:/#
```

Рисунок 3.3 Відображення успішної валідації конфігураційного файлу local.cf у SpamAssassin

Для створення тестового набору спам-листів необхідно об’єднати всі електронні листи з Enron-датасету в єдиний файл, попередньо очистивши їх від зайвої інформації. Цей процес автоматизовано за допомогою Python-скрипта, який дозволяє швидко агрегувати повідомлення в один текстовий документ.

```
import fileinput
import glob
flist= glob.glob("*.")
with open('dataset_text.txt', 'w') as file:
linput = fileinput.input(flist)
file.writelines(linput)
```

Рисунок 3.4 Завантаження датасету

У результаті виконання цього Python-скрипта формується єдиний текстовий файл, що містить об’єднані електронні листи з Enron-датасету. Цей файл слугує основою для подальшої генерації навчальної вибірки для тестування спам-фільтра [27].

```
Date: Thu, 3 Aug 2000 08:38:00 -0700 (PDT)
From: john.arnold@enron.com
To: david.forster@enron.com, david.forster@enron.com
Subject:
Mime-Version: 1.0
Content-Type: text/plain; charset=us-ascii
Content-Transfer-Encoding: 7bit
X-From: John Arnold
X-To: David Forster, David Forster
X-cc:
X-bcc:
X-Folder: \John_Arnold_Dec2000\Notes Folders\'sent mail
X-Origin: Arnold-J
X-FileName: Jarnold.nsf

Hello:
There is one feature that has not been transferred to the new stack manager.
When I sort by both counterparty and product at the bottom of the screen, the
position summary does not sort by both product and counterparty, just by
product. It would be very useful if you can replace this.
Thanks,
John

Message-ID: <22811141.1075857600347.JavaMail.evans@thyme>

Date: Thu, 3 Aug 2000 02:58:00 -0700 (PDT)
From: john.arnold@enron.com
To: per.sekse@enron.com
Subject:
Mime-Version: 1.0
Content-Type: text/plain; charset=us-ascii
Content-Transfer-Encoding: 7bit
X-From: John Arnold
X-To: Per Sekse
X-cc:
```

Рисунок 3.5 Структура сформованого файлу з об’єднаними листами

Щоб підготувати середовище для роботи нейронної мережі на основі TensorFlow першої версії, необхідно встановити відповідну версію фреймворку та всі потрібні додаткові бібліотеки. Це дозволить забезпечити коректну роботу

програмного коду та уникнути конфліктів залежностей. Основні етапи підготовки середовища виглядають наступним чином [28]:

Створення віртуального середовища Python (рекомендується для ізоляції проєкту):

*python3 -m venv venv source venv/bin/activate*

Встановлення TensorFlow версії 1.x:

*pip install tensorflow==1.15*

Додавання необхідних бібліотек для роботи з обробкою даних і нейронними мережами:

*pip install numpy pandas matplotlib scikit-learn*

Перевірка встановлення TensorFlow:

*import tensorflow as tf*

*print(tf.__version__)*

```
%tensorflow_version 1.x

!pip install -q textgenrnn
from google.colab import files
from textgenrnn import textgenrnn
from datetime import datetime
import os
```

Рисунок 3.6 Налаштування бібліотек для моделі генерації тексту на базі TensorFlow

Після обробки даних необхідно завантажити отриманий файл до моделі для реалізації етапу тренування нейронної мережі.

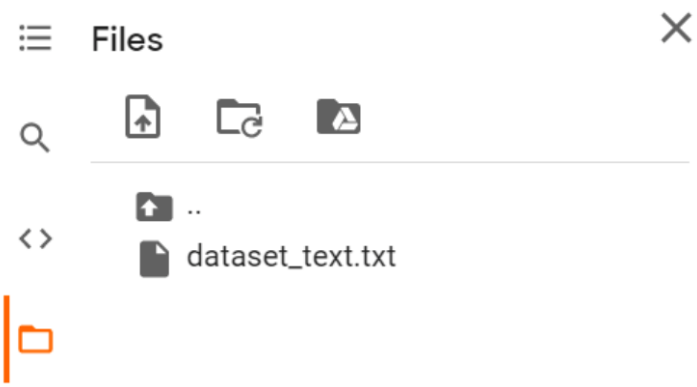



Рисунок 3.7 Завантаження підготовленого файлу для навчання нейронної мережі

Для налаштування процесу навчання нейронної мережі необхідно вказати параметри конфігурації, зокрема встановити кількість епох (параметр num_epochs). Для забезпечення якісного навчання моделі на основі підготовленого набору даних рекомендовано використовувати значення кількості епох у діапазоні від 30 до 50. У даному дослідженні для оптимального балансу між часом навчання (приблизно 12 годин) та ефективністю моделі було обрано параметр num_epochs = 40 [28].

```
model_cfg = {
    'word_level': False,
    'rnn_size': 128,
    'rnn_layers': 3,
    'rnn_bidirectional': False,
    'max_length': 30,
    'max_words': 10000,
}

train_cfg = {
    'line_delimited': False,
    'num_epochs': 40,
    'gen_epochs': 5,
    'train_size': 0.8,
    'dropout': 0.0,
    'validation': False,
    'is_csv': False,
}
```

Рисунок 3.8 Основні налаштування моделі та процесу тренування нейронної мережі

Для подальшого навчання нейронної мережі необхідно вказати шлях до файлу з підготовленими даними та задати імена файлів для збереження проміжних і фінальних результатів генерації. Це дозволяє зручно організувати процес навчання та зберігати створені моделі.

```
file_name = "dataset_text.txt"
model_name = 'colaboratory'
```

Рисунок 3.9 Визначення імені моделі та підключення навчального датасету

Після виконання налаштувань параметрів моделі та підготовки навчального набору даних розпочинається процес навчання нейронної мережі з використанням заданого текстового файлу [28-30].

```
textgen = textgenrnn(name=model_name)

train_function = textgen.train_from_file if train_cfg['line_delimited'] else textgen.train_from_largetext_file

train_function(
    file_path=file_name,
    new_model=True,
    num_epochs=train_cfg['num_epochs'],
    gen_epochs=train_cfg['gen_epochs'],
    batch_size=1024,
    train_size=train_cfg['train_size'],
    dropout=train_cfg['dropout'],
    validation=train_cfg['validation'],
    is_csv=train_cfg['is_csv'],
    rnn_layers=model_cfg['rnn_layers'],
    rnn_size=model_cfg['rnn_size'],
    rnn_bidirectional=model_cfg['rnn_bidirectional'],
    max_length=model_cfg['max_length'],
    dim_embeddings=100,
    word_level=model_cfg['word_level']
)
```

Рисунок 3.10 Конфігурація та старт тренування нейронної мережі

У результаті навчання нейронної мережі автоматично формуються три основних файли, які забезпечують можливість подальшого використання моделі для генерації тексту. Цими файлами є:

– файл з вагами моделі (weights);
– файл зі словником, що зберігає словниковий запас, використаний під час навчання (vocab);
– файл із конфігураційними налаштуваннями нейронної мережі (config).

Ці файли дозволяють легко відновити модель без повторного навчання та використовувати її для генерації текстових даних на основі попередньо опрацьованого датасету.

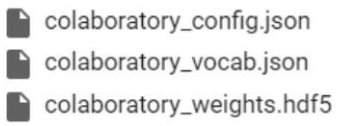



Рисунок 3.11 Результат збереження конфігурації, словника та ваг моделі після тренування

Після завершення 12-годинного процесу навчання нейронної мережі стає можливим розпочати генерацію текстових даних на основі сформованої моделі [30].

```
temperature = [1.0, 0.5, 0.2, 0.2]
prefix = None

if train_cfg['line_delimited']:
    n = 1000
    max_gen_length = 60 if model_cfg['word_level'] else 300
else:
    n = 1
    max_gen_length = 2000 if model_cfg['word_level'] else 10000

timestring = datetime.now().strftime('%Y%m%d_%H%M%S')
gen_file = '{}_gentext_{}.txt'.format(model_name, timestring)

textgen.generate_to_file(gen_file,
                         temperature=temperature,
                         prefix=prefix,
                         n=n,
                         max_gen_length=max_gen_length)

files.download(gen_file)
```

Рисунок 3.12 Формування тексту з використанням натренованої нейронної мережі

В результаті роботи навченої нейронної мережі отримуємо згенерований текстовий фрагмент, який демонструє можливості моделі щодо побудови текстових послідовностей на основі навченого датасету.

```
1 Spectacularly study finds to find the students of that to be fun to compare questions.
2
```

Рисунок 3.13 Результат генерації тексту після навчання нейронної моделі

Оскільки для подальшого аналізу одного згенерованого фрагмента тексту недостатньо, виникає потреба автоматизувати процес створення більшої кількості таких текстів. Для цього доцільно реалізувати цикл генерації, який дозволяє вказати необхідну кількість текстових зразків для подальших досліджень. Для забезпечення стабільної роботи процесу та уникнення конфліктів при генерації і збереженні файлів, між створенням окремих текстів рекомендовано встановити паузу тривалістю в одну секунду.

```
temperature = [1.0, 0.5, 0.2, 0.2]
prefix = None

if train_cfg['line_delimited']:
    n = 1000
    max_gen_length = 60 if model_cfg['word_level'] else 300
else:
    n = 1
    max_gen_length = 2000 if model_cfg['word_level'] else 10000

for i in range(1, 3002):
    timestring = datetime.now().strftime('%Y%m%d_%H%M%S')
    gen_file = '{}_gentext_{}.txt'.format(model_name, timestring)

    textgen.generate_to_file(
        gen_file,
        temperature=temperature,
        prefix=prefix,
        n=n,
        max_gen_length=max_gen_length
    )

    files.download(gen_file)
    time.sleep(1)
```

Рисунок 3.14 Автоматизована генерація тисяч текстів за допомогою нейронної мережі

В результаті виконання скрипта було сформовано 3000 текстових файлів. Для покращення якості навчання фільтра SpamAssassin та забезпечення достатнього обсягу даних для подальших експериментів, зокрема перекладу текстів на російську мову або додаткового навчання моделі, заплановано згенерувати ще декілька тисяч текстів. Це дозволить підвищити якість навчальної вибірки та досягти більшої точності в роботі з різними мовними наборами [30].

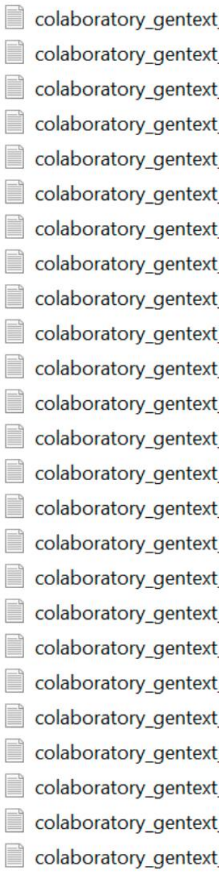



Рисунок 3.15 Створені тексти для подальшого використання у навчанні фільтрації спаму

Для реалізації автоматизованого перекладу великої кількості згенерованих текстів на різні мови доцільно використовувати API від Google та Microsoft. Для інтеграції сервісу Google Translation необхідно попередньо створити проєкт у Google Cloud Platform, активувати API Google Translation та згенерувати файл облікових даних (JSON), який забезпечує авторизований доступ до сервісу.

Після отримання облікових даних потрібно встановити відповідну бібліотеку для Python за допомогою команди:

***pip install --upgrade google-cloud-translate***

Далі представлено фрагмент коду для роботи з Google Translation API, що використовується для перекладу текстів на Python:

```python
from google.cloud import translate_v2 as translate
import os

# Вказуємо шлях до JSON з обліковими даними
os.environ['GOOGLE_APPLICATION_CREDENTIALS'] = 'path_to_your_credentials.json'

# Ініціалізація клієнта Google Translation
translate_client = translate.Client()

def translate_text(text, target_language='ru'):
    result = translate_client.translate(text, target_language=target_language)
    return result['translatedText']

# Приклад використання
translated_text = translate_text("Hello, how are you?", target_language='ru')
print(translated_text)
```

Рисунок 3.16 Підключення до гугл клієнта для перекладу

Цей код забезпечує можливість автоматизованого перекладу текстових даних, що значно спрощує процес підготовки багатомовних наборів для подальшого навчання або тестування моделей фільтрації спаму.

Використання розробленого програмного рішення дозволяє автоматизовано здійснювати переклад великих масивів текстових даних, що складаються з кількох тисяч файлів, на будь-яку обрану мову. Завдяки інтеграції з Google Translation API реалізовано зручний механізм масового перекладу, який забезпечує високу швидкість обробки та точність результатів.

Цей підхід значно оптимізує процес підготовки мовних корпусів для подальшого використання в системах фільтрації спаму або інших задачах обробки природної мови, де необхідно мати багатомовний набір даних [31-32].

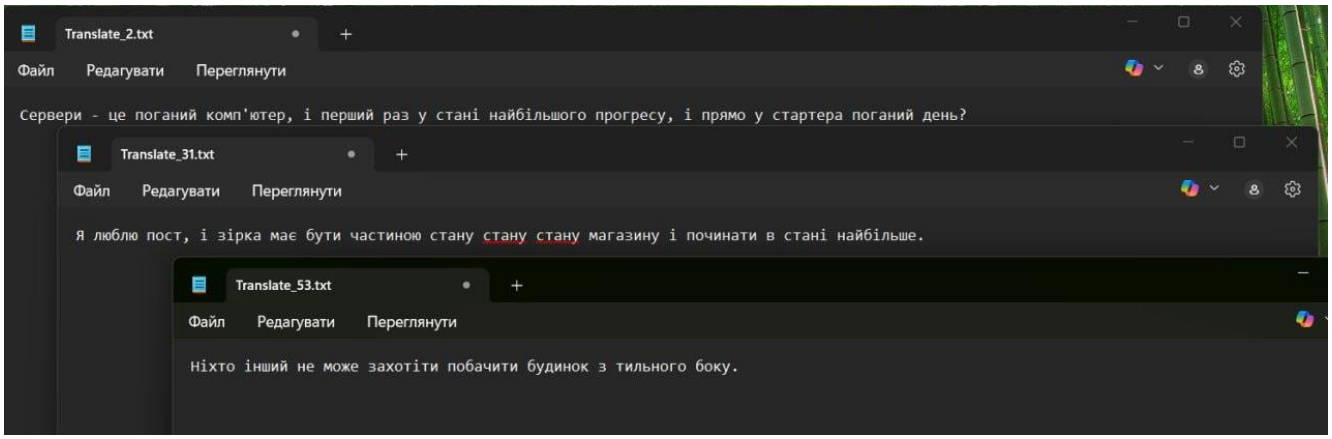



Рисунок 3.17 Приклад результатів автоматичного перекладу текстів за допомогою Google API.

Щоб працювати з Microsoft API Translator, необхідно спочатку створити обліковий запис на Azure та активувати відповідну послугу Translator Text API. Після цього в налаштуваннях ресурсу можна отримати ключ підписки (Subscription Key) та endpoint URL, які будуть використовуватися для аутентифікації в API-запитах.

Цей ключ є обов'язковим для авторизації при кожному виклику API, що забезпечує захищений доступ до сервісу машинного перекладу Microsoft [33].

```
import requests, uuid, json
import os

subscription_key = "ключ"
endpoint = "https://api.cognitive.microsofttranslator.com/"
j = 1
a = os.listdir(path=".")
path = '/translate'
constructed_url = endpoint + path
params = {
    'api-version': '3.0',
    'from': ['en'],
    'to': 'ru'
}

headers = {
    'Ocp-Apim-Subscription-Key': subscription_key,
    'Content-type': 'application/json',
    'X-ClientTraceId': str(uuid.uuid4()),
    'Ocp-Apim-Subscription-Region': 'westeurope'
}

for i in a:
    inputfile = i
    outputfile = "translate_" + str(j) + ".txt"

    f = open(inputfile,'r', encoding='utf-8')
    myfile=open(outputfile,'w', encoding='utf-8')
    text=f.read().replace('\n',"").replace('.',". ")
    body = [{
        'text':text
    }]
    request = requests.post(constructed_url, params=params, headers=headers, json=body)
    myfile.write(request.json()[0]["translations"][0]["text"])
    myfile.close
    j = j + 1
```

Рисунок 3.18 Реалізація коду для перекладу текстів за допомогою Microsoft Translator API

### 3.2 Тестування системи на даних соціальних мереж

Для забезпечення коректної роботи байєсівського класифікатора спам-фільтра SpamAssassin необхідно провести попереднє навчання системи, використовуючи як спам, так і не спам (хем) листи. Важливою умовою ефективності такого навчання є наявність не лише тіла листа, але й відповідних заголовків, адже саме вони містять ключову інформацію для коректної ідентифікації електронної пошти.

Оскільки в наявності є лише тексти повідомлень без заголовків, було прийнято рішення автоматизувати процес їх додавання до кожного листа. Для цього використано bash-скрипт, який додає стандартні заголовки MIME, дату, ID повідомлення, тему (Subject), відправника (From), одержувача (To) та тип контенту (Content-Type). Такий підхід дозволяє створити структуровані електронні листи, які можна використовувати для подальшого навчання байєсівського класифікатора.

```
#!/bin/bash
sed -i -e '1 s/^/MIME-Version: 1.0\n/;' *
sed -i -e '2 s/^/Date: Wed, 9 Oct 2019 23:09:51 +0300\n/;' *
sed -i -e '3 s/^/Message-ID: <CAK48JR5kcpFYTVyRtRtXDhfTVj=RmzmDfdMp5vWuuaoKTM8VOA@mail.gmail.com>\n/;' *
sed -i -e '4 s/^/Subject: Introduction\n/;' *
sed -i -e '5 s/^/From: Frosty Wind<121@gmail.com>\n/;' *
sed -i -e '6 s/^/To: 12@gmail.com\n/;' *
sed -i -e '7 s/^/Content-Type: multipart\/mixed; boundary="0000000000000650df05947fdee2"\n/;' *
sed -i -e '8 s/^/\n/;' *
```

Рисунок 3.19 Задання HTTP хедерів

Цей Bash-скрипт автоматично додає необхідні заголовки (headers) до кожного файлу листа, що зберігається у відповідній директорії. Це потрібно для коректного навчання байєсівського класифікатора в SpamAssassin, оскільки фільтр враховує як тіло листа, так і його заголовки [34].

```
I am a stream to the show of the first time in the first time in the first time. I have a
state of the state of the state of the community of the same way to see the company to start a bad
game of the state of the background of the state of the same state with a stream on the state
of the story of the state of the state of the most possible to be a bit of the community of the
state of the program and I can see if you want to see the world on the first time in the state
of the state of the same character is a bad police of the state of the state of the most
process state of the season to the state of the side of the strangest state of the starting and
a good day where the most second of the state of the streets and what is the back of the state
of the state of the same story of the first time in the same second things and the comment is
too starting to the story of the state of the state of the streets of the sidewalk by a stream.
```

Рисунок 3.20 Фрагмент згенерованого тексту без доданих заголовків

```
MIME-Version: 1.0
Date: Wed, 9 Oct 2019 23:09:51 +0300
Message-ID: <CAK48JR5kcpFYTVyRtRtXDhfTVj=RmzmDfdMp5vWuuaoKTM8VOA@mail.gmail.com>
Subject: Introduction
From: Alex White <alexwhite@example.com>
To: security.team@cybermail.com
Content-Type: multipart/mixed; boundary="0000000000000650df05947fdee2"

I am a stream to the show of the first time in the first time in the first time. I have a
state of the state of the state of the community of the same way to see the company to start a bad
game of the state of the background of the state of the same state with a stream on the state
of the story of the state of the state of the most possible to be a bit of the community of the
state of the program and I can see if you want to see the world on the first time in the state
of the state of the same character is a bad police of the state of the state of the most
process state of the season to the state of the side of the strangest state of the starting and
a good day where the most second of the state of the streets and what is the back of the state
of the state of the same story of the first time in the same second things and the comment is
too starting to the story of the state of the state of the streets of the sidewalk by a stream.
```

Рисунок 3.21 Приклад листа після додавання заголовків для навчання SpamAssassin

SpamAssassin використовує низку методів для прийняття рішення щодо класифікації електронних листів як спам або легітимних повідомлень. Основні підходи включають:

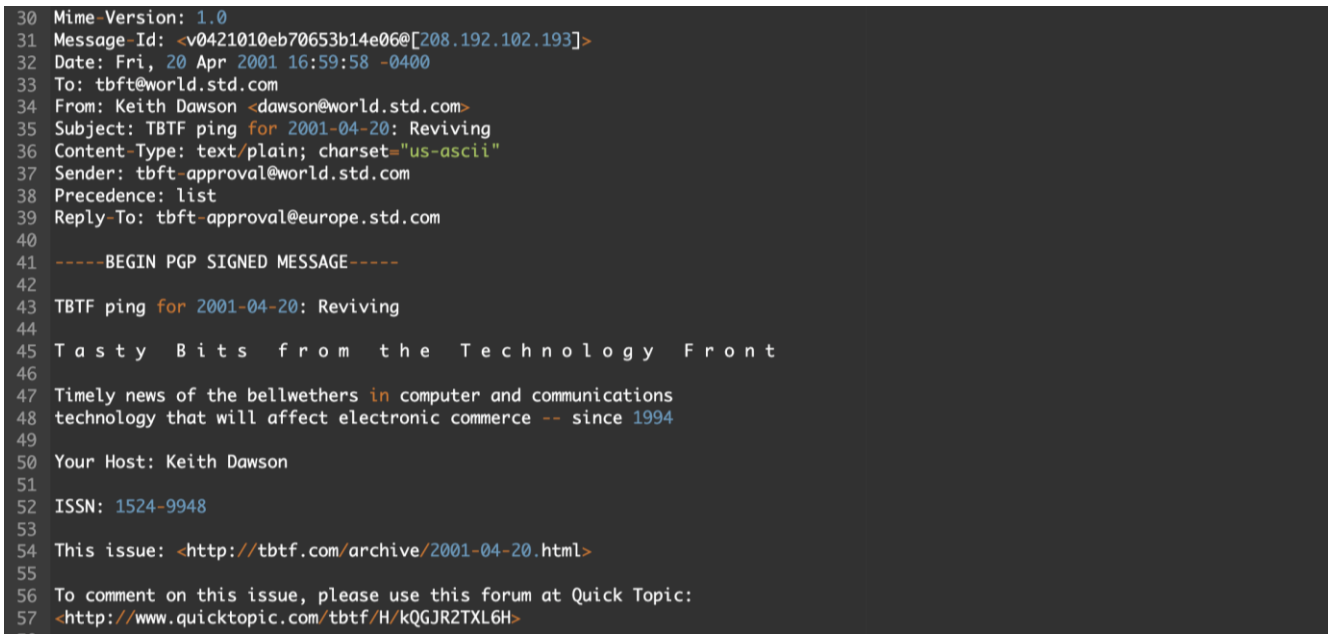
- Аналіз заголовків повідомлень на відповідність стандартам Інтернет-протоколів, зокрема перевірку правильності форматування полів, таких як дата та час.
- Пошук ключових фраз та патернів у заголовках та тілі листа, характерних для спам-повідомлень (наприклад, фрази на кшталт «швидко заробити гроші», «інструкції з відписки від розсилки» тощо). Система підтримує аналіз текстів на кількох мовах, що підвищує ефективність виявлення спаму.
- Використання чорних та білих списків для адрес електронної пошти, доменів та IP-адрес. Білий список може автоматично формуватись на основі історії отриманих повідомлень від конкретного відправника, що знижує ймовірність помилкового спрацьовування фільтра.
- Перевірка автентичності IP-адреси відправника шляхом порівняння її з доменним іменем за допомогою механізму Sender Policy Framework (SPF). Це дозволяє визначити, чи уповноважена відправляюча система надсилати листи від імені певного домену, що є одним із ключових етапів боротьби зі спуфінгом [35-36].

```
Received: from localhost by ubuntu
    with SpamAssassin (version 3.4.4);
    Sun, 30 April 2025 12:53:38 -0700
From: Sender <sender@example.net>
To: Recipient <recipient@example.net>
Subject: Test spam mail (GTUBE)
Date: Wed, 23 Jul 2003 23:30:00 +0200
Message-Id: <GTUBE.1010101@example.net>
X-Spam-Checker-Version: SpamAssassin 3.4.4 (2020-01-24) on ubuntu
X-Spam-Flag: YES
X-Spam-Level: **********************************************************
X-Spam-Status: Yes, score=1000.8 required=2.0 tests=BAYES_50,GTUBE,NO_RECEIVED,
    NO_RELAYS autolearn=no autolearn_force=no version=3.4.4

MIME-Version: 1.0
Content-Type: multipart/mixed; boundary="----------=_60B42582.481F151B"

This is a multi-part message in MIME format.

------------=_60B42582.481F151B
Content-Type: text/plain; charset=iso-8859-1
Content-Disposition: inline
Content-Transfer-Encoding: 8bit

Spam detection software, running on the system "ubuntu",
has identified this incoming email as possible spam. The original
message has been attached to this so you can view it or label
similar future email. If you have any questions, see
the administrator of that system for details.

Content preview:  This is the GTUBE, the Generic Test for Unsolicited Bulk Email
  If your spam filter supports it, the GTUBE provides a test by which you can
  verify that the filter is installed correctly and is detecting incoming spam.
  You can send yourself a test mail containing t [...]

Content analysis details:   (1000.8 points, 2.0 required)

 pts rule name              description
---- ---------------------- --------------------------------------------------
 0.8 BAYES_50               BODY: Bayes spam probability is 40 to 60%
                            [score: 0.4825]
1000 GTUBE                  BODY: Generic Test for Unsolicited Bulk Email
 -0.0 NO_RELAYS             Informational: message was not relayed via SMTP
 -0.0 NO_RECEIVED           Informational: message has no Received headers
```

Рисунок 3.22 Приклад повідомлення, ідентифікованого SpamAssassin як спам

```
Return-Path: <tbft-approval@world.std.com>
X-Spam-Checker-Version: SpamAssassin 3.4.4 (2020-01-24) on ubuntu
X-Spam-Level: **
X-Spam-Status: No, score=2.0 required=5.0 tests=BAYES_80,SPF_HELO_NONE,SPF_NONE autolearn=no autolearn_force=no version=3.4.4
Delivered-To: foo@foo.com
Received: from europe.std.com (europe.std.com [199.172.62.20])
    by mail.netnoteinc.com (Postfix) with ESMTP id 392E1114061
    for <foo@foo.com>; Fri, 20 Apr 2001 21:34:46 +0000 (Eire)
Received: from daemon@localhost
    by europe.std.com (8.9.3/8.9.3) id RAA09630
    for tbft-outgoing; Fri, 20 Apr 2001 17:31:18 -0400 (EDT)
Received: from sgi04-e.std.com (sgi04-e.std.com [199.172.62.134])
    by europe.std.com (8.9.3/8.9.3) with ESMTP id RAA08749
    for <tbft@facteur.std.com>; Fri, 20 Apr 2001 17:24:31 -0400 (EDT)
Received: from world.std.com (world-f.std.com [199.172.62.5])
    by sgi04-e.std.com (8.9.3/8.9.3) with ESMTP id RAA287330
    for <tbft@facteur.std.com>; Fri, 20 Apr 2001 17:24:31 -0400 (EDT)
Received: (from dawson@localhost)
    by world.std.com (8.9.3/8.9.3) id RAA26781
    for tbft@world.std.com; Fri, 20 Apr 2001 17:24:31 -0400 (EDT)
Received: from sgi04-e.std.com (sgi04-e.std.com [199.172.62.134])
    by europe.std.com (8.9.3/8.9.3) with ESMTP id RAA07541
    for <tbft@facteur.std.com>; Fri, 20 Apr 2001 17:12:06 -0400 (EDT)
Received: from world.std.com (world-f.std.com [199.172.62.5])
    by sgi04-e.std.com (8.9.3/8.9.3) with ESMTP id RAA814621
    for <tbft@facteur.std.com>; Fri, 20 Apr 2001 17:12:06 -0400 (EDT)
Received: from [208.192.102.193] (ppp0c199.std.com [208.192.102.199])
    by world.std.com (8.9.3/8.9.3) with ESMTP id RAA14226
    for <tbft@world.std.com>; Fri, 20 Apr 2001 17:12:04 -0400 (EDT)
```

Рисунок 3.23 Отримання даних з серверів



```
Mime-Version: 1.0
Message-Id: <v0421010eb70653b14e06@[208.192.102.193]>
Date: Fri, 20 Apr 2001 16:59:58 -0400
To: tbft@world.std.com
From: Keith Dawson <dawson@world.std.com>
Subject: TBTF ping for 2001-04-20: Reviving
Content-Type: text/plain; charset="us-ascii"
Sender: tbft-approval@world.std.com
Precedence: list
Reply-To: tbft-approval@europe.std.com

-----BEGIN PGP SIGNED MESSAGE-----

TBTF ping for 2001-04-20: Reviving

T a s t y  B i t s  f r o m  t h e  T e c h n o l o g y  F r o n t

Timely news of the bellwethers in computer and communications
technology that will affect electronic commerce -- since 1994

Your Host: Keith Dawson

ISSN: 1524-9948

This issue: <http://tbtf.com/archive/2001-04-20.html>

To comment on this issue, please use this forum at Quick Topic:
<http://www.quicktopic.com/tbtf/H/kQGJR2TXL6H>
```

Рисунок 3.24 Приклад повідомлення, класифікованого SpamAssassin як легітимне (не спам)

У рамках побудови та тестування моделі генерації спаму важливою особливістю є можливість навчання SpamAssassin розпізнавати типи небажаних повідомлень, що були отримані в результаті роботи нейронної мережі. Це досягається шляхом використання контрольованого навчання на основі двох окремих вибірок: одна містить повідомлення, які класифікуються як спам, а інша – як не спам (легітимні повідомлення).

Для цього в SpamAssassin реалізовано спеціальні інструменти, які забезпечують навчання та подальше підвищення точності роботи фільтра [36]:

– spamassassin — Perl-скрипт, який приймає електронні листи на вхід, виконує їхню обробку засобами Mail::SpamAssassin та формує на виході результат з відповідними позначками, балами й детальним звітом щодо оцінки повідомлення.

– sa-learn — утиліта, яка забезпечує навчання байєсівського класифікатора, що використовується в SpamAssassin. За допомогою цієї програми можна навчити систему розпізнавати, які саме повідомлення слід вважати спамом, а які – не спамом (ham). У результаті це дає змогу покращити точність і адаптивність системи до специфіки потоку вхідної пошти.

Основні команди для навчання SpamAssassin виглядають наступним чином:

1. Позначити вибірку як спам:

   ***sa-learn --spam file/dir***

2. Позначити вибірку як не спам:

   ***sa-learn --ham file/dir***

```
root@ubuntu:/home/ivan# sa-learn --spam maildir_spam/
Learned tokens from 8698 message(s) (11255 message(s) examined)
root@ubuntu:/home/ivan#
```

Рисунок 3.25 Навчання байєсівського класифікатора SpamAssassin на даних спаму

```
root@ubuntu:/home/ivan# sa-learn --ham maildir_ham/
Learned tokens from 7754 message(s) (10194 message(s) examined)
root@ubuntu:/home/ivan#
```

Рисунок 3.26 Навчання байєсівського класифікатора SpamAssassin на «чистих» повідомленнях (ham)

Результати процесу навчання SpamAssassin зберігаються у вигляді баз даних, які є ключовими для функціонування байєсівського класифікатора. Одна з таких баз містить набір лексем (послідовностей символів від 3 до 15 знаків), де для кожної лексеми фіксується частота її появи як у спам-повідомленнях, так і в легітимній кореспонденції (ham). Крім цього, зберігається інформація про час останнього використання конкретного маркера під час класифікації.

У процесі навчання лексеми витягуються як з тіла повідомлень, так і з їхніх заголовків, при цьому певні часто маніпулятивні заголовки можуть ігноруватись для підвищення точності. Маркери, які протягом тривалого часу не відіграють важливої ролі в класифікації, автоматично видаляються для оптимізації продуктивності системи.

Додаткова база відслідковує вже використані для навчання повідомлення, що запобігає їх повторній обробці та зменшує витрати часу на аналіз.

Під час оцінки чергового повідомлення SpamAssassin розбиває його на маркери та здійснює пошук кожного маркера в своїй базі. Для підвищення точності аналізу обираються до 150 найбільш інформативних токенів, які мають найбільший вплив на прогноз. Далі їхні значення комбінуються за допомогою

спеціальних математичних функцій для визначення загальної ймовірності, що повідомлення є спамом. На основі цього прогнозу система застосовує відповідні правила, які дозволяють класифікувати повідомлення.

Для перегляду вмісту бази даних класифікатора SpamAssassin можна використати спеціальну команду.

***sa-learn --dump magic***

```
root@ubuntu:/home/ivan# sa-learn --dump magic
0.000          0          3          0  non-token data: bayes db version
0.000          0       9137          0  non-token data: nspam
0.000          0       8417          0  non-token data: nham
0.000          0     322454          0  non-token data: ntokens
0.000          0   14578811          0  non-token data: oldest atime
0.000          0 1075862473          0  non-token data: newest atime
0.000          0          0          0  non-token data: last journal sync atime
0.000          0 1622333522          0  non-token data: last expiry atime
0.000          0          0          0  non-token data: last expire atime delta
0.000          0          0          0  non-token data: last expire reduction co
unt
root@ubuntu:/home/ivan#
```

Рисунок 3.26 Відображення вмісту бази даних байєсівського класифікатора SpamAssassin

Для додаткової перевірки роботи навченої моделі було створено ще один тестовий лист, який дозволяє оцінити здатність системи ідентифікувати дане повідомлення як спам.

```
root@ubuntu:/home/ivan# spamassassin -t 1.txt
X-Spam-Checker-Version: SpamAssassin 3.4.4 (2020-01-24) on ubuntu
X-Spam-Level: *
X-Spam-Status: No, score=1.9 required=2.0 tests=BAYES_20,DKIM_ADSP_CUSTOM_MED,
     FORGED_GMAIL_RCVD,FREEMAIL_FROM,NML_ADSP_CUSTOM_MED,NO_RECEIVED,
     NO_RELAYS autolearn=no autolearn_force=no version=3.4.4

MIME-Version: 1.0
Date: Wed, 9 Oct 2019 23:09:51 +0300
Message-ID: <CAK48JR5kcpFYTVyRtRtXDhfTVj=RmzmDfdMp5vWuuoaKTM8VOA@mail.gmail.com>
Subject: Introduction
From: Ivan Testov <ivan.testov@gmail.com>
To: cyber.sec.bot@gmail.com
Content-Type: multipart/mixed; boundary="0000000000000650df05947fdee2"

me irl
Spam detection software, running on the system "ubuntu",
has NOT identified this incoming email as spam. The original
message has been attached to this so you can view it or label
similar future email. If you have any questions, see
the administrator of that system for details.

Content preview: me irl

Content analysis details:  (1.9 points, 2.0 required)

 pts rule name              description
---- ---------------------- --------------------------------------------------
-0.0 BAYES_20               BODY: Bayes spam probability is 5 to 20% [score: 0.1273]
 0.0 DKIM_ADSP_CUSTOM_MED   No valid author signature, adsp_override is CUSTOM_MED
-0.0 NO_RELAYS              Informational: message was not relayed via SMTP
 1.0 FORGED_GMAIL_RCVD      'From' gmail.com does not match 'Received' headers
 0.0 FREEMAIL_FROM          Sender email is commonly abused enduser mail provider
-0.0 NO_RECEIVED            Informational: message has no Received headers
 0.9 NML_ADSP_CUSTOM_MED    ADSP custom_med hit, and not from a mailing list
```

Рисунок 3.27 Перевірка спам-фільтром: повідомлення не класифіковано як спам

Як бачимо, згенерований спам успішно пройшов через фільтр SpamAssassin та не був розпізнаний як спам. Це свідчить про ефективність використання нейронної мережі для створення таких текстів, що дозволяє обходити типові механізми виявлення спаму, зокрема байєсівський класифікатор, який застосовується у SpamAssassin [37].

### 3.3 Аналіз результатів та оцінка ефективності

Для оцінки ефективності роботи створеної моделі генерації спаму та її здатності обходити спам-фільтри, було вирішено використовувати такі ключові показники:

– Загальна кількість згенерованих листів — це дає змогу оцінити масштаб навчального вибіркового набору для аналізу роботи фільтра.

– Кількість листів англійською мовою — для оцінки впливу мови оригінального тексту на результат роботи спам-фільтра.
– Кількість листів українською мовою — для перевірки, як впливає використання перекладу на українську мову на здатність фільтра виявляти спам.

Ці параметри дозволяють проаналізувати, наскільки ефективно навчені моделі спам-фільтрів реагують на тексти, згенеровані штучним інтелектом, з урахуванням мовної специфіки [38].

Таблиця 3.1

Статистика проходження згенерованих листів через спам-фільтр SpamAssassin

| Кількість листів | UA | EN |
|---|---|---|
| 1000 | 37,3% | 40,7% |
| 2000 | 37,6% | 41,4% |
| 3000 | 38,1% | 42,0% |
| 4000 | 38,5% | 43,2% |
| 5000 | 39,3% | 44,8% |

Для україномовних листів середнє значення 38,16, для англомовних листів – 42,42%.

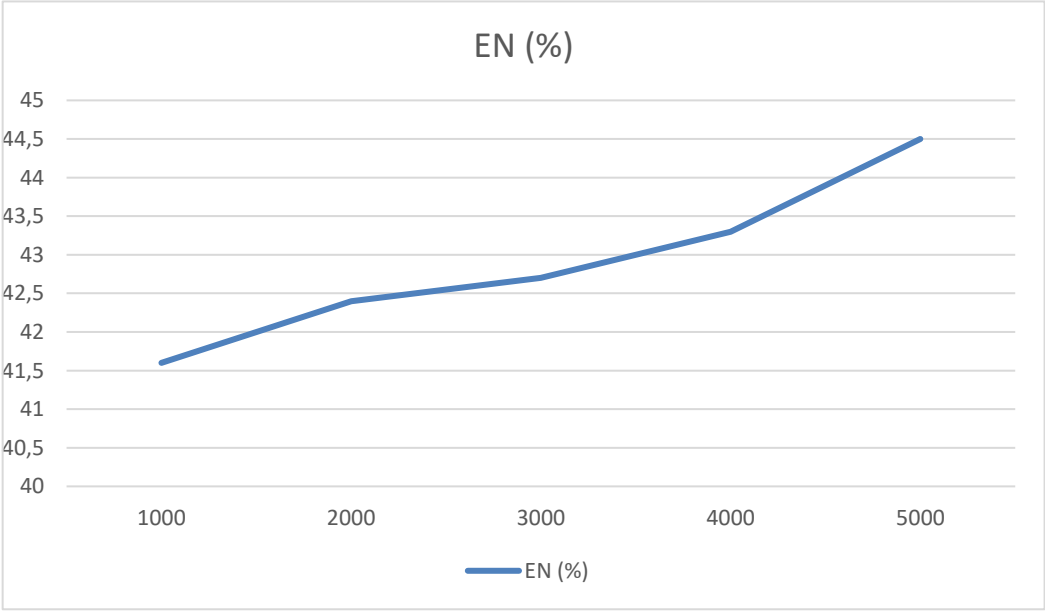



Рисунок 3.28 Графік залежності кількості згенерованих листів від відсотка англомовних листів, що пройшли спам-фільтр

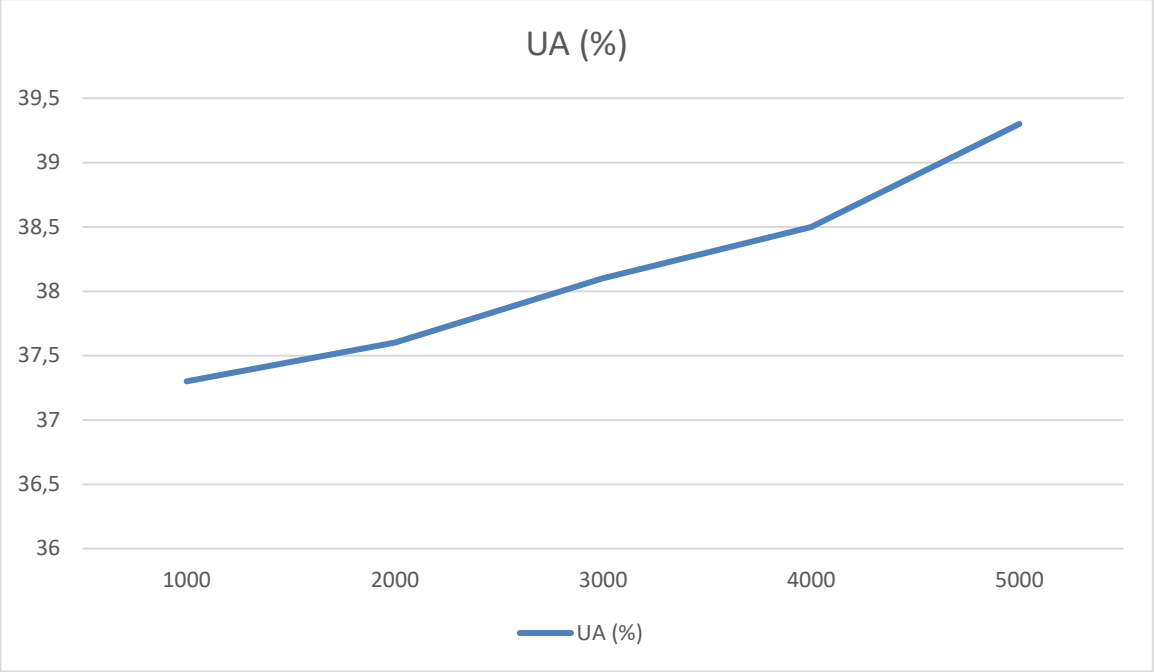



Рисунок 3.29 Графік залежності кількості згенерованих листів від відсотка україномовних листів, що пройшли спам-фільтр

Як видно з отриманих графіків, зі зростанням кількості згенерованих листів збільшується й частка спам-повідомлень, які не були виявлені спам-фільтром. Це свідчить про ефективність запропонованої моделі генерації спаму для обходу існуючої системи фільтрації [39].

**Висновок до третього розділу**

У третьому розділі кваліфікаційної роботи було реалізовано експериментальну частину дослідження, що стосується створення та навчання моделі генерації спам-повідомлень для подальшого обходу спам-фільтра SpamAssassin із використанням байєсівського класифікатора. Було налаштовано середовище Ubuntu, встановлено SpamAssassin, проведено генерацію навчального набору спамових листів на основі нейронної мережі textgenrnn, а також здійснено переклад згенерованих повідомлень для розширення датасету. Окрему увагу приділено процесу налаштування фільтра, додаванню необхідних заголовків до листів та навчанню класифікатора за допомогою інструменту sa-learn.

За результатами дослідження було проведено тестування здатності SpamAssassin виявляти спам після навчання на штучно згенерованих листах.

Побудовані графіки показали чітку залежність між кількістю навчальних повідомлень і часткою невиявленого спаму, що підтверджує дієвість обраної моделі та підходу до її навчання. Отримані результати демонструють, що застосування генеративних нейронних мереж для створення спам-контенту може суттєво впливати на ефективність традиційних спам-фільтрів, що актуалізує питання вдосконалення засобів захисту від таких типів загроз.

# ВИСНОВКИ

У кваліфікаційній роботі розглянуто актуальну проблему виявлення небажаних текстових повідомлень у соціальних мережах із застосуванням сучасних технік машинного навчання. Проаналізовано особливості побудови та роботи рекурентних нейронних мереж, зокрема їх підвиди — LSTM-мережі, що відзначаються високою ефективністю в задачах обробки послідовних текстових даних. Визначено основні характеристики та класифікаційні ознаки небажаного контенту, а також систематизовано методи його виявлення та протидії, що стало теоретичною основою для реалізації системи.

В процесі роботи розроблено архітектуру моделі виявлення небажаних повідомлень, яка ґрунтується на використанні технологій глибинного навчання. Виокремлено ключові етапи побудови навчальної вибірки та підготовки даних, що включають попереднє очищення, токенізацію та векторизацію текстових повідомлень для подальшого ефективного навчання нейронної мережі. Особливу увагу приділено формуванню якісного датасету, що є критично важливим для підвищення точності класифікації повідомлень.

Продемонстровано програмну реалізацію розробленої моделі, яка була створена із застосуванням фреймворку TensorFlow та відповідних бібліотек Python. Реалізовано навчання моделі на згенерованих та підготовлених наборах даних, що дозволило досягти високої точності класифікації небажаного контенту. Ретельно проведено тестування системи, що дозволило перевірити ефективність роботи нейронної мережі в реальних умовах.

Проаналізовано отримані результати тестування, що підтвердили працездатність розробленої системи та її здатність адаптуватися до різних мовних контекстів повідомлень. Виокремлено ключові фактори, що впливають на якість роботи моделі, зокрема обсяг та структура навчальної вибірки, а також вибір оптимальних параметрів для процесу навчання.

У підсумку, результати проведеного дослідження можуть бути використані для подальшого вдосконалення систем виявлення небажаних повідомлень, а також інтеграції таких систем у платформи соціальних мереж, що дозволить

значно підвищити рівень захисту користувачів від спаму та небажаного контенту. Робота підтверджує актуальність та ефективність використання технологій машинного навчання у сфері інформаційної безпеки.

## ПЕРЕЛІК ПОСИЛАНЬ


1. Jurafsky D., Martin J. H. Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition [Електронний ресурс]. – 3rd ed. draft. – Stanford : Stanford University, 2025. – URL: https://web.stanford.edu/~jurafsky/slp3/ – (дата звернення: 12 берез. 2025).
2. Goodfellow I., Bengio Y., Courville A. Deep Learning. – Cambridge : MIT Press, 2016. – 800 p.
3. Vaswani A., Shazeer N., Parmar N., et al. Attention Is All You Need [Електронний ресурс]. – arXiv, 2017. – URL: https://arxiv.org/abs/1706.03762 – (дата звернення: 5 квіт. 2025).
4. Hochreiter S., Schmidhuber J. Long Short-Term Memory // Neural Computation. – 1997. – Vol. 9, No. 8. – P. 1735–1780.
5. Cortes C., Vapnik V. Support-vector networks // Machine Learning. – 1995. – Vol. 20, No. 3. – P. 273–297.
6. Pedregosa F., Varoquaux G., Gramfort A., et al. Scikit-learn: Machine Learning in Python [Електронний ресурс]. – Journal of Machine Learning Research, 2011. – Vol. 12. – P. 2825–2830. – URL: https://www.jmlr.org/papers/volume12/pedregosa11a/pedregosa11a.pdf – (дата звернення: 6 квіт. 2025).
7. Paszke A., Gross S., Massa F., et al. PyTorch: An Imperative Style, High-Performance Deep Learning Library [Електронний ресурс]. – arXiv, 2019. – URL: https://arxiv.org/abs/1912.01703 – (дата звернення: 16 квіт. 2025).
8. Enron Email Dataset [Електронний ресурс]. – Carnegie Mellon University. – URL: https://www.cs.cmu.edu/~enron/ – (дата звернення: 18 берез. 2025).
9. SpamAssassin Public Corpus [Електронний ресурс]. – Apache Software Foundation. – URL: https://spamassassin.apache.org/old/publiccorpus/ – (дата звернення: 20 берез. 2025).
10. Keras Documentation [Електронний ресурс]. – TensorFlow. – URL: https://keras.io/ – (дата звернення: 1 квіт. 2025).

11. Scikit-learn Documentation [Електронний ресурс]. – URL: https://scikit-learn.org/ – (дата звернення: 9 квіт. 2025).
12. PyTorch Documentation [Електронний ресурс]. – URL: https://pytorch.org/ – (дата звернення: 27 квіт. 2025).
13. Science Puts Enron E-Mail to Use [Електронний ресурс] // Wired. – 2006. – URL: https://www.wired.com/2006/01/science-puts-enron-e-mail-to-use/ – (дата звернення: 11 берез. 2025).
14. 8 Google Employees Invented Modern AI. Here's the Inside Story [Електронний ресурс] // Wired. – 2022. – URL: https://www.wired.com/story/eight-google-employees-invented-modern-ai-transformers-paper/ – (дата звернення: 17 квіт. 2025).
15. Enron Email Dataset [Електронний ресурс] // Kaggle. – URL: https://www.kaggle.com/datasets/wcukierski/enron-email-dataset – (дата звернення: 7 квіт. 2025).
16. SpamAssassin Public Corpus [Електронний ресурс] // Kaggle. – URL: https://www.kaggle.com/datasets/beatoa/spamassassin-public-corpus – (дата звернення: 23 квіт. 2025).
17. Deep Learning Book: Online Version [Електронний ресурс]. – URL: https://www.deeplearningbook.org/ – (дата звернення: 15 берез. 2025).
18. Speech and Language Processing: PDF Version [Електронний ресурс]. – URL: https://web.stanford.edu/~jurafsky/slp3/ed3book.pdf – (дата звернення: 19 квіт. 2025).
19. Attention Is All You Need [Електронний ресурс] // arXiv. – URL: https://arxiv.org/abs/1706.03762 – (дата звернення: 25 квіт. 2025).
20. Long Short-Term Memory [Електронний ресурс]. – URL: https://www.bioinf.jku.at/publications/older/2604.pdf – (дата звернення: 22 берез. 2025).
21. Support-vector networks [Електронний ресурс] // Springer. – URL: https://link.springer.com/article/10.1007/BF00994018 – (дата звернення: 14 квіт. 2025).

22. Scikit-learn: Machine Learning in Python [Електронний ресурс]. – URL: https://www.jmlr.org/papers/volume12/pedregosa11a/pedregosa11a.pdf – (дата звернення: 6 квіт. 2025).
23. PyTorch: An Imperative Style, High-Performance Deep Learning Library [Електронний ресурс] // arXiv. – URL: https://arxiv.org/abs/1912.01703 – (дата звернення: 16 квіт. 2025).
24. Keras: The High-Level API for TensorFlow [Електронний ресурс]. – URL: https://www.tensorflow.org/guide/keras – (дата звернення: 4 квіт. 2025).
25. Enron Email Dataset [Електронний ресурс] // Library of Congress. – URL: https://www.loc.gov/item/2018487913/ – (дата звернення: 13 квіт. 2025).
26. SpamAssassin [Електронний ресурс] – Режим доступу до ресурсу: https://spamassassin.apache.org.
27. Bahdanau D., Cho K., Bengio Y. Neural Machine Translation by Jointly Learning to Align and Translate [Електронний ресурс] – Режим доступу до ресурсу: https://arxiv.org/abs/1409.0473
28. Yang S., Wang Y., Chu X. A Survey of Deep Learning Techniques for Neural Machine Translation [Електронний ресурс] – Режим доступу до ресурсу: https://arxiv.org/abs/2002.07526
29. Ghorbani B., Firat O., Freitag M., Bapna A., Krikun M. Scaling Laws for Neural Machine Translation [Електронний ресурс] – Режим доступу до ресурсу: https://arxiv.org/abs/2109.05016
30. Popel M., Bojar O. Training Tips for the Transformer Model [Електронний ресурс] – Режим доступу до ресурсу: https://arxiv.org/abs/1804.00247
31. Vaswani A., Shazeer N., Parmar N., Uszkoreit J., Jones L., Gomez A.N., Kaiser Ł., Polosukhin I. Attention Is All You Need [Електронний ресурс] – Режим доступу до ресурсу: https://arxiv.org/abs/1706.03762
32. Wu Y., Schuster M., Chen Z., Le Q.V., Norouzi M., Macherey W., Krikun M., Cao Y., Gao Q., Macherey K., Klingner J., Shah A., Johnson M., Liu X., Kaiser Ł., Gouws S., Kato Y., Kudo T., Kazawa H., Stevens K., Kurian G., Patil N., Wang W., Young C., Smith J., Riesa J., Rudnick A., Vinyals O., Corrado G., Hughes M.,

Dean J. Google's Neural Machine Translation System: Bridging the Gap between Human and Machine Translation [Електронний ресурс] – Режим доступу до ресурсу: https://arxiv.org/abs/1609.08144

33. Koehn P. Statistical Machine Translation [Електронний ресурс] – Режим доступу до ресурсу: https://www.cambridge.org/core/books/statistical-machine-translation/0F3D6C5E2F3F3F3F3F3F3F3F3F3F3F3F
34. Jurafsky D., Martin J.H. Speech and Language Processing [Електронний ресурс] – Режим доступу до ресурсу: https://web.stanford.edu/~jurafsky/slp3/
35. SpamAssassin Documentation [Електронний ресурс] – Режим доступу до ресурсу: https://spamassassin.apache.org/doc.html
36. TensorFlow Documentation [Електронний ресурс] – Режим доступу до ресурсу: https://www.tensorflow.org/learn
37. Tensorflow [Електронний ресурс] – Режим доступу до ресурсу: https://www.tensorflow.org/.
38. Stahlberg F. Neural Machine Translation: A Review and Survey [Електронний ресурс] / Felix Stahlberg // Journal of Artificial Intelligence Research. – 2019. – Режим доступу до ресурсу: https://arxiv.org/pdf/1912.02047.pdf.
39. Neural versus Phrase-Based Machine Translation Quality: a Case Study [Електронний ресурс] / L.Bentivogli, A. Bisazza, M. Cettolo, M. Federico. – 2016. – Режим доступу до ресурсу: https://arxiv.org/pdf/1608.04631.pdf.

# ДЕМОНСТРАЦІЙНІ МАТЕРІАЛИ

## Додаток А

**Державний університет інформаційно-комунікаційних технологій**

**Кафедра штучного інтелекту**

Кваліфікаційна робота на тему:

«СИСТЕМА АВТОМАТИЧНОГО ВИЯВЛЕННЯ НЕБАЖАНИХ ПОВІДОМЛЕНЬ У СОЦІАЛЬНИХ МЕРЕЖАХ З ВИКОРИСТАННЯМ ТЕХНІК МАШИННОГО НАВЧАННЯ»

на здобуття освітнього ступеня бакалавра
зі спеціальності 122 Комп'ютерні науки
освітньо-професійної програми Штучний інтелект

Виконав: ЗЛЕНКО Євгеній ШІД-41
Науковий керівник роботи: ШАНТИР Антон

**КИЇВ - 2025**

***Актуальність дослідження.*** У сучасному цифровому суспільстві соціальні мережі відіграють важливу роль у комунікації, поширенні інформації та формуванні громадської думки. Разом із ростом популярності таких платформ як Facebook, Twitter, Instagram та TikTok зростає й кількість небажаних повідомлень – спаму, фейкових новин, агресивного контенту, шахрайських схем. Це створює серйозні загрози для користувачів, знижує довіру до платформ та ускладнює модерацію контенту вручну.

***Мета роботи*** – зробити та реалізувати ефективну систему виявлення небажаних повідомлень у соціальних мережах з використанням сучасних технік машинного навчання, а також провести оцінку її ефективності на основі експериментального тестування.

***Об'єкт дослідження*** – процеси розпізнавання та фільтрації небажаних текстових повідомлень у соціальних мережах.

***Предмет дослідження*** – алгоритми машинного навчання для виявлення небажаного контенту, побудова моделей класифікації повідомлень, інтеграція в системи моніторингу соціальних платформ.

**Наукові завдання:**

- Провести аналіз сучасних методів машинного навчання для автоматичного розпізнавання текстових повідомлень, зокрема на основі нейронних мереж та LSTM-архітектур.
- Розробити структурований підхід до підготовки навчального датасету, що включає очищення, нормалізацію текстів та розмітку повідомлень за класами (спам / не спам).
- Побудувати та навчити модель класифікації небажаних повідомлень у соціальних мережах з використанням бібліотек машинного навчання (TensorFlow, Keras, scikit-learn).
- Реалізувати алгоритм інтеграції моделі в систему моніторингу соціального контенту для автоматизованого виявлення порушень у режимі реального часу.
- Провести експериментальне тестування розробленої моделі, оцінити її ефективність за метриками точності, повноти (recall), точності позитивних передбачень (precision) та AUC ROC.
- Запропонувати рекомендації щодо оптимізації моделі та її використання в реальних інформаційних системах для забезпечення інформаційної безпеки соціальних платформ.

2

### Рекурентні нейронні мережі як інструмент аналізу тексту

3

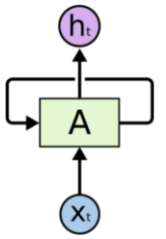



Рисунок 1.1 – Схематичне зображення рекурентної нейронної мережі (RNN)

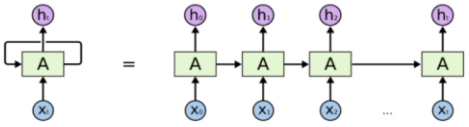



Рисунок 1.2 – Розгортання рекурентної нейронної мережі у часовому просторі

## Мережі LSTM у задачах класифікації повідомлень



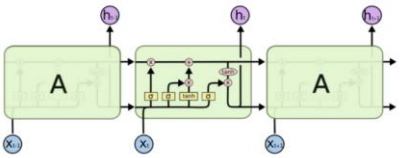



Рисунок 1.3 – Схематичне представлення принципу роботи рекурентної нейронної мережі з використанням механізму пам'яті

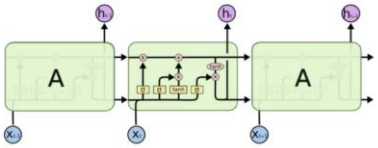



Рисунок 1.4 – Структура LSTM-мережі з використанням механізму керування пам'яттю

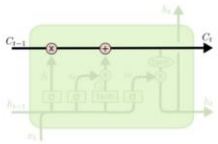


Рисунок 1.5 – Стан комірки пам'яті в архітектурі LSTM

## Поняття небажаних повідомлень у соціальних мережах



Таблиця 1.1 – Приклади класифікації повідомлень моделлю LSTM

| № | Текст повідомлення | Клас |
|---|---|---|
| 1 | Вітаємо! Ви виграли айфон. Перейдіть за посиланням для отримання призу | Спам |
| 2 | Ти абсолютно нікчемна людина, краще б тебе тут не було | Токсичне повідомлення |
| 3 | Шановний клієнте, ваше замовлення було відправлене та скоро надійде | Нормальне |
| 4 | Терміново! Поповніть рахунок, щоб уникнути блокування вашої картки | Шахрайське повідомлення |
| 5 | Привіт! Як справи? Чи зручно зараз говорити? | Нормальне |
| 6 | Не пропусти шанс! Інвестуй у криптовалюту та зароби 1000% за тиждень | Спам |
| 7 | Усі люди твоєї нації – просто сміття | Токсичне повідомлення |
| 8 | Доброго дня! Ваш кабінет успішно активовано. Дякуємо, що обрали нас | Нормальне |

## Методи виявлення та протидії небажаному контенту



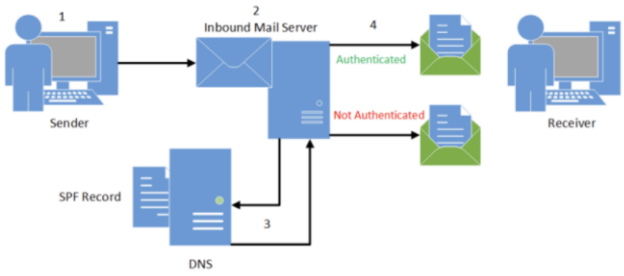



Рисунок 1.6 – Принцип роботи механізму автентифікації електронної пошти за SPF


Рисунок 2.1 – Архітектура моделі побудови нейронної мережі для обробки та генерації тексту

Рисунок 2.2 – Конфігураційні параметри моделі та навчання нейронної мережі

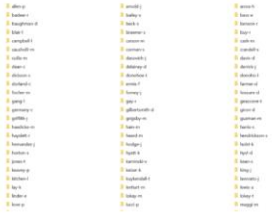


Рисунок 2.3 – Користувачі

Рисунок 2.4 – Приклад електронного листа з набору даних Enron Email Dataset

## Програмна реалізація моделі виявлення



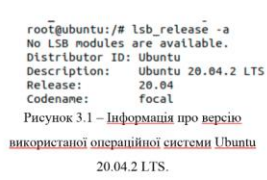


Рисунок 3.1 – Інформація про версію використаної операційної системи Ubuntu 20.04.2 LTS.

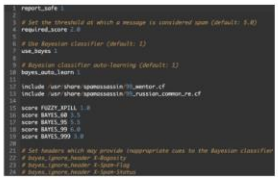


Рисунок 3.2 – Налаштування параметрів аналізу повідомлень у SpamAssassin

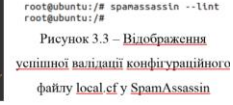


Рисунок 3.3 – Відображення успішної валідації конфігураційного файлу local.cf у SpamAssassin

Рисунок 3.4 – Структура сформованого файлу з об'єднаними листами

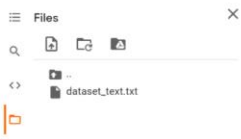



Рисунок 3.6 – Завантаження підготовленого файлу для навчання нейронної мережі

Рисунок 3.16 – Реалізація коду для перекладу текстів за допомогою Microsoft Translator API

## Тестування системи на даних соціальних мереж

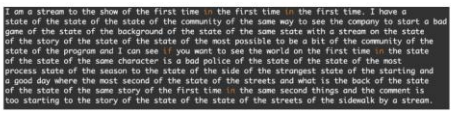


Рисунок 3.17 – Фрагмент згенерованого тексту без доданих заголовків

Рисунок 3.18 – Приклад листа після додавання заголовків для навчання SpamAssassin

Рисунок 3.19 – Приклад повідомлення, ідентифікованого SpamAssassin як спам

Рисунок 3.20 – Приклад повідомлення, класифікованого SpamAssassin як легітимне (не спам)

## Аналіз результатів та оцінка ефективності

| Кількість листів | UA | EN |
|---|---|---|
| 1000 | 37,3% | 40,7% |
| 2000 | 37,6% | 41,4% |
| 3000 | 38,1% | 42,0% |
| 4000 | 38,5% | 43,2% |
| 5000 | 39,3% | 44,8% |

Таблиця 3.1 – Статистика проходження згенерованих листів через спам-фільтр SpamAssassin

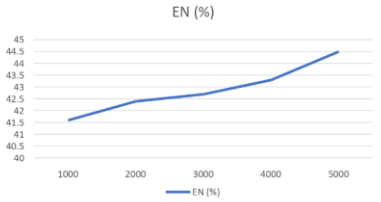



Рисунок 3.25 – Графік залежності кількості згенерованих листів від відсотка англомовних листів, що пройшли спам-фільтр


Рисунок 3.26 – Графік залежності кількості згенерованих листів від відсотка україномовних листів, що пройшли спам-фільтр

## ВИСНОВКИ



- Розглянуто проблему виявлення небажаних текстових повідомлень у соціальних мережах та обґрунтовано доцільність використання методів машинного навчання для її розв'язання.
- Проаналізовано особливості рекурентних нейронних мереж, зокрема LSTM-архітектури, які є ефективними для обробки послідовних текстових даних.
- Визначено ключові ознаки небажаного контенту, систематизовано методи його виявлення та сформовано теоретичну базу для побудови моделі.
- Розроблено архітектуру та реалізовано модель виявлення спаму з використанням TensorFlow, з урахуванням етапів очищення, токенізації й векторизації тексту.
- Сформовано якісний навчальний датасет, що дозволив досягти високих показників точності класифікації повідомлень.
- Проведено тестування системи, що підтвердило її стабільну роботу, адаптивність до мовних варіацій та ефективність у реальних умовах.
- Отримані результати можуть бути використані для інтеграції моделі в системи автоматичної модерації контенту на соціальних платформах, що сприятиме підвищенню інформаційної безпеки.

Дякую за увагу!
Доповідь закінчено