Issue 15, 2024

https://doi.org/10.23939/sisn2024.15.102

УДК 004

УДОСКОНАЛЕННЯ МЕТОДІВ ЗБЕРІГАННЯ ТЕКСТОВИХ ДАНИХ

Василь Литвин[1], Артем Каланча[2], Дмитро Угрин[3], Марія Талах[4]

[1] Національний університет "Львівська політехніка",
кафедра інформаційних систем та мереж, Львів, Україна
[2–4] Чернівецький національний університет імені Юрія Федьковича,
кафедра комп'ютерних наук, Чернівці, Україна
[1] E-mail: vasyl.v.lytvyn@lpnu.ua, ORCID: 0000-0002-9676-0180
[2] E-mail: kalancha.artem@chnu.edu.ua, ORCID: 0009-0004-1451-7470
[3] E-mail: d.ugryn@chnu.edu.ua, ORCID: 0000-0003-4858-4511
[4] E-mail: m.talah@chnu.edu.ua, ORCID: 0000-0002-5067-6848

У дослідженні проаналізовано якісні характеристики повідомлень у месенджері Telegram, використаних як вихідні дані для подальшого аналізу текстового контенту. Здійснено ретельний огляд параметрів цих повідомлень, таких як їх формат, розмір, наявність шумів та швидкодія. Основна мета статті – моделювання оптимального підходу до збереження великого обсягу даних перед важливим етапом аналізу тексту. Під час дослідження детально проаналізовано літературні джерела із цієї тематики. Розглянуто основні переваги та недоліки наявних алгоритмів переднього опрацювання даних, а також проблеми, пов'язані з чистотою даних і їх впливом на потенційні результати дослідження. У межах програмних експериментів оцінено вплив попереднього опрацювання даних на розмір збережених даних для подальшого використання, а також на швидкість генерації вхідних даних. Серед запропонованих методів виділено метод збереження очищених токенів у форматі рядка та метод збереження кодів слів у форматі рядка разом зі словником слово-код, використання яких дасть змогу забезпечити ефективний розподіл завдань системи аналізу текстів протягом періоду доби.

Ключові слова: текстовий аналіз; попередня обробка тексту; база даних; кодування.

Постановка проблеми

Для алгоритмів опрацювання природних мов [1, 2] основним джерелом інформації є текст, форми та стилістика якого можуть відрізнятися, передаючи смислове навантаження людської мови. Часто для навчання таких моделей використовують масив текстів із різними характеристиками, такими як стилістика, мова та об'єм. Кожен елемент цього масиву може мати додаткові метадані, такі як ім'я автора, дата і час, мова тощо. Навчання може відбуватися як одномоментно, так і неперервно.

За одномоментного навчання зазвичай використовують попередньо створений набір даних у форматі CSV, який залишається незмінним з часом. У другому випадку дані постійно надходять через спеціальні канали та потребують подальшого опрацювання та взаємодії з наявною моделлю.

Повний цикл опрацювання природних мов передбачає низку етапів, і одним із ключових аспектів є попереднє опрацювання текстових даних. Проте через постійний потік нового текстового матеріалу виникає необхідність у впровадженні додаткових етапів, таких як завантаження та зберігання текстової інформації. У цій роботі детально описано процес завантаження інформації з месенджера Telegram [3],

а також подано обґрунтування методу та формату подальшого зберігання перед основним етапом опрацювання природної мови. Об'єктом дослідження слугують повідомлення новинних Telegram-каналів, які регулярно надають значний обсяг текстового матеріалу для подальшого аналізу.

Формулювання цілі статті

У сфері опрацювання повідомлень в Telegram-каналах актуалізувалось постійне збільшення обсягу інформації. Це потребує уважної оцінки вхідних даних для ефективного проєктування системи з урахуванням наявних ресурсів. Обсяг даних зростає щодня, що ставить перед нами виклик – визначити масштаби та нашу здатність адекватно опрацьовувати всю необхідну текстову інформацію. Систематичний аналіз та адаптація до цього постійного збільшення обсягу даних є ключовим елементом для забезпечення ефективності функціонування системи підтримки прийняття рішень, що ґрунтується на аналізі тексту.

Мета дослідження полягає у моделюванні ефективних стратегій оптимального зберігання текстових даних за допомогою алгоритмів опрацювання природних мов. У межах цього завдання досліджено вплив методів зберігання даних на ефективність наступних етапів опрацювання природної мови, враховуючи розмір, формат та інші характеристики текстів. Дослідження основано на повідомленнях з обмеженого списку Telegram-каналів, зібраних протягом певного часового інтервалу.

Ця стаття аналізує специфічне джерело текстових даних – повідомлення в Telegram-каналах. Використання саме цього джерела даних вважаємо новизною, оскільки раніше такі дані, ймовірно, не аналізували для задач опрацювання природної мови. Досліджено оптимальні стратегії зберігання текстових даних перед подальшим опрацюванням за допомогою алгоритмів дослідження природної мови. Такий аналіз впливу методів зберігання на ефективність наступних етапів може бути новим. Проаналізовано вплив попереднього опрацювання даних на розмір збережених для подальшого використання даних, а також на швидкість генерації вхідних даних. Такий комплексний аналіз кількісних характеристик, ймовірно, раніше не виконували. Розробляється ефективна стратегія оптимального зберігання текстових даних з Telegram-каналів для подальшого опрацювання, що враховує обсяг, формат та інші характеристики текстів.

Попереднє опрацювання даних

Один із критичних етапів опрацювання великих обсягів даних – це попереднє опрацювання даних [4], яка може потребувати значних ресурсів, як програмних, так і часових. Аналіз розміру даних, які проходять через етап попереднього опрацювання, а також швидкість їх опрацювання, дає змогу ефективніше планувати ресурси сховища, де будуть зберігатися необхідні дані для подальшого аналізу, і розробляти оптимальну стратегію попереднього опрацювання текстів, щоб забезпечити мінімальний час виконання операцій.

У контексті оптимізації зберігання текстових даних для подальшого опрацювання алгоритмами дослідження природної мови розглянуто ключові етапи попереднього опрацювання текстової інформації, серед них виокремлення зайвих символів, токенізація, перетворення тексту у нижній регістр, вилучення стоп-слів, лематизація та стемінг.

Під час попереднього аналізу виявлено, що деякі тексти містять велику кількість символів Emoji та чисел. З метою підвищення ефективності аналізу та уникнення проблем зі збереженням тексту [5] ми вирішили відокремити ці символи від подальших аналітичних процесів. Це є необхідно, оскільки ці символи істотно не впливають на зміст і можуть стати джерелом технічних ускладнень. За результатами виконаних експериментів здійснено додатковий аналіз тексту на виявлення неочищеної інформації.

Токенізація – розподіл тексту на окремі одиниці, відомі як токени, що можуть містити слова, фрази, речення тощо. Основна мета токенізації полягає в розділенні тексту на окремі елементи для подальшого аналізу та опрацювання.

Вилучення стоп-слів – це видалення загальних та неінформативних слів, які не дають значущої інформації для конкретного тексту. Це допопмагає зосередитися на ключових словах, підвищуючи точність аналізу тексту.

Лематизація та стемінг – це методи зведення слів до їх базових форм для спрощення подальшого аналізу. Вони допомагають уніфікувати слова, щоб вони являли собою ті самі концепції, незважаючи на різні граматичні форми чи закінчення. Вибір між лематизацією та стемінгом залежить від потреб проєкту та бажаного балансу між точністю та обчислювальною складністю.

Рис. 1. Етапи попереднього опрацювання тексту. Аналіз попереднього опрацювання тексту

Ці етапи попереднього опрацювання не лише оптимізують текст для подальшого використання в аналізі, але й враховують специфічні особливості текстової інформації, такі як символи Emoji, щоб забезпечити точність та релевантність аналітичних результатів.

Аналіз останніх досліджень та публікації

Передові системи попереднього опрацювання тексту отримали значний поштовх завдяки досягненням у галузі машинного навчання та суміжних технологій. Розглянемо використання методів машинного навчання та штучного інтелекту для аналізу особливостей тексту з метою підвищення загальної ефективності систем. Дослідники вже досягли значного прогресу в розробленні та оптимізації алгоритмів попереднього опрацювання текстової інформації. Основна мета цього огляду – досягти глибокого розуміння сучасного стану цієї сфери. Цей всебічний аналіз спрямований на виявлення можливих напрямів для подальшого розвитку та вдосконалення у сфері попереднього опрацювання текстових даних з метою позитивного впливу на результативність системи загалом та оптимізацію витрат часу та інших ресурсів.

У дослідженні Mohammad [6] основну увагу звернено на такі проблеми, як акроніми, посилання та пропущені символи у словах. Ці артефакти часто виникають через неграмотне введення або емоційний стан авторів, що може негативно впливати на якість опрацювання. З цієї причини для таких текстів рекомендують застосовувати методи нечіткої відповідності для коректного визначення правильних лексичних форм слів. У контексті опрацювання текстів, які порушують граматичні правила, важливо правильно визначити власні назви та використовувати їх у відповідній формі. Крім того, автор розглядає аспекти емоційності повідомлень та можливість появи слів із довгими повтореннями символів, що ускладнює процес нормалізації. У висновку дослідження вказано, що попереднє опрацювання обмежено впливає на результати такої роботи, які порушують граматичні правила, і рекомендовано зосередитися на виборі оптимального алгоритму опрацювання з подальшим налаштуванням його параметрів.

Дослідження Camacho-Collados [7] пропонує критичний погляд на відомі алгоритми попереднього опрацювання текстової інформації, зосереджуючись на недоліках приведення слів до трансфор-

мація власних назв та імен у загальноприйняті слова, що може змінювати контекст тексту. Наприклад, назва компанії “Apple” може бути трансформована у “apple”. Дослідження також розглядає негативний вплив лематизації, оскільки вона може недостатньо враховувати синтаксичні відтінки тексту та рідко застосовується у системах, основаних на нейронних мережах. Автор аналізує вплив таких процесів, як токенізація, нормалізація, лематизація та робота з проблемними словами, на результати алгоритмів класифікації тематики тексту та сентиментальної поляризації. У висновку автор вказує, що токенізація істотніше впливає на результати, ніж інші етапи обробки, особливо у випадку текстів загальної тематики. З іншого боку, для спеціалізованих текстів один лише процес токенізації може значно знизити точність порівняно з загальними текстами.

Автори роботи [8] дослідили проблематику передопрацювання коротких повідомлень з соціальної мережі Twitter. Вони розглянули складнощі, пов’язані з існуванням нікнеймів (імен користувачів), гіперпосилань, сленгових висловів, хештегів та вираження емоцій у текстах. Крім того, вони проаналізували подання текстів у формах уніграм, біграм та триграм у контексті аналізу сентиментів цих повідомлень. У результаті дослідження встановлено, що поєднання попереднього опрацювання текстів та подання їх у вигляді уніграм є одним з найефективніших підходів для вирішення поставленої завдання.

У статтях [9–11], які вивчали слов’янські мови, проаналізовано вплив української мови на алгоритм лематизації, а також розроблено метод визначення словосполучень за ключовими словами.

Виклад основного матеріалу

Для виконання експериментів заплановано використовувати віртуальне середовище Anaconda, IDE JupyterLab [12], мову програмування Python [13] та базу даних MongoDB [14]. MongoDB може виявитися зручним для зберігання великого масиву текстів, порівняно з традиційними SQL-системами, з кількох причин. По-перше, гнучка схема MongoDB дає змогу легко зберігати неструктуровані дані, такі як тексти, без необхідності строго визначати структуру. Порівняння реляційних та нереляційних баз даних [15, 16] свідчить, що за допомогою індексації та механізмів пошуку у MongoDB можна ефективніше опрацьовувати великі обсяги текстової інформації. Крім того, можливість горизонтального масштабування дає змогу розподіляти дані на кілька серверів, забезпечуючи оптимальну швидкодію під час роботи з великими обсягами текстового контенту. Функція GridFS MongoDB дає змогу зберігати та отримувати великі бінарні файли, зокрема вектори. Незважаючи на те, що GridFS від MongoDB призначена переважно для зберігання файлів, її також можна використовувати для зберігання великих векторів у вигляді двійкових даних, роблячи цей тип бази даних універсальнішим.

Етапи експерименту такі: спочатку потрібно завантажити інформацію зі сторонніх джерел за допомогою API, після чого зберегти ці дані в документо-орієнтовану базу даних. Потім необхідно оцінити обсяг збереженої інформації, а також проаналізувати середню довжину повідомлень та кількість слів у каналах з поступовим застосуванням алгоритмів попередньої опрацювання текстів. Ці алгоритми передбачають видалення Emoji, слів та пунктуації, а також розділення тексту на токени та видалення стоп-слів.

Після цього необхідно проаналізувати спосіб збереження результатів у базу даних, враховуючи обсяг, швидкість зберігання та попереднього опрацювання текстів. Використовуючи метадані повідомлень за датою публікації, оцінюють швидкість генерування даних протягом декількох днів та окремо доби.

Завершальним етапом є оцінка швидкодії кожного з етапів попереднього опрацювання текстів, а також аналіз можливостей перекладу іноземних текстів для розширення охоплення аналізу.

Подібну схему, але лише для задачі кластеризації за допомогою повідомлень у Twitter, подано у статті “Intelligent Analysis of Ukrainian-language Tweets for Public Opinion Research based on NLP Methods and Machine Learning Technology” [17].

Першим кроком є отримання вхідних даних, що передбачає використання Telegram API [1] для завантаження останніх повідомлень із восьми україномовних каналів за січень 2024 р. Для цього використовують бібліотеку Telethon [18], яка спочатку завантажує всю інформацію в оперативну

пам'ять, а потім зберігає у MongoDB. Результатом роботи утиліти є масив сутностей *Message* з рядками полів: text, row_text, is_reply, datetime, forward, buttons, button_count, file, photo, document, web_preview, audio, voice, video, video_note, gif, sticker, contact, game, geo, invoice, poll, venue, action_entities, via_bot, via_input_bot, client. У контексті месенджера Telegram повідомлення можуть мати різні форми, не тільки текстовий вигляд, а також опитування, зображення, відео, геодані тощо. Тому для аналізу тексту використовують лише певні стовпці Telegram-повідомлення, такі як його ідентифікатор, дата та час, текст повідомлення та автор, що забезпечує достатньо інформації для аналізу. Після збереження "сирої" інформації в базі даних MongoDB можна оцінити обсяг пам'яті, який займає ця інформація. За результатами збереження інформації у сирому вигляді, за наявності 12115 повідомлень, з'ясовано, що обсяг стиснутої інформації становить 3,06 МБ, а оригінальної версії – 6,3 МБ. Середній розмір одного документа становить 519 байтів.

Ураховуючи кількість слів або загальну кількість символів, в сумі для всіх каналів це становить приблизно 234 символи. Втім, для різних каналів довжина повідомлень може істотно варіюватися, від мінімуму – 132 символи до максимуму 366 символів, як проілюстровано на рис. 2.

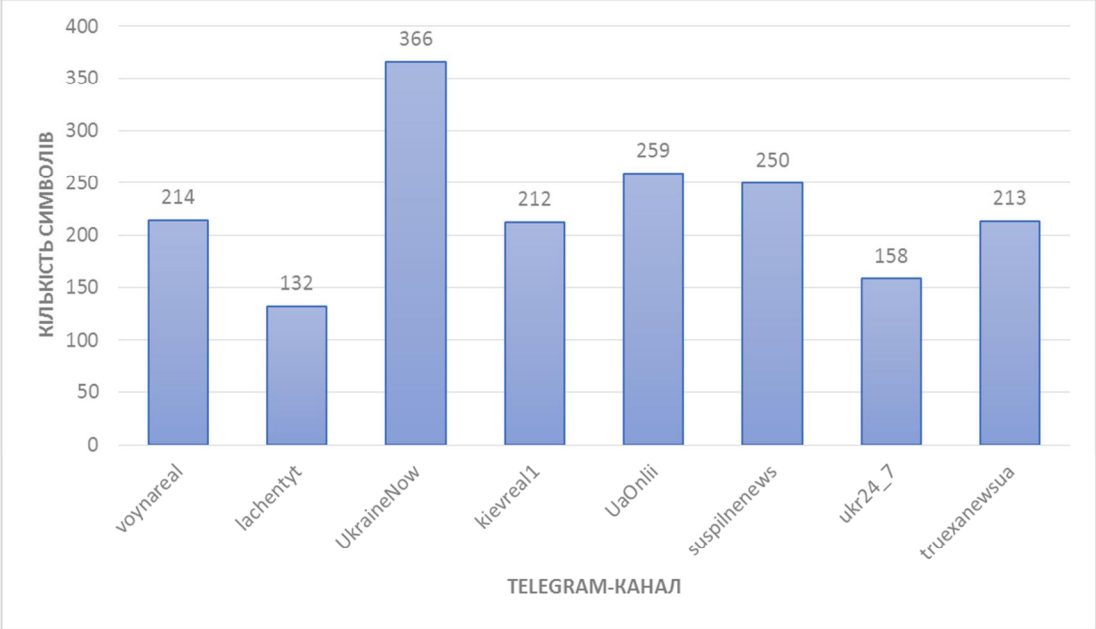

Рис. 2. Середня кількість символів в одному повідомленні для досліджуваних каналів

За наявності значної кількості емоджі у повідомленнях необхідно вилучити їх, що можна зробити за допомогою бібліотеки emoji. Після повторного аналізу довжини повідомлень виявляється, що тепер їх середня довжина становить 232 символів, що на 0,9 % менше, ніж у попередньому варіанті тексту. Навіть цей невеликий обсяг тексту може спричиняти істотні незручності під час оброблення та кодування тексту, а також впливати на розмір збережених даних через проблеми з кодуванням.

Числа також малоінформативні для опрацювання природної мови, їх вилучають за допомогою регулярних виразів. Після цієї процедури оцінка довжини повідомлень показує суттєве зменшення розміру – на 3 %, що дорівнює 229 символів.

Після виконання описаних вище операцій залишається зайвий шум у тексті, який перешкоджає зрозумінню та аналізу. Цей шум може походити з різних джерел, важливо виявити його та видалити для отримання чистої інформації.

Один з типів зайвої інформації – це закінчення відмінюваних числівників після видалення чисел. Наприклад, якщо в оригінальному тексті було слово "121-а", після видалення числа залишиться тільки "-а", яка не має значення для аналізу. Це може стати перешкодою для опрацювання тексту та виявлення

суттєвої інформації. Ще одним джерелом шуму є токени, що складаються лише зі спеціальних символів. Ці токени не мають семантичного навантаження та лише заважають розуміти текст. Вони можуть виникати внаслідок розділення слів або інших механізмів опрацювання тексту. Крім того, у деяких повідомленнях трапляються слова, об'єднані символами '/' та '+', які потрібно роз'єднати для правильного аналізу. Наприклад, у фразі "житомирщину/вінниччину" слова "житомирщину" та "вінниччину" мають різні значення, але разом можуть спотворювати результати аналізу, якщо не будуть роз'єднані. У текстах часто трапляються посилання на вебсайти, не важливі для аналізу тексту. Наприклад, посилання типу "//www.site.com/news/news-1-..." можуть бути сторонніми для автора повідомлення і не мають семантичної цінності для аналізу. Такі посилання можна ігнорувати або вилучати з тексту. Отже, виявлення та видалення зайвої інформації є важливим етапом в опрацюванні тексту.

Після застосування регулярних виразів для очищення текстів від надмірної інформації були видалені всі посилання та всі слова, що складалися з одного алфавітного або спеціального символу. Це привело до зменшення обсягу тексту на додаткові 4 %. Тепер середня довжина текстів становить 220 символів, що сприяє поліпшенню зрозуміння та аналізу текстів. Вилучення зайвої інформації допомогло зробити тексти конкретнішими та кориснішими для подальшого використання.

Для ефективного опрацювання тексту в алгоритмах потрібно розділити його на токени. Токен, по суті, є словом без зайвих пробілів. Розділивши текст на токени, отримуємо інформацію, що в середньому кожне повідомлення містить 34 токени (слова). Нижче наведено діаграму для всіх каналів (рис. 3), на якій проілюстровано незбалансованість серед каналів. Оскільки кожен канал надає різний обсяг тексту у різний час і не має критичних вимог чи обмежень до публікації повідомлень, спостерігається різноманітність обсягів текстів. Якщо порівняти середню довжину повідомлень (рис. 2) з діаграмою (рис. 3), стає очевидною відсутність непропорційності у використанні довгих слів: кількість токенів пропорційно зростає з довжиною тексту. Також виникає питання аналізу занадто коротких повідомлень, зокрема пошуку оптимального мінімального порога кількості токенів та вибору етапу, на якому здійснюється ця фільтрація.

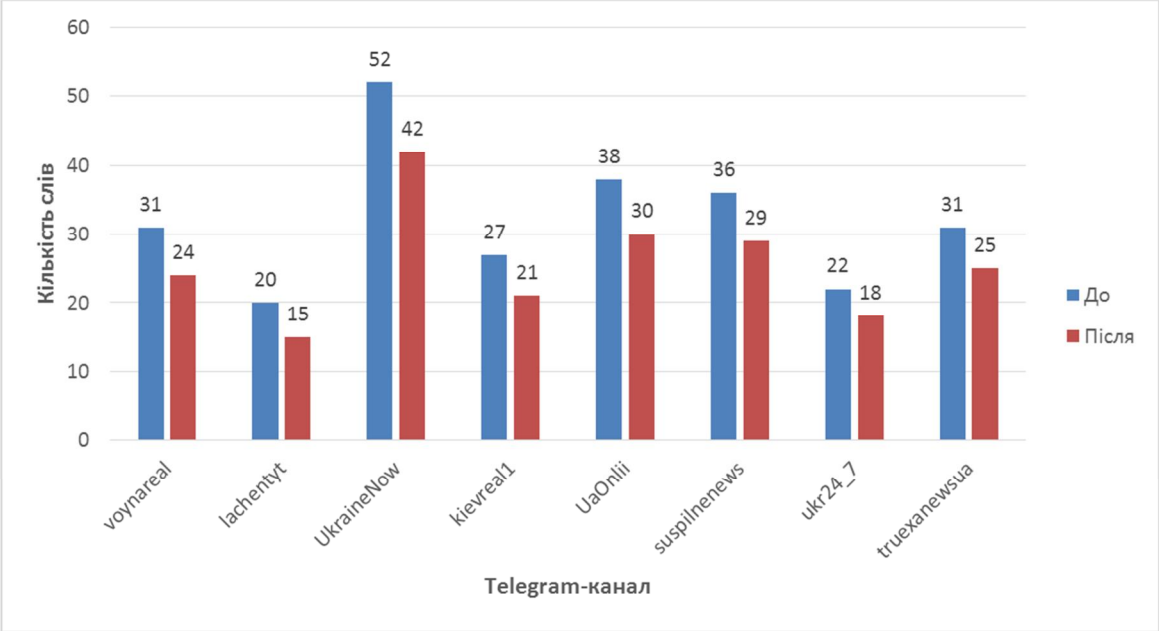

Рис. 3. Діаграма середньої кількості слів за каналом до та після попереднього опрацювання текстів

Серед цих токенів часто трапляються слова, які не містять суттєвої інформації, такі як сполучники, прислівники, вигуки, займенники тощо. Для виконання цієї процедури необхідно мати словник слів, які слід вилучити. Використовуючи цей словник, можна видалити будь-який токен зі загального списку.

Після такої процедури середня кількість токенів зменшилася з 34 до 27, тобто на 21 %. На рис. 3 наведено діаграму для всіх каналів, з якої, порівнявши кількість токенів до та після видалення стоп-слів, можна зробити висновок що відсоток стоп-слів у кожному з каналів наближений до середнього значення 21 %.

Оскільки документо-орієнтована база даних MongoDB дає змогу зберігати цілі списки в одному стовпці об'єкта, ми вирішили спробувати зберегти масив токенів так само, як робили із "сирими" даними. Однак це призвело до збільшення об'єму використаної пам'яті, а саме 3,99 МБ в стисненому вигляді й аж 8,03 МБ в оригінальному, за середнього розміру документа 662 байти. Отже, навіть якщо обсяг видалених непотрібних даних великий, зберігання у вигляді списку потребує значно більше ресурсів пам'яті.

Один зі способів – подати токени у вигляді рядка, об'єднуючи їх за допомогою будь-якого одиничного символу, у цьому випадку – пробілу. Після об'єднання токенів у текст і повторної процедури збереження в базі даних ми досягли зменшення розміру колекції на 17 % від початкового розміру. Тепер він становить 5,33 МБ в оригінальному вигляді та 2,56 МБ в стисненому із середнім розміром документа 440 байтів.

Можна також запропонувати модифікований спосіб збереження тексту, але це потребуватиме додаткових обчислювальних ресурсів. Суть цього методу полягає в кодуванні слів і наданні кожному слову в усіх текстах унікального числа, яке буде його замінювати. У такому разі використовується менше символів для збереження. Для реалізації цього методу на вході алгоритму необхідний масив очищених токенів. Першим кроком алгоритму є створення списку унікальних токенів та підрахунок кількості їх застосувань у текстах. Наступним кроком є впорядкування цього списку за кількістю застосувань і надання кожному токену коду; значенням коду може бути порядковий номер унікального токену в списку. На цьому етапі у нас є словник токенів, де кожному токену відповідає його код, причому чим частіше слово трапляється у текстах, тим менше це число-код, який зберігатимемо в базі даних для подальшого використання. Маючи словник токен-код, ми можемо трансформувати масив токенів у масив чисел, а потім перетворити ці числа на рядок, об'єднавши їх пробілом в один великий рядок для збереження в базі даних. На виході алгоритму отримаємо інформацію у вигляді рядка чисел, яку будемо зберігати в базі даних. У подальших ітераціях, додавши нові повідомлення до бази даних, ми будемо лише розширювати наявний словник токен-код, а не змінювати його. Щоб видобути інформацію з бази даних, нам потрібно одночасно зі списком повідомлень видобувати і словник, розділяючи повідомлення на числа у форматі рядка, перетворюючи їх на цілі числа та замінюючи числа на токени.

Звісно, такий підхід не лише спричиняє додаткові ризики, але й потребує додаткових витрат. Основний ризик, пов'язаний з цим алгоритмом, полягає в цілісності даних, оскільки втрата словника призводить до втрати всієї інформації у основній таблиці повідомлень. Крім того, треба пам'ятати про додаткові маніпуляції з інформацією перед збереженням у базу даних та після видобування з неї. Як показано в таблиці нижче, час, витрачений на кодування і декодування тексту, незначний, оскільки опрацювання повного циклу ~12000 повідомлень займає менше ніж 1 секунду. Це робить цей підхід привабливим з погляду економії пам'яті.

Після поглибленого аналізу процесу кодування повідомлень виявлено додаткові джерела шуму. Одне з них – це слова, які можна внести до списку стоп-слів. Стоп-слова є дуже поширеним явищем у текстах і являють собою слова, які не мають важливого значення для розуміння тексту. Оскільки процес доповнення списку стоп-слів є безмежним, завжди будуть слова, які потрібно вносити в нову редакцію списку.

Також виявлено слова, змінені для вираження емоційного забарвлення, наприклад, "дуууже", "велииикий" та інші подібні. Це може створювати додатковий шум у тексті та ускладнювати його аналіз. Вирішення цієї проблеми – складне завдання. Поки що не існує універсального методу для

ефективного усунення цього типу шуму без втрати суттєвої інформації. Отже, цей шум залишиться і впливатиме на кінцевий результат. Окрім цього, варто не забувати про наявність деяких власних імен, які в контексті аналізу текстів можуть змішуватись з загальними поняттями, вливаючи на кінцевий результат аналізу. Боротьба з таким негативним впливом потребує недоцільно великих ресурсів, наприклад, робочих годин людини, яка власноруч складає список із сучасних власних імен. Важливо постійно вдосконалювати методи аналізу тексту та пошуку ефективних способів боротьби з шумом для забезпечення максимально точного та надійного аналізу.

Аналізуючи ефективність використання пам'яті, важливо звернути увагу на один ключовий аспект. Як ми вже зазначали, цей алгоритм використовує дві колекції: одну для словника та іншу для повідомлень, де перша таблиця займає 1,15 МБ, а друга – 1,61 КБ, загалом – близько 2,76 МБ. Порівнявши з попереднім методом, який полягав у збереженні звичайних токенів у один рядок, можна зробити висновок, що старий спосіб є ефективнішим, оскільки потребував менше пам'яті, а саме 2,56 МБ. Однак, якщо розглянути середньостатистичний розмір одного документа з колекції повідомлень, стає зрозуміло, що його об'єм зменшився удвічі, з 440 байтів до 249 байтів.

У поточній конфігурації може здатися, що словник є зайвим тягарем. Однак потрібно врахувати, що обсяг словника та повідомлень зростатиме нелінійно, як показано на рис. 4. Приріст обсягу словника виявляється меншим за приріст обсягу повідомлень зі збільшенням кількості вхідних повідомлень. Одночасно, якщо обсяг словника теоретично може мати певну межу, обсяг повідомлень є безмежним. Такий підхід оптимальний для великого обсягу текстів, які використовують один словник. Проте якщо необхідно кодувати невеликий обсяг текстів, цей метод може стати контрпродуктивним, адже спочатку обсяг словника є більшим за обсяг повідомлень.

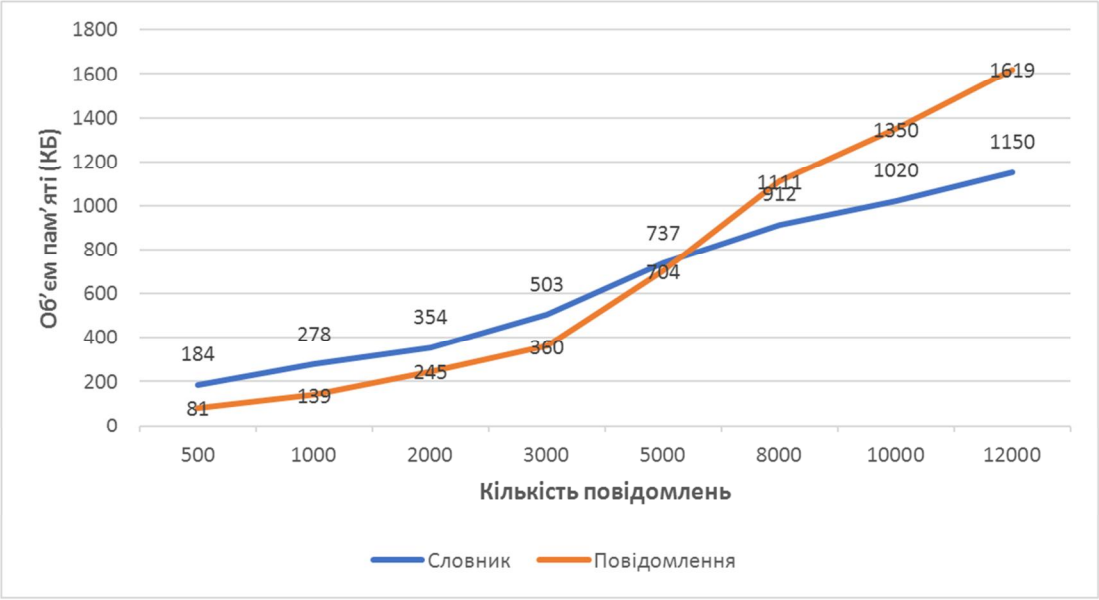

Рис. 4. Залежність обсягу словника та повідомлень в базі даних від кількості вхідних повідомлень з використанням стратегії кодування тексту

Звісно, таке перетворення потребує дзеркального процесу декодування, щоб видобути увесь словник кодів та необхідну частину закодованих повідомлень з бази даних. Такий процес навіть менш ресурсомісткий порівняно з кодуванням – на 600 %, адже він не потребує операції пошуку обчислюваного токена у великому масиві наявних токен-кодів.

Враховуючи потребу опрацювання тексту в режимі реального часу з невеликими інтервалами, що тривають кілька годин, важливим аспектом є швидкість генерації даних. Аналізуючи повідомлення, згруповані за днями, можемо простежити певну динаміку: в дні, коли відбуваються важливі події, такі як масштабні ракетні обстріли, кількість повідомлень різко зростає на всіх каналах. На рис. 5 показано залежність загальної кількості повідомлень (вісь Y) від дня публікації (вісь X), де чітко видно аномальну турбулентність 2 та 23 січня, а упродовж решти днів кількість повідомлень залишається близькою до середнього показника.

Рис. 5. Діаграма швидкості генерування даних за днями

Якщо згрупувати повідомлення за годиною публікації, виявляється, що з четвертої години ранку загальна кількість повідомлень зростає на восьми вибраних джерелах інформації до пікових значень протягом трьох годин від 5^{00} до 9^{00} та після незначного спаду утримується на цьому рівні до дев'ятої години вечора. На рис. 6 показано залежність кількості повідомлень (вісь Y) від години публікування (вісь X).

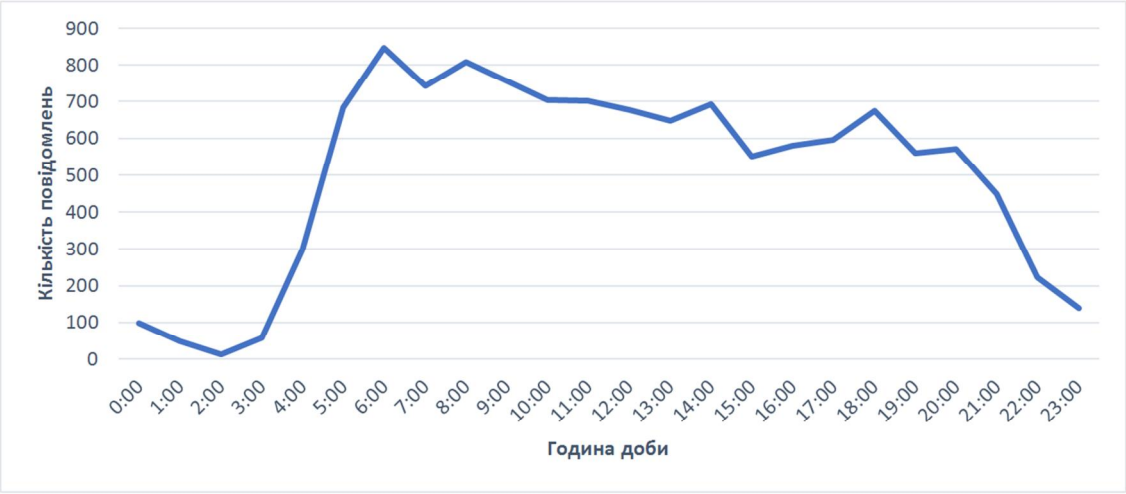

Рис. 6. Діаграма середньостатистичної швидкості генерування даних за годинами

Ще одним важливим аспектом, якого варто розглянути, є використання інформаційних джерел іноземною мовою, зокрема, багато з них є англомовними. Отже, потрібно або створити паралельну систему, або здійснювати переклад. У разі використання безкоштовної моделі Google-перекладача проблем з перекладом англійських текстів на українську практично не виникає, за винятком окремих випадків, таких як переклад власних імен. Проте цей процес займає дуже багато часу (див. таблицю, що містить переклад дев'яти повідомлень), що створює зворотний тиск на систему попереднього опрацювання даних, коли тисячі завантажених повідомлень стоятимуть у черзі на переклад протягом хвилин.

Часові витрати на кожну операцію попереднього опрацювання

Процедура	Час роботи процесора	Загальний час
Завантаження повідомлень	–	1,20 хв
Збереження сирих даних у БД	531 мс	889 мс
Видалення Emoji	**1,92 с**	2,34 с
Видалення чисел	31 мс	81 мс
Токенізація	2,52 с	2,84 с
Видалення стоп-слів	6,89 с	8,23 с
Збереження токенів у БД	203 мс	410 мс
Переклад дев'яти повідомлень	469 мс	3,32 с
Кодування токенів	858 мс	1,09 с
Декодування токенів	128 мс	227 мс

У ході експериментів виконано статистичні розрахунки роботи кожного з етапів попереднього опрацювання даних для того, щоб виявити вузькі місця обчислень. Всі дані наведено в таблиці, де враховано час роботи процесора та загальний час роботи, в який входить очікування комп'ютером процесів введення/виведення.

Висновки

Експериментально оцінено вплив попереднього опрацювання даних на обсяг збережених даних та час їх опрацювання. Запропоновано ефективну стратегію оптимального зберігання текстових даних для наступних етапів опрацювання із використанням алгоритмів дослідження природної мови. Серед запропонованих методів виділено метод збереження очищених токенів у форматі рядка та метод збереження кодів слів у форматі рядка разом зі словником слово-код. Найменш ефективним методом є збереження очищених токенів у форматі масиву рядків.

Оцінювання продуктивності та обсягу даних після різних етапів опрацювання свідчить про ефективність запропонованих підходів. Досліджуючи різні методи зберігання, ми встановили, що зберігання масивів токенів у вигляді рядків, об'єднаних одним символом, зменшує розмір даних на 17 %. Крім того, новий метод кодування, хоча і пов'язаний із ризиками для цілісності, забезпечує значну економію пам'яті, особливо для великих обсягів тексту з одним словником.

Міркування щодо ефективності висвітлили компроміс між використанням пам'яті та витратами на кодування, причому оптимальний підхід залежить від обсягу тексту. У дослідженні також проаналізовано швидкість генерування даних протягом днів і годин, визначено періоди пікової активності, що має вирішальне значення для опрацювання в реальному часі.

Зроблені висновки сприяють підвищенню ефективності зберігання та опрацювання текстових даних, що має вирішальне значення для розвитку та масштабування додатків для опрацювання природної мови в різних контекстах.

Список літератури

1. Talakh, M. V. (2019). PART 7. Using text mining for the analysis of social networks. In Ushenko, Y., Ostapov, S. & Golub, S., (Eds.), *Information technologies Part 1. Application in computer vision, recognition and intelligent monitoring systems Yuriy Ushenko, Serhiy Ostapov, Serhiy Golub* (pp. 157–173). LAP LAMBERT Academic Publishing.

2. Talakh, M. V., Holub, S. & Lazarenko Y. (n. d.). Intelligent monitoring of software test automation of Web sites. *International Scientific and Practical Conference "Intellectual Systems and Information Technologies"*, 46–51.

3. Telegram (n. d.). *Telegram APIs.* Retrieved February 1, 2024, from https://core.telegram.org/api

4. Chai, C. (2023). Comparison of text preprocessing methods. *Natural Language Engineering, 29(3),* 509–553. https://doi.org/10.1017/S1351324922000213

5. R-Project (n. d.). *Unicode: Emoji, accents, and international text.* Retrieved February 10, 2024, from https://cran.r-project.org/web/packages/utf8/vignettes/utf8.html

6. Mohammad, F. (2018). Is preprocessing of text really worth your time for online comment classification? *eprint arXiv, 1806(029908)*, 1–5. https://doi.org/10.48550/arXiv.1806.02908

7. Camacho-Collados, J. (2018). On the Role of Text Preprocessing in Neural Network Architectures: An Evaluation Study on Text Categorization and Sentiment Analysis. *eprint arXiv, 1707(01780)*, 1–4. https://doi.org/10.48550/arXiv.1707.01780

8. Kumar, K., & Harish, B. S. (2017). Classification of Short Text Using Various Preprocessing Techniques: An Empirical Evaluation. *Proceedings of the 5th ICACNI 2017, Vol. 3 (10.1007/978-981-10-8633-5_3)*, 19–24. http://dx.doi.org/10.1007/978-981-10-8633-5_3

9. Mediakov, O. (2024). Information Technology for Generating Lyrics for Song Extensions Based on Transformers. In Mediakov, O., Vysotska, V., Uhryn, D., Ushenko, Y. & Hu, C., (Eds.), *International Journal of Modern Education and Computer Science (IJMECS), 16(1)*, 23–36.

10. Lytvyn, V. (2018). Analysis of statistical methods for stable combinations determination of keywords identification. In Lytvyn, V., Vysotska, V., Uhryn, D., Hrendus, M. & Naum, O. (Eds.), *Information technology: Eastern-European Journal of Enterprise Technologies, 2/2(92)*, 23–37.

11. Lytvyn, V. (2017). Development of a method for determining the keywords in the slavic language texts based on the technology of web mining. In Lytvyn, V., Vysotska, V., Pukach, P., Brodyak, O. & Ugryn D. (Eds.), *Information technology. Industry control systems: Eastern-European Journal of Enterprise Technologies, 2/2(86)*, 14–23.

12. JupyterLab (n. d.). *JupyterLab Documentation.* Retrieved February 1, 2024, from https://jupyterlab.readthedocs.io/en/stable/index.html

13. Python (n. d.). *Our Documentation.* Retrieved February 1, 2024, from https://www.python.org/doc/

14. Mongo (2024). *Mongo: The developer data platform*. Retrieved February 1, 2024, from https://www.mongodb.com/

15. Parker, Z., Poe, S. & Vrbsky, S. (2013). Comparing nosql mongodb to an sql db. *Proceedings of the 51st ACM Southeast Conference.* https://dl.acm.org/doi/10.1145/2498328.2500047

16. Li, Y. & Manoharan, S. (2013). A performance comparison of SQL and NoSQL databases. *IEEE Pacific RIM Conference on Communications, Computers, and Signal Processing – Proceedings*, 15–19. https://doi.org/10.1109/PACRIM.2013.6625441

17. Prokipchuk, O. (2023). Intelligent Analysis of Ukrainian-language Tweets for Public Opinion Research based on NLP Methods and Machine Learning Technology. In Prokipchuk, O., Vysotska, V., Pukach, P., Lytvyn, V., Uhryn, D., Ushenko, Y. & Hu, Z. (Eds.), *International Journal of Modern Education and Computer Science(IJMECS), 15(3)*, 70–93.

18. Telethon (2024). *Telethon's Documentation.* Retrieved February 1, 2024, from https://docs.telethon.dev/en/stable/index.html

References

1. Talakh, M. V. (2019). Part 7. Using text mining for the analysis of social networks. In Ushenko, Y., Ostapov, S. & Golub, S., (Eds.), *Information technologies Part 1. Application in computer vision, recognition and intelligent monitoring systems Yuriy Ushenko, Serhiy Ostapov, Serhiy Golub* (pp. 157–173). LAP LAMBERT Academic Publishing.

2. Talakh, M. V., Holub, S. & Lazarenko Y. (n.d.). Intelligent monitoring of software test automation of Web sites. *International Scientific and Practical Conference "Intellectual Systems and Information Technologies"*, 46–51.

3. Telegram (n. d.). *Telegram APIs.* Retrieved February 1, 2024, from https://core.telegram.org/api

4. Chai, C. (2023). Comparison of text preprocessing methods. *Natural Language Engineering, 29(3),* 509–553. https://doi.org/10.1017/S1351324922000213

5. R-Project (n. d.). *Unicode: Emoji, accents, and international text.* Retrieved February 10, 2024, from https://cran.r-project.org/web/packages/utf8/vignettes/utf8.html

6. Mohammad, F. (2018). Is preprocessing of text really worth your time for online comment classification? *eprint arXiv, 1806(029908)*, 1–5. https://doi.org/10.48550/arXiv.1806.02908

7. Camacho-Collados, J. (2018). On the Role of Text Preprocessing in Neural Network Architectures: An Evaluation Study on Text Categorization and Sentiment Analysis. *eprint arXiv, 1707(01780)*, 1–4. https://doi.org/10.48550/arXiv.1707.01780

8. Kumar, K., & Harish, B.S. (2017). Classification of Short Text Using Various Preprocessing Techniques: An Empirical Evaluation. *Proceedings of the 5th ICACNI 2017, Vol. 3 (10.1007/978-981-10-8633-5_3)*, 19–24. http://dx.doi.org/10.1007/978-981-10-8633-5_3

9. Mediakov, O. (2024). Information Technology for Generating Lyrics for Song Extensions Based on Transformers. In Mediakov, O., Vysotska, V., Uhryn, D., Ushenko, Y. & Hu, C., (Eds.), *International Journal of Modern Education and Computer Science (IJMECS), 16(1)*, 23–36.

10. Lytvyn, V. (2018). Analysis of statistical methods for stable combinations determination of keywords identification. In Lytvyn, V., Vysotska, V., Uhryn, D., Hrendus, M. & Naum, O., (Eds.), *Information technology: Eastern-European Journal of Enterprise Technologies, 2/2(92)*, 23–37.

11. Lytvyn, V. (2017). Development of a method for determining the keywords in the slavic language texts based on the technology of web mining. In Lytvyn, V., Vysotska, V., Pukach, P., Brodyak, O. & Ugryn D. (Eds.), *Information technology. Industry control systems: Eastern-European Journal of Enterprise Technologies, 2/2(86)*, 14–23.

12. JupyterLab (n. d.). *JupyterLab Documentation*. Retrieved February 1, 2024, from https://jupyterlab.readthedocs.io/en/stable/index.html

13. Python (n.d.). *Our Documentation.* Retrieved February 1, 2024, from https://www.python.org/doc/

14. Mongo (2024). *Mongo: The developer data platform*. Retrieved February 1, 2024, from https://www.mongodb.com/

15. Parker, Z., Poe, S. & Vrbsky, S. (2013). Comparing nosql mongodb to an sql db. *Proceedings of the 51st ACM Southeast Conference.* https://dl.acm.org/doi/10.1145/2498328.2500047

16. Li, Y. & Manoharan, S. (2013). A performance comparison of SQL and NoSQL databases. *IEEE Pacific RIM Conference on Communications, Computers, and Signal Processing – Proceedings*, 15–19. https://doi.org/10.1109/PACRIM.2013.6625441

17. Prokipchuk, O. (2023). Intelligent Analysis of Ukrainian-language Tweets for Public Opinion Research based on NLP Methods and Machine Learning Technology. In Prokipchuk, O., Vysotska, V., Pukach, P., Lytvyn, V., Uhryn, D., Ushenko, Y. & Hu, Z., (Eds.), *International Journal of Modern Education and Computer Science(IJMECS), 15(3)*, 70–93.

18. Telethon (2024). *Telethon`s Documentation.* Retrieved February 1, 2024, from https://docs.telethon.dev/en/stable/index.html

IMPROVEMENT OF TEXT DATA STORAGE METHODS

Vasyl Lytvyn[1], Artem Kalancha[2], Dmytro Uhryn[3], Maria Talakh[4]

[1] Lviv Polytechnic National University,
Department of Information Systems and Sources, Lviv, Ukraine
[2–4] Yuri Fedkovich Chernivtsi National University
Department of Computer Sciences, Chernivtsi, Ukr
[1] E-mail: vasyl.v.lytvyn@lpnu.ua, ORCID: 0000-0002-9676-0180
[2] E-mail: kalancha.artem@chnu.edu.ua, ORCID: 0009-0004-1451-7470
[3] E-mail: d.ugryn@chnu.edu.ua, ORCID: 0000-0003-4858-4511
[4] E-mail: m.talah@chnu.edu.ua, ORCID: 0000-0002-5067-6848

In this research, an analysis of the qualitative characteristics of messages in the Telegram messenger was carried out, which are used as raw data for further analysis of textual content. A thorough review of the parameters of these messages, such as their format, size, presence of noise, and speed. The main goal of the article is to model the optimal approach to saving a large amount of data before the important stage of text analysis. During the research, a detailed analysis of literary sources devoted to this topic was carried out. The article examines the main advantages and disadvantages of existing data preprocessing algorithms, as well as problems related to data purity and their impact on potential research results. As part of the software experiments, the impact of data preprocessing on the size of the saved data for further use, as well as on the speed of input data generation, was evaluated. Among the proposed methods, the method of saving cleared tokens in string format and the method of saving word codes in string format together with the word-code dictionary were highlighted. This is aimed at ensuring the effective distribution of tasks of the text analysis system during the period of the day.

Key words: text analysis; text preprocessing; database; encoding.