КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
ІМЕНІ ТАРАСА ШЕВЧЕНКА
Факультет комп'ютерних наук та кібернетики
Кафедра інтелектуальних програмних систем

Кваліфікаційна робота
на здобуття ступеню бакалавра
за спеціальністю 121 Інженерія програмного забезпечення

на тему:

АВТОМАТИЧНЕ РЕФЕРУВАННЯ ТЕКСТУ НА ОСНОВІ ЗАПИТУ

Виконала студентка 4-го курсу
Юлія САМАРЦЕВА

(підпис)

Науковий керівник:
Доцент, кандидат фізико-математичних наук
Ярослав ЛІНДЕР

(підпис)

Засвідчую, що в цій курсовій роботі немає запозичень з праць інших авторів без відповідних посилань
Студент

(підпис)

Роботу розглянуто й допущено до захисту на засіданні кафедри інтелектуальних програмних систем
«25» травня 2022 р.,
протокол № 10

Завідувач кафедри
О. І. Провотар

(підпис)

Київ – 2022

РЕФЕРАТ

Обсяг роботи: 40 сторінок, 6 ілюстрацій, 19 джерел посилань.

АВТОМАТИЧНЕ РЕФЕРУВАННЯ ТЕКСТІВ, ЕКСТРАКТНЕ РЕФЕРУВАННЯ, МЕТОДИ РЕФЕРУВАННЯ, ОБРОБКА ПРИРОДНОЇ МОВИ, ВЕБ-РОЗШИРЕННЯ.

Об’єктом розробки є процес екстракції фрагмента тексту, який найбільше відповідає даного запиту. Предметом роботи є програмний засіб у вигляді веб-розширення для пошуку найбільш релевантного тексту до даного запиту.

Метою роботи є аналіз методів реферування текстових даних, створення алгоритму для реферування тексту за запитом та створення програмного засобу, що буде виконувати аналіз тексту. Готовий програмний продукт повинен мати такі характеристики:

- Використання на різних веб-сайтах, що містять текстову інформацію
- Можливість вводити запит
- Можливість користувацьких налаштувань, таких як вибір розміру фрагмента, відключення пошуку за запитом, тощо.

Методи розроблення: метод екстрактного реферування текстів на основі запитів, його імплементація та використання у проєктуванні розширення для браузера, об'єктноорієнтоване програмування, принципи архітектури проєктування розширень браузера.

Інструменти розроблення: для розроблення додатка - Javascript, CSS, HTML, Google Cloud Functions; для розробки алгоритму для реферування - безкоштовне, вільно поширюване інтегроване середовище PyCharm 2020.3.2, мова Python.

Результати роботи: У результаті роботи була досліджена проблема реферування тексту, проаналізовано різні види алгоритмів для реферування тексту, розроблено алгоритм для екстрактного реферування на основі запиту та розроблено програмний продукт у вигляді веб-розширення, який дозволяє наочно побачити результат роботи алгоритму.

Програмний продукт може застосовуватися на будь-яких веб-сайтах для пошуку найважливішої інформації за запитом. Алгоритм також може бути застосований для аналізу великого обсягу тексту для створення реферату на основі запиту.

Зміст

РЕФЕРАТ.....1
СКОРОЧЕННЯ ТА УМОВНІ ПОЗНАЧЕННЯ.....5
ВСТУП.....7
РОЗДІЛ 1 ЗАВДАННЯ АВТОМАТИЧНОГО РЕФЕРУВАННЯ ТЕКСТУ 10
1.1 Загальні відомості про реферування тексту 10
1.2 Систематизація текстових реферувальних систем 12
1.3 Екстрактні моделі для реферування тексту 15
1.3.1 Латентно семантичний аналіз 15
1.3.2 TextRank 18
1.3.3 Методи, які використовують нейронні мережі 20
1.4 Rouge 21
1.4.1 Rouge-N: Статистика спільних випадків N-грами 21
1.4.2 Rouge-L: Найдовша загальна підпослідовність 22
1.4.3 Rouge-W: Зважена найдовша загальна підпослідовність.....23
1.4.4 Rouge-S: Статистика поява Skip-Bigram.....23
1.4.5 Недоліки використання Rouge оцінок.....24
РОЗДІЛ 2 АВТОМАТИЧНЕ РЕФЕРУВАННЯ ТЕКСТУ НА ОСНОВІ ЗАПИТУ 24
2.1 Постановка задачі 24
2.2 Загальний процес реферування на основі запиту 26
2.3 Детальний опис запропонованого алгоритму 27
2.3.1 Пошук релевантних речень 28
2.3.2 Генерація текстового реферату.....29
РОЗДІЛ 3 ВИКОРИСТАННЯ РОЗРОБЛЕНОГО АЛГОРИТМУ У ВЕБ-РОЗШИРЕННІ.....30
3.1 Аналіз відомого програмного забезпечення 30

3.2 Деталі технології розробки веб-розширення 31
3.3 Опис інтерфейсу 33
РОЗДІЛ 4 ОЦІНКА МОДЕЛІ 35
4.1 Час роботи алгоритму 35
4.2 Тестування 36
ВИСНОВКИ 37
ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ 38

СКОРОЧЕННЯ ТА УМОВНІ ПОЗНАЧЕННЯ

UX – User Experience, який досвід або враження отримує користувач від роботи з інтерфейсом;

UI – User Interface, як виглядає інтерфейс і які фізичні характеристики набуває;

API – Application Programming Interface, Прикладний програмний інтерфейс;

IDE – Integrated Development Environment, Інтегроване Середовище Розробки;

CORS – Cross-Origin Resource Sharing, Спільне використання ресурсів з різних джерел;

HTTP – HyperText Transfer Protocol, протокол передачі гіпертекстових документів;

LSA – Latent Semantic Analysis, Латентно семантичний аналіз;

SVD – Singular Value Decomposition, Сингулярний розклад матриці;

TF-IDF – Term Frequency - Inverse Document Frequency, Частота входження терму в текст - Обернена частота входження. Статистичний показник, що використовується для оцінки важливості слів у контексті документа;

NLP – Natural Language Processing, Обробка природної мови;

GloVe – Global Vectors for Word Representation, Глобальні вектори для репрезентації слів;

CNN – Convolutional Neural Network, Згорткова нейронна мережа;

RNN – Recurrent Neural Network, Рекурентна нейронна мережа;

IR – Information Retrieval, Інформаційний пошук;

ROUGE – Recall-Oriented Understudy for Gisting Evaluation, Набір показників, що використовується для оцінки автоматичного реферування та машинного перекладу в обробці природної мови;

LCS – Longest Common Subsequence, Найдовша Загальна Підпослідовність;

WLCS – Weighted Longest Common Subsequence, Зважена Найдовша Загальна Підпослідовність.

ВСТУП

Актуальність роботи та підстави її виконання. Текстові дані є досить розповсюдженими у повсякденному житті. Є також багато категорій текстової інформації та різних видів ресурсів, такі як новини, статті, книги, тощо. Однак, більшість тексту в інтернеті є досить великою за об’ємом та може містити багато нерелевантної для користувача інформації. В результаті, для пошуку необхідної інформації користувачі повинні перечитувати весь текст для того, щоб знайти відповідь на необхідне запитання. Реферування тексту може допомогти ефективно скоротити текст до менших об’ємів зберігаючи зміст та знайти необхідну частину, де буде міститись корисна інформація. Корисність буде визначатися потребами користувача, а саме тим, наскільки фрагмент зберігає вміст тексту та відноситься до даного запиту. Реферування тексту вручну є трудомістким завданням, тому автоматизація цього процесу є актуальним завданням на сьогодні.

Традиційні методи для реферування за допомогою екстракції генерують лише один фіксований фрагмент з даного вхідного тексту. Таким чином, вони знаходять всю частину тексту, яка містить необхідну інформацію, але це призводить до того, що фрагмент може виявитись досить великим. З іншої сторони, якщо зменшувати бажаний розмір отриманого фрагмента, можна втратити потенціально важливу інформацію для даних потреб користувача.

Для того, щоб зробити пошук інформації у тексті більш ефективним та зручним і позбавитись від згаданих проблем, можна створити метод, який буде крім тексту також приймати на вхід запит користувача. В результаті, в залежності від запиту, на вихід може бути отримано багато різних фрагментів тексту. Це відрізняє даний метод від традиційного методу, коли на вихід може бути отриманий лише один фрагмент тексту.

Мета роботи. Аналіз наявних підходів для автоматичного реферування текстів, розробка алгоритму для реферування на основі запиту та створення програмного засобу, що буде виконувати аналіз тексту.

Завдання роботи. Для досягнення мети було поставлено такі завдання:

- Аналіз та систематизація методів реферування документів
- Дизайн екстрактного алгоритму для реферування на основі запиту
- Створення дизайну та архітектури програмної системи, що буде використовувати розроблений алгоритм
- Аналіз отриманих результатів

Об'єкт, методи й засоби розроблення. Об’єктом розроблення програмного веб-розширення є процес знаходження фрагмента тексту, що найбільш точно відповідає даному запиту.

Розробці програмного забезпечення передує створення моделі для реферування тексту на основі запиту. Основу для цього склав аналіз наявних підходів для реферування текстів та їх систематизація і знаходження такого алгоритму, що буде ефективно розв'язувати дану проблему.

Можливі сфери застосування. Алгоритм для реферування тексту на основі запиту може бути використаний для пошуку фрагмента тексту, що відповідає даному запиту та містить ту саму інформацію, що й у початковому тексті. Програмний продукт у вигляді веб-розширення може застосовуватися для реферування текстів з веб-сайтів.

Робота починається з загальної інформації про реферування документів (Розділ 1), включаючи фактори за якими можна класифікувати реферування документів та розглянуті різні наявні види екстрактного реферування текстів. Розділ 2 розглядає запропонований алгоритм для реферування за допомогою

запиту, а Розділ 3 розглядає розроблене веб-розширення використовуючи відповідні техніки розглянуті в попередніх розділах. Розділ 4 містить аналіз отриманих результатів та висновки щодо виконаної роботи.

Робота складається зі вступу, 4 розділів, висновків та списку літератури.

РОЗДІЛ 1 ЗАВДАННЯ АВТОМАТИЧНОГО РЕФЕРУВАННЯ ТЕКСТУ

1.1 Загальні відомості про реферування тексту

Зростання кількості інформації в інтернеті призвело до величезної кількості змісту у вигляді веб-сторінок, що може бути у різній формі: новини, зображення, бази даних, відео, тощо. Отже, напрям NLP почав використовуватися для вилучення релевантної інформації з інформаційних ресурсів. Реферування тексту є підрозділом NLP, що допомагає користувачеві знайти необхідну інформацію з великого документа. Реферування також може бути застосоване для вилучення важливої інформації з кількох документів. Це дозволяє користувачу скоротити час читання кількох джерел, що прискорює процес пошуку інформації. Реферат документа дозволяє надати ключові моменти у стислому форматі [12].

Процес реферування документа складається з кількох кроків: ідентифікація джерела, інтерпретація змісту та формування реферату. Загальний процес реферування тексту показаний на рисунку 1.1.

Процес стемінгу (Stemming) дозволяє зводити слова до певної основи шляхом відкидання спільної частини, наприклад закінчення або суфіксу. Лематизація (Lemmatization) переводить слова до леми (її нормальної форми) з використанням словникового запасу та морфологічного аналізу слів, зазвичай спрямованих на видалення лише закінчень і повернення основи.

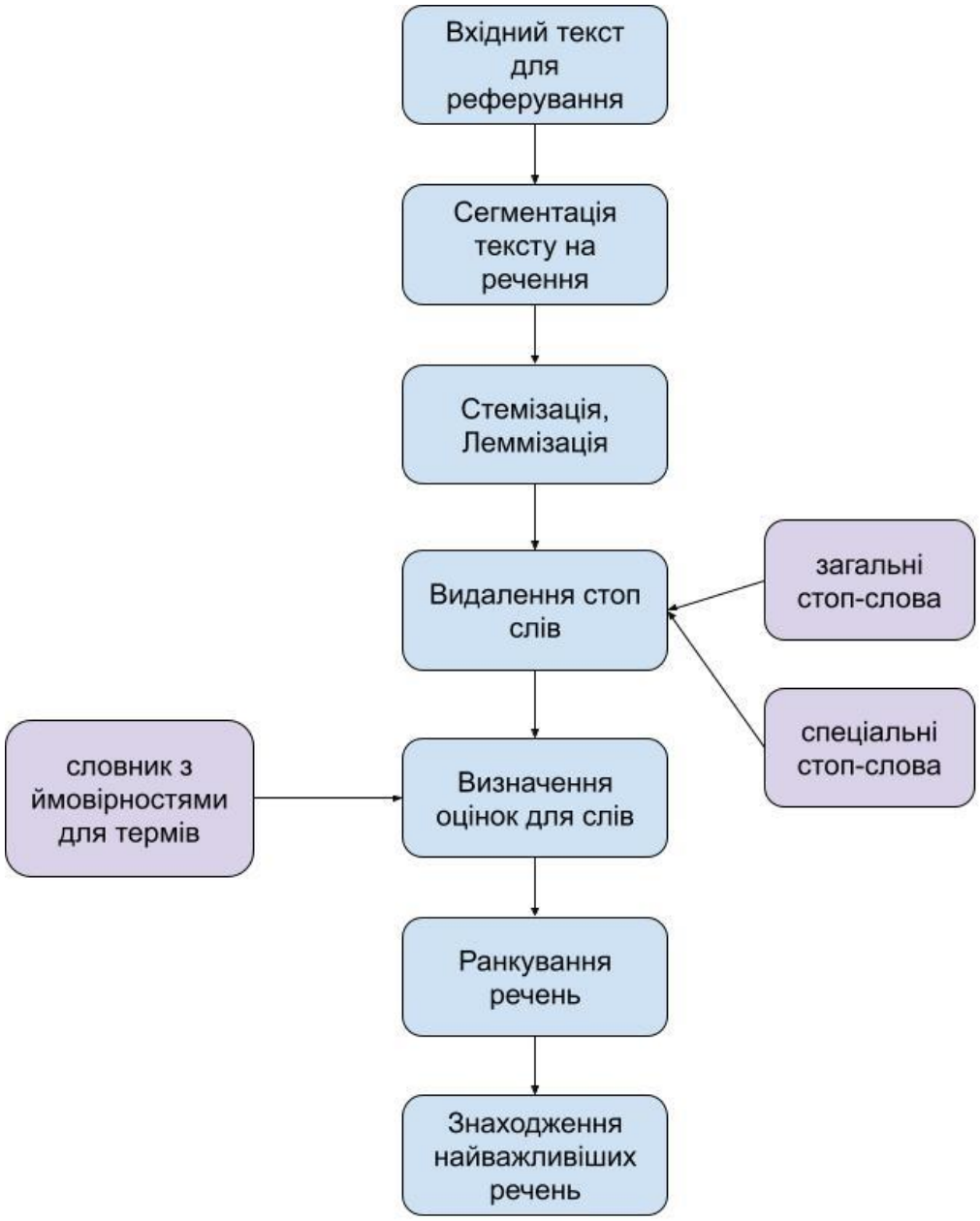


Рис. 1.1 Процес реферування тексту

Наприклад, після лематизації та стемінгу “am, are, is” можуть бути всі переведені у слово “be” або “car, cars, car's, cars'” у слово car. Так, речення “the girl’s cars are colourful” буде переведене у речення “the girl car be colour”.

Більшість робіт з екстрактного реферування використовують статистичні моделі. Наприклад, такі алгоритми знаходять ранжування речень

відповідно до відношення речень один до одного використовуючи попередньо визначені формули. Наприклад, для алгоритму Луна використовується сума частот значущих слів, для латентно-семантичного аналізу кореляція з основними поняттями або темами і схожість між двома реченнями для TextRank алгоритмів.

Пізніше з розвитком машинного навчання, задача реферування тексту стала використовувати моделі машинного навчання, що визначають чи включати речення у реферат. Алгоритми машинного навчання визначають необхідну репрезентацію тексту та прогнозування того, чи включати речення у реферат.

Дослідження у сфері NLP зазвичай фокусувалося на аналізі якісних текстів і створення великих за обсягом рефератів, а проблема реферування тексту використовуючи запит не мала широкого розповсюдження у NLP спільнотах, і була більш поширена у IR спільноті.

1.2 Систематизація текстових реферувальних систем

1) **Однодокументні та багатодокументні**: Є декілька видів реферувальних систем. По-перше, реферувальні системи поділяються на однодокументні та багатодокументні. Багатодокументні алгоритми, крім тієї статті, яку потрібно реферувати, потребують на вхід також інші документи з цієї само категорії або ресурсу.

2) **Напрям екстракції та напрям абстракції**: Також, реферати будуються за двома напрямками – екстракція та абстракція. З аналізу даних напрямків можна побачити, що метод абстракції є набагато складнішим, тому що необхідно, щоб дана модель також генерувала зв’язні фрази на основі вивченого тексту.

Екстракція – обираються релевантні фрагменти з даного вхідного документа та об’єднуються для того, щоб утворити підсумок.

Абстракція – створюється фрагмент тексту, що зберігає оригінальний задум. Це відбувається аналогічно тому, як люди будують підсумок тексту.

Крім того, є такі методи, коли використовується скомбінована модель, де спочатку використовуються методи екстракції, а після цього методи абстракції. Наприклад, такий підхід використовується для Pointer-Generator Networks шляхом перемикання ймовірностей [2] або ж просто послідовне використання спочатку екстрактної моделі, а потім абстрактної моделі до обраних речень [3].

3) **Загальний аналіз та на основі запитів**: Наступною категоризацією є методи, що базуються на загальному аналізі тексту та на основі запитів. При використанні загального аналізу, фрагмент тексту в результаті створений використовуючи всю інформацію знайдену в документах, тоді як при використанні запитів, фрагмент тексту в результаті також повинен бути зосереджений на інформації з вхідного документа яка відповідає запиту користувача.

Підхід пошуку речень, які містять такі самі слова, як і запит, може виявитися досить неточними. Оскільки зазвичай існує багато способів вираження даного поняття, терміни в запиті користувача можуть не збігатися з термінами документа. Крім того, більшість слів мають по декілька значень, тому терміни в запиті користувача можуть не відповідати буквально термінам в документі. Кращий підхід дозволить обирати речення на основі значення документа.

4) **Навчання з учителем та без вчителя**: Методи реферування також можна класифікувати на основі необхідного набору даних. Навчання з учителем потребує набору даних, що складається з пар текст-підсумок, інакше їх називають методами без вчителя. У багатьох випадках варіанти з вчителем працюють краще ніж без вчителя з точки зору точності, але вони вимагають етапу навчання та наявності відповідного набору даних для нього.

5) **“Переносне навчання”**: Іншим підходом є використання “Переносного навчання”, коли використовується вже вивчена модель для створення нової моделі реферування [4]. Однак даний підхід зазвичай використовується для узагальнення даних для яких відома певна область і є підходом для багатодокументних алгоритмів.

6) **Веб-документне та не веб-документне**: Більшість алгоритмів для реферування тексту є не веб-документними. Однак, веб-документи зазвичай містять текстову інформацію, яка є більш різноманітною та може містити додаткову інформацію. [13]

Отже, підходи для реферувальних систем можна поділити на такі категорії:

- Однодокументне та багатодокументне реферування
- Реферування за допомогою екстракції, абстракції або змішаний підхід
- Реферування базовані на загальному аналізі тексту та на основі запитів
- Навчання з учителем (контрольоване) та без вчителя
- Використання “Переносного навчання”

- Веб-документне та не веб-документне

Для поставленої задачі програмного додатка, необхідно використовувати екстрактне реферування тексту. Тому, на далі у роботі будуть розглянуті найпопулярніші алгоритми для екстрактного реферування тексту.

1.3 Екстрактні моделі для реферування тексту

1.3.1 Латентно семантичний аналіз

Латентно семантичний аналіз – це алгебраїко-статистичний метод, який працює на основі прихованих семантичних структур слів і речень. Цей підхід не потребує жодної підготовки чи зовнішніх знань, тобто є підходом для навчання без вчителя. LSA аналізує контекст вхідного документа, наприклад, які слова використовуються разом і які загальні слова можна побачити в різних реченнях. Велика кількість спільних слів серед речень свідчить про те, що речення є семантично пов’язаними. [8]

В основі латентно семантичного аналізу лежить поняття сингулярного розкладу матриці (SVD). Нехай A – це будь-яка m на n матриця. Тоді, сингулярним розкладом матриці буде називатися розклад на множники у вигляді:

$$A = U\Sigma V^T = u_1\sigma_1 v_1^T + \cdots + u_r\sigma_r v_r^T$$

де U – дійсна або комплексно унітарна матриця розміру m на m;

Σ – прямокутна діагональна матриця з невід’ємними числами на діагоналі;

V - дійсна або комплексно унітарна матриця розміру n на n.

Значення u з SVD називаються ліво-сингулярними векторами, а v називаються право-сингулярними векторами. Значення σ називаються сингулярними значеннями.

Для того, щоб скористатися методом SVD, необхідно представити документ у вигляді матриці, де стовпці є реченнями, а рядки — словами/фразами. Значення в матриці використовуються для представлення важливості слів у реченнях. Однак, для зменшення складності алгоритму, треба зменшити кількість вхідних слів. Це можна зробити прибравши стоп-слова, використовуючи лише корені слів, тощо. Є декілька різних методів для того, щоб визначити значення в матриці:

- Частота слова в реченні: клітинка матриці заповнюється частотою слова в реченні
- Бінарна репрезентація: клітинка матриці заповнюється з 0 або 1 в залежності від того чи існує дане слово в реченні
- TF-IDF: клітинка матриці заповнюється TF-IDF значенням слова
- Модифікований TF-IDF: клітинка матриці заповнюється спочатку значеннями TF-IDF, але потім ті клітинки, що мають значення менше або дорівнює середньому в рядку змінюються на 0.

Визначення TF-IDF

Для того, щоб обрахувати TF-IDF необхідно окремо обрахувати величини TF та IDF і перемножити їх. Для обрахування частоти входження терму у текст (TF), можна скористатися формулою:

$$TF = \frac{n_i}{\sum_k n_k}$$

де n_i – число входжень слова в документ.

Для обрахування оберненої частоти входження (IDF), можна скористатися формулою:

$$IDF = \log \frac{|D|}{|d_i \supset t_i|}$$

де |D| – кількість документів колекції;

$|d_i \supset t_i|$ – кількість документів, в яких зустрічається слово t_i (коли $n_i \neq 0$).

Вибір речень

Після розкладення матриці методом SVD необхідно обрати найважливіші з них. Для цього для кожного речення обраховується значення:

$$s_k = \sqrt{\sum_{i=1}^{n} v_{k,i}^2 \cdot \sigma_i^2}$$

В результаті до кінцевого фрагмента тексту обираються речення з найбільшими значеннями у векторі s. Більші значення будуть набувати ті речення, що містять найбільше інформації за однією з тем документа.

Косинус подібності

Косинус подібності — це міра, яку можна використовувати для порівняння документів або для визначення рейтингу документів щодо заданого вектора слів запиту.

$$\cos(X, Y) = \frac{\sum x_i \cdot y_i}{\sqrt{\sum (x_i)^2 \cdot \sqrt{(y_i)^2}}}$$

де X та Y є представленнями на основі моделі векторного простору.

1.3.2 TextRank

TextRank є алгоритмом на основі екстракції, який представляє тексти у вигляді графів для визначення речень і ключових слів для вилучення та використовує PageRank алгоритми для їх ранжування. [11]

TextRank алгоритм зазвичай складається з таких кроків:

1. Визначення текстових одиниць, які найкраще визначають поставлене завдання та додавання їх як вершин у граф.
2. Визначення відношень між цими вершинами і використання цих відносин для визначення ребер між вершинами графі. Граф може бути орієнтований або неорієнтований, зважений або незважений.
3. Ітерація алгоритму ранжування (PageRank) до його збіжності.
4. Сортування вершин базуючись на їх фінальній оцінці. Значення додані до кожної вершини використовуються для прийняття рішення щодо ранжування або вибору вершини.

Тепер необхідно застосувати цей загальний алгоритм для задачі реферування тексту. Спочатку потрібно побудувати граф, де вершини графа виконують роль текстових одиниць, які будуть надалі оцінені. Вершинами графа у даному випадку можна представити кожне речення в документі. Ребра між вершинами повинні показувати наскільки пов’язаними є певні два речення. Тобто, ребра між вершинами будуть представляти перекриття змісту в цих двох реченнях, а саме можна просто використати спільну кількість слів між двома реченнями. Крім того, щоб уникнути використання довгих речень, можна використати коефіцієнт нормалізації та розділити перекриття вмісту між двома реченнями на довжину кожного речення. Отже, нехай є два речення

S_i та S_j, де кожне речення має N_i кількість слів: $S_i = w_1^i, w_2^i, \dots, w_{N_i}^i$, тоді схожість між двома реченнями можна визначити за формулою:

$$Similarity(S_i, S_j) = \frac{|\{w_k | w_k \in S_i \,\&\, w_k \in S_j\}|}{\log(|S_i|) + \log(|S_j|)}$$

Можна використовувати і інші методи визначення схожості між двома реченнями, такі як косинус подібності, найдовша загальна підпослідовність, тощо.

Отже, нехай G = (V, E) – орієнтований граф з множиною вершин V та множиною ребер E. Для кожної вершини V_i, нехай $In(V_i)$ – це множина вершин які вказують на них, а $Out(V_i)$ – множина вершин, на які вказує вершина V_i. Тоді оцінка вершини V_i визначається як:

$$S(V_i) = (1 - d) + d \cdot \sum_{j \in In(V_i)} \frac{1}{|Out(V_j)|} S(V_j)$$

де d – коефіцієнт від 0 до 1, що моделює вірогідність переходу з даної вершини до іншої випадкової вершини у графі.

У контексті веб-серфінгу, цей алгоритм модельований таким чином, що користувач натискає на послання з вірогідністю d, і переходить на нову сторінку з вірогідністю 1-d, і зазвичай цей коефіцієнт встановлюється рівним 0.85.

Однак, у нашому випадку реферування тексту граф є зваженим, тому використовується модифікована формула:

$$WS(V_i) = (1 - d) + d \cdot \sum_{V_k \in In(V_i)} \frac{w_{ji}}{\sum_{V_k \in Out(V_j)} w_{jk}} WS(V_j)$$

Починаючи з довільних значень призначених кожній вершині у графі, ітерації продовжуються доки не буде досягнутий певний поріг значень. Після цього кожній вершині призначається оцінка, яка вказує важливість цієї вершини у графі.

1.3.3 Методи, які використовують нейронні мережі

Для розв'язання задачі реферування можна також використовувати нейронні мережі прямого поширення. У порівнянні з традиційними моделями, моделі, що створені на основі нейронного навчання, зазвичай досягають кращої точності, однак вони потребують початкові дані для тренування. Зазвичай нейронні мережі використовують такий загальний алгоритм:

1. Слова перетворюються на вектори дійсних чисел, що називаються вкладеннями слів (word embeddings)
2. Речення або документи закодовуються як вектори використовуючи вкладення слів
3. Ці репрезентації речень або слів після цього подаються в модель для вибору певних речень (екстрактне реферування) або генерування речень (абстрактне реферування).

Нейронні мережі можуть бути використані на кожному з цих кроків. Наприклад, на першому етапі можна використати методику word2vec, що дозволяє представляти кожне слово як вектор. Ці вектори представлені таким чином, що це дозволяє виявляти слова синоніми і знаходити рівень семантичної подібності між словами, представленими цими векторами. Інші моделі, які використовуються крім word2vec методики є CW вектори, GloVe. На другому кроці зазвичай використовуються мережі CNN або RNN для закодування, щоб знайти особливості тесту або речень. На третьому кроці

моделі нейронних мереж можна використовувати регресори для ранжування або вибору речень (для методів екстракції) або декодерів (для методів абстракції). [14]

Отже, для розв'язання задачі екстрактного реферування, модель для нейронного навчання повинна розв'язувати такі дві проблеми: як представляти слова для подальшого використання у нейронних мережах та як обирати необхідні речення для того, щоб кінцевий реферат містив всю необхідну інформацію та не містив повторів.

1.4 Rouge

Rouge є програмним пакетом, що містить інструменти для автоматичного визначення якості реферату за допомогою його порівняння з іншими рефератами, які були автоматично створені людьми. Методи зазвичай оцінюють лексичне перекривання між текстами, такі як n-грами, послідовності слів та пари слів між рефератами, створеними алгоритмами та ідеальними рефератами, створеними людьми. [16]

Залежно від функції, яка використовується для обчислення, є декілька типів для Rouge оцінок: ROUGE-N, ROUGE-L, ROUGE-W і ROUGE-S.

1.4.1 Rouge-N: Статистика спільних випадків N-грами

Rouge-N – це оцінка заснована на використовуванні n-грам між поточним рефератом та ідеальним рефератом.

$$RougeN = \frac{\sum_{S\in\{ReferenceSummaries\}} \sum_{gram_n \in S} Counnt_{match}(gram_n)}{\sum_{S\in\{ReferenceSummaries\}} \sum_{gram_n \in S} Count(gram_n)}$$

де n – довжина n-грам $gram_n$;

$Count_{match}(gram_n)$ - максимальна кількість n-грам що знаходяться як в поточному рефераті, так і в ідеальному рефераті.

Для того, щоб обрахувати Rouge оцінку, якщо існує декілька зразків рефератів, треба обчислити Rouge оцінку попарно між поточним рефератом s та кожним іншим зразковим рефератом r_i. Це може бути записано формулою:

$$RougeN_{multi} = argmax_i RougeN(r_i, s)$$

Такий самий спосіб може бути використаний і для інших оцінок Rouge-L, Rouge-W та Rouge-S, якщо існує декілька зразків рефератів.

1.4.2 Rouge-L: Найдовша загальна підпослідовність

Послідовність $Z = [z_1, z_2, \ldots, z_n]$ є підпослідовністю $X = [x_1, x_2, \ldots,]$, якщо існує зростаюча послідовність індексів $I = [i_1, i_2, \ldots, i_k]$ в X таких що для кожного $j = 1, 2, \ldots, k$ ми маємо $x_{ij} = z_j$. [17] Нехай дано дві послідовності X і Y, тоді найдовша загальна підпослідовність (LCS) X і Y, є спільною підпослідовністю з максимальною довжиною.

Тоді, для того, щоб використати LCS для оцінки реферату, ми оброблюємо речення в рефераті як послідовність слів. Чим довше є LCS між двома реченнями, тим більш схожими є ці речення. Нехай є два речення – X довжини m (речення з поточного реферату) та Y довжини n (речення зі зразку), тоді Rouge-L обраховується таким чином:

$$R_{ics} = \frac{LCS(X, Y)}{m}$$

$$P_{ics} = \frac{LCS(X, Y)}{n}$$

$$RougeL = \frac{(1 + \beta^2) R_{ics} P_{ics}}{R_{ics} + \beta^2 P_{ics}}$$

1.4.3 Rouge-W: Зважена найдовша загальна підпослідовність

Для покращення попереднього методу обчислення Rouge-L, можна запам’ятовувати довжину послідовних збігів до даного моменту як двовимірну динамічну таблицю обчислень LCS. Це називається WLCS, що використовує k для того, щоб вказувати довжину поточних послідовних збігів, що закінчуються словами x_i та y_j.

$$R_{wlcs} = f^{-1}(\frac{WLCS(X,Y)}{f(m)})$$

$$P_{wlcs} = f^{-1}(\frac{WLCS(X,Y)}{f(n)})$$

$$F_{wlcs} = \frac{(1+\beta^2)R_{wlcs}P_{wlcs}}{R_{wlcs}+\beta^2 P_{wlcs}}$$

де f^{-1} є оберненою функцією до f.

1.4.4 Rouge-S: Статистика появ Skip-Bigram

Skip-Bigram – це будь-яка пара слів, яка дозволяє довільні пропуски (для уникнення сполучних слів, тощо). Використання Skip-Bigram дозволяє вимірювати перекриття між поточним реченням та зразковим реченням.

Нехай є два речення – X довжини m (речення з поточного реферату) та Y довжини n (речення зі зразка), тоді Rouge-L обраховується таким чином:

$$R_{skip2} = \frac{SKIP2(X,Y)}{C(m,2)}$$

$$P_{skip2} = \frac{SKIP2(X,Y)}{C(n,2)}$$

$$RougeS = \frac{(1+\beta^2)R_{skip2}P_{skip2}}{R_{skip2}+\beta^2 P_{skip2}}$$

де SKIP2(X,Y) – кількість skip-bigram збігів між X та Y;

β – коефіцієнт, що контролює відносну важливість P_{skip2} та R_{skip2};

C – це функція комбінації.

1.4.5 Недоліки використання Rouge оцінок

Однак, такий показник як Rouge має серйозні обмеження, що не дозволяють адекватно порівнювати результати між декількома методами. Наприклад, оцінка Rouge може мати такі обмеження:

1. Вона оцінює лише вибір контенту, не враховуючи такі аспекти як плавність, граматичність, узгодженість речень.
2. Щоб оцінити вибір вмісту, методи Rouge оцінок зазвичай дивляться переважно на лексичне перекривання між текстами. Методи абстрактного реферування можуть генерувати реферати з тим самим вмістом, але без лексичного перекривання.
3. Враховуючи суб’єктивність реферування тексту, Rouge метрика була розроблена для використання з декількома рефератами для текстів на вхід. Однак, останні набори даних, такі як CNN/DailyMail містять лише один реферат для кожного тексту.

РОЗДІЛ 2 АВТОМАТИЧНЕ РЕФЕРУВАННЯ ТЕКСТУ НА ОСНОВІ ЗАПИТУ

2.1 Постановка задачі

В цій роботі буде розглянуте завдання однодокументної реферувальної системи, що буде базуватися на напрямку екстракції. Додатково до заданого тексту на вхід також буде подаватися запит користувача. Через відсутність наборів даних для реферування з запитом користувача, буде використаний підхід для навчання без учителя.

Отже, розглянувши існуючі реферувальні системи та алгоритми, можна навести вимоги до реалізації програмного застосунку та алгоритму:

- Алгоритм повинен базуватися на напрямку екстракції, працювати для однодокументних систем;
- Застосунок повинен підтримувати загальне реферування та реферування з запитом користувача;
- Результат роботи програмного застосунку має бути рефератом вхідного тексту. Текст результату має бути виділений відповідно на веб-сайті або показаний у вигляді окремого документа;
- Застосунок повинен працювати на веб-сайтах англійською мовою (пізніше може бути розширено на інші мови).

Для розробки веб-розширення був обраний напрям екстракції, а не абстракції. Напрям абстракції зазвичай вимагає використання різних засобів розуміння тексту та обробки природної мови. Через це більшість існуючих алгоритмів використовують саме напрям екстракції і тому він є більш дослідженим. Крім цього, методи абстракції зазвичай вимагають детального аналізу тексту та його обробки, що зазвичай займає велику кількість часу. Методи абстракції можуть бути придатними для загального реферування, коли обробку можна було б виконувати в офлайн-режимі. Однак, для задачі реферування на основі запиту, коли текст повинен бути оброблений досить швидко після створення запиту користувачем, реферування на основі абстракції може виявитися занадто довгим процесом. Незважаючи на те, що реферування використовуючи напрямок екстракції не може гарантувати узгодженості розповіді, такий підхід є досить інформативним для загального розуміння користувачем та оцінки релевантності необхідного фрагмента до

певного запиту. Також, зважаючи на середній розмір веб-сторінки, алгоритм екстрактного реферування може працювати досить ефективно не змушуючи користувача чекати занадто довго. [13] Однак, такий підхід реферування можна було б включити у веб-розширення як додаткову функцію.

2.2 Загальний процес реферування на основі запиту

Тепер розглянемо завдання реферування тексту, коли додатково до нього також на вхід є запит.

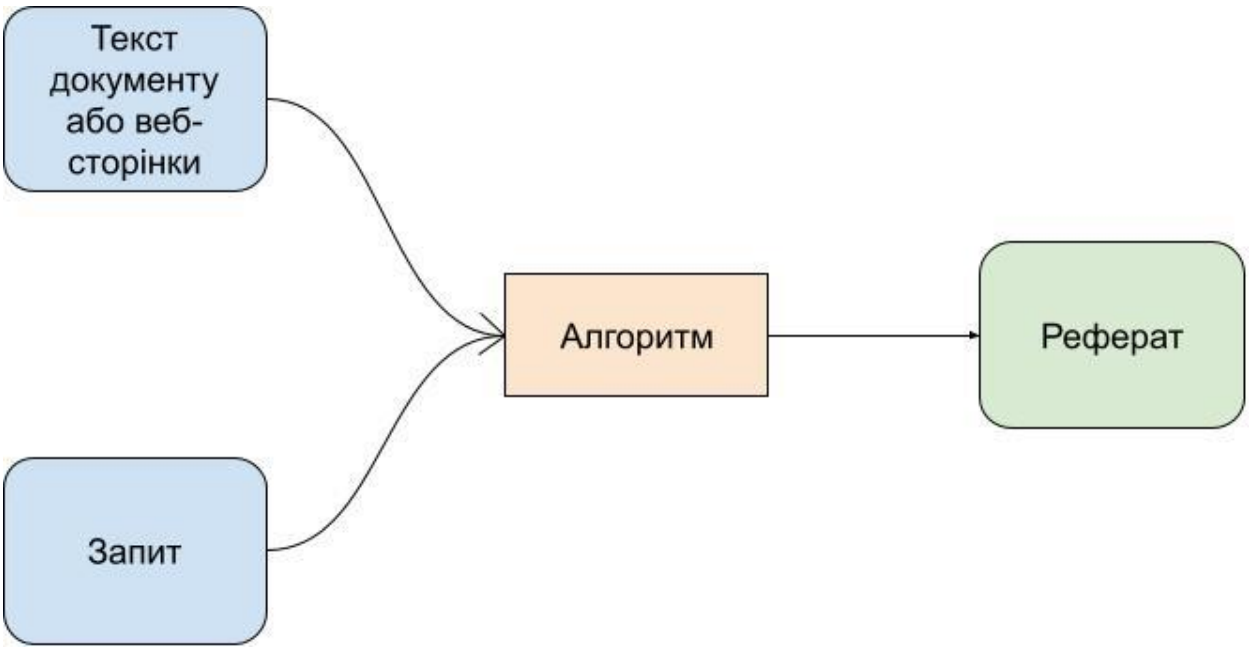


Рис. 2.1 Загальний процес реферування тексту на основі запиту

Алгоритми загального реферування стали досить популярними через наявність наборів даних, що містять пари документ-реферат. Реферування за запитом, з іншої сторони, є менш поширеним через відсутність таких наборів даних. Існуючі набори даних для реферування на основі запиту (такі як Dang, 2005 або Baumel et al. 2016) є відносно невеликими для навчання і в основному використовуються для оцінки роботи алгоритму [19]. Це обмежує

використання машинного навчання для розв'язання даної задачі. Через це дане завдання на разі залишається складним через відсутність запитів у наборах даних для загального реферування.

Через відсутність цих ресурсів, запропонований алгоритм буде використовувати знаходження релевантних речень за допомогою натренованої моделі SBERT та використовувати попередні здобутки з алгоритмів загального реферування.

2.3 Детальний опис запропонованого алгоритму

У цьому розділі буде описаний запропонований метод узагальнення тексту за допомогою запиту. Даний метод має бути використаний у веб-розширенні, тому для його побудови будуть також враховані його вимоги та обмеження. Запропонований спосіб складається з контролера для пошуку релевантних речень до запиту, генератора текстового реферату та вихідного модуля для виведення підсумків у вигляді реферату. Модуль для пошуку релевантних речень може бути за необхідності відключений, тоді реферат буде загальним.

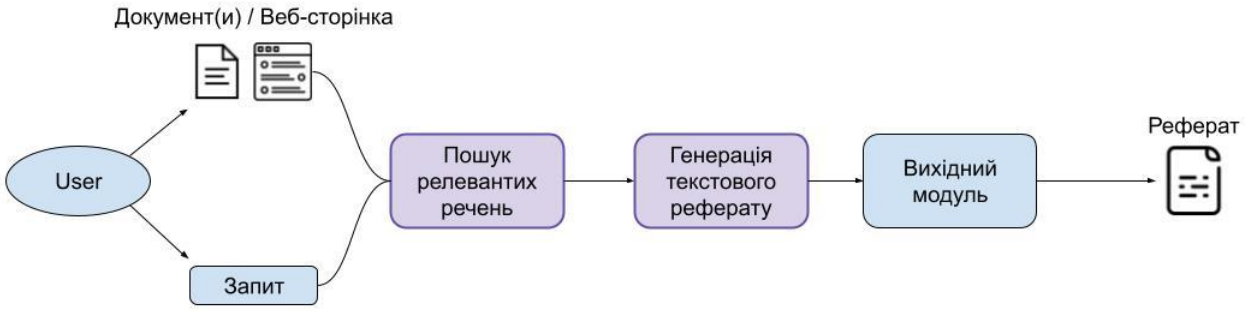


Рис. 2.2 Головна ідея запропонованого алгоритму для реферування тексту за допомогою запиту

2.3.1 Пошук релевантних речень

Для знаходження найбільш схожих речень до даного запиту, можна використовувати модель SBERT. Це спеціально модифікована модель BERT, що дозволяє знаходити схожі речення за менший проміжок часу, ніж це було би необхідно з BERT або RoBERT моделлю, зберігаючи точність моделі BERT [18]. Загальна структура алгоритму для пошуку релевантних речень до даного запиту буде виглядати таким чином:

1. Перетворення запиту у вектор за допомогою моделі SBERT;
2. Перетворення всіх речень у документі або веб-сторінці у вектор за допомогою моделі SBERT;
3. Знаходження речень, які мають найменший косинус подібності або скалярний добуток з запитом.

Для вибору моделі пошуку релевантних речень необхідно проаналізувати бажаний час роботи алгоритму, симетричність або асиметричність семантичного пошуку, вміст вхідного тексту та розмір моделі.

Для симетричного семантичного пошуку запит та речення повинні мати приблизно однакову довжину та вміст. Для асиметричного семантичного аналізу зазвичай є короткий запит (наприклад, запитання або ключові слова) і необхідно знайти довше речення, шо відноситься до даного запиту. В нашому випадку модель повинна працювати для асиметричного семантичного аналізу, тому необхідно розглядати MS MARCO моделі. Ці моделі були натреновані використовуючи запити з пошукової системи Bing.

Деякі моделі також є досить великими за розміром, що лімітує їх використання у Google Cloud функціях. Незважаючи на те, що такі моделі зазвичай мають найкращу якість, вони працюють повільніше та займають

багато місця. Інші моделі, що є меншими за розміром, є кращими для пошуку в режимі реального часу, що буде досить корисним для використання в подальшому у програмному застосуванні.

На останньому етапі необхідно обрати чи обраховувати скалярний добуток або косинус подібності. Це залежить від того, яка модель для використання була обрана на початку. Моделі, що використовують косинус подібності, зазвичай віддають перевагу пошуку більш коротких уривків тексту, тоді як моделі, що використовують скалярний добуток, надають перевагу пошуку більш довгих уривків.

Зважаючи на ці фактори була обрана модель msmarco-distilbert-base-v2, що є досить малою за розміром (253 мегабайти), працює для асиметричних запитів та налаштована під використання косинуса подібності під час обрахування подібності між реченнями.

2.3.2 Генерація текстового реферату

Після того, як були обрані релевантні речення до заданого запиту, для генерації текстового реферату можна було б використовувати будь-який метод, описаний у другому розділі для екстрактного реферування. Однак, через обмеженість обчислювальних ресурсів та необхідність запуску даного алгоритму у реальному часі, було обрано алгоритм на основі LSA, що було описано у розділі 1.3.1. Алгоритм також не потребує додаткової інформації про текст та попереднього тренування. Для заповнення значень матриці в алгоритмі LSA був обраний параметр TF-IDF.

РОЗДІЛ 3 ВИКОРИСТАННЯ РОЗРОБЛЕНОГО АЛГОРИТМУ У ВЕБ-РОЗШИРЕННІ

3.1 Аналіз відомого програмного забезпечення

На сьогодні є багато програмних реалізацій для реферування тексту, однак недостатньо рішень, які використовують реферування, що базується на запиті користувача. Наведемо деякі з них:

- Sumy [6] – модуль для Python, що дозволяє знайти фрагменти різними методами, такими як Luhn, Edmundson, LSA, TextRank and LexRank, SumBasic, KL-Sum, Reduction. Є також деякі інші схожі бібліотеки, але ця реалізація є найбільш популярною та підтримує більшість існуючих методів.

- AutoSummarizer [7] – веб-сайт для реферування тексту, можна обрати лише кількість речень, які будуть знаходитись у спрощеному фрагменті, недоступний вибір алгоритму. Інші аналоги у вигляді веб-сайтів також мають схожу функціональність, можуть дозволяти вибір алгоритму або відсоток стиснення тексту.

- TLDR This [5] – існує у вигляді веб-версії та у вигляді веб-розширення, однак при її використанні немає можливості для введення запиту.

Крім того, у роботі були проаналізовані також і інші додатки, але вони мають схожі недоліки - більшість програмних рішень існують у вигляді веб-сайтів, тому необхідно вручну копіювати текст для реферування, немає підтримки реферування за запитом або необхідно використовувати алгоритм вручну.

Існує мало додатків, що дозволяють необізнаним у темі людям використовувати ці алгоритми. Використання веб-розширення може бути

зручнішим у даному випадку через можливість запуску на веб-сайті одразу, без додаткового копіювання тексту. Виділені фрагменти у самому тексті також можуть допомогти побачити, де саме знаходяться найважливіші частини. Це може бути реалізовано, тому що в даній роботі розглядається екстрактне реферування, і кінцевий результат буде складатися із речень з початкового тексту який подавався на вхід. Також більшість додатків мають лише загальне реферування тексту, без можливості використання запиту.

Для розробки веб-розширення також був обраний саме метод екстракції, бо цей підхід є більш зручним для виділення речень у тексті. Цей спосіб є одним з найбільш досліджених та він не потребує додаткового навчання використовуючи схожі документи перед цим. В результаті веб-розширення зможе працювати з будь-якими текстами, не буде обмежень на теми текстів. Однак, у майбутньому можна також додати функціональність для абстрактного реферування в окремому вікні.

3.2 Деталі технології розробки веб-розширення

Метою для цієї роботи було розробка веб-розширення, яке могло б використовувати описаний у минулих розділах алгоритм. Через те, що алгоритм є однодокументним (тобто він не потребує більше документів з даної теми для попередньої обробки), то даний підхід може бути використаний на будь-якому веб-сайті без попереднього тренування. Зважаючи на середній розмір тексту на веб-сторінках та використаний алгоритм, реферат є готовим відносно швидко через декілька секунд після спрацювання алгоритму.

Через те, що обробка тексту є доволі ресурсомістким процесом, для розробки розширення був обраний браузер Chrome та використана Google Cloud функція для оброблення тексту у хмарному середовищі використовуючи мову Python.

Архітектура

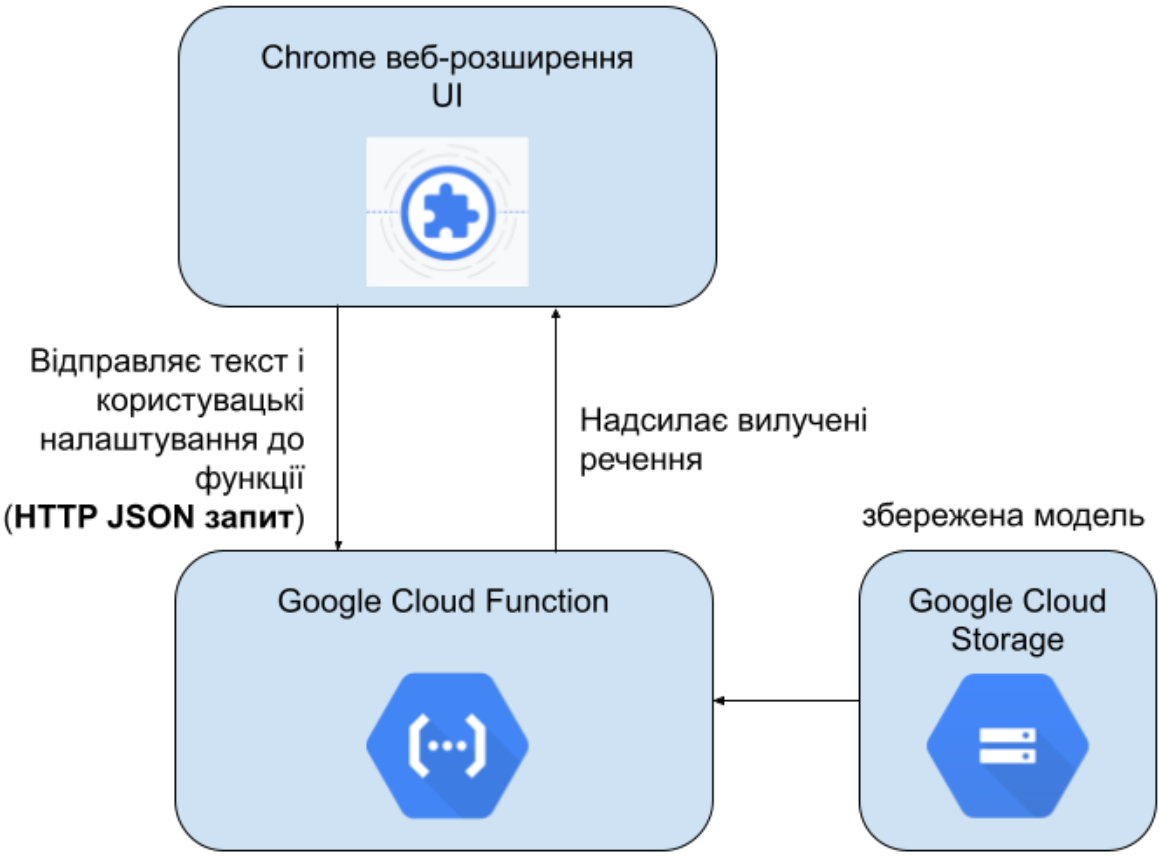


Рис. 3.1 Архітектура додатка

В якості UI у цьому випадку використовується Chrome веб-розширення, яке буде передавати користувацькі налаштування і текст до Google Cloud функції використовуючи API запити. Альтернативно, можна було використати інші засоби для виконання роботи на сервері або для UI, однак така архітектура дозволяє обробляти текст з будь-якого ресурсу чи веб-сайту.

Chrome веб-розширення містить файл manifest.json, де описується основна інформація про розширення та папок для javascript, CSS, HTML файлів. При запуску додатка запускається popup.html файл, що показує основний інтерфейс. Після цього при натисненні на кнопку Find Text, додаток

запам’ятовує налаштування та текст з веб-сайту і надсилає його до функції у хмарі за допомогою Fetch API. Запит надходить до відповідного URL вказаного у Google Cloud інтерфейсі. Після отримання результату через декілька секунд у вигляді Promise об’єкту, додаток запускає скрипт, в якому потрібні речення виділяються на веб-сторінці.

Google Cloud функція отримує задані користувачем значення та текст з веб-сайту та оброблює його використовуючи згаданий алгоритм. При цьому через наявність механізмів безпеки для сучасних браузерів потрібно запобігти помилкам з CORS. Тому, коли клієнт хоче надіслати HTTP запит (в цьому випадку клієнтом є розширення браузера), то браузер спочатку насилає “preflight” запит, який є OPTIONS запитом. На серверній стороні, додаток має відповісти на цей запит з інформацією про дозволені методи та кодом 204. Після цього звичайний запит також може бути оброблений, однак до загального заголовка також повинні бути додані необхідні CORS заголовки для необхідної обробки веб-розширенням.

Для того, щоб зменшити час на завантаження та розмір моделі на етапі пошуку релевантних речень, було використано Google Cloud Storage. Для серіалізації та десеріалізації моделі була використана бібліотека Pickle. Модель в коді була оголошена як глобальна змінна для того, щоб вона була збережена у кеші та повторно використана для подальших викликів функцій.

3.3 Опис інтерфейсу

Для початку роботи з додатком, необхідно його встановити як розширення до веб-браузеру Chrome. Після цього розширення буде доступне для використання на будь-якому веб-сайті.

При запуску з’являється вікно, де необхідно ввести даний запит та обрати бажаний розмір фрагмента. Запит можна не вводити, тоді в результаті буде показаний загальний реферат тексту без його відповідності до запиту.

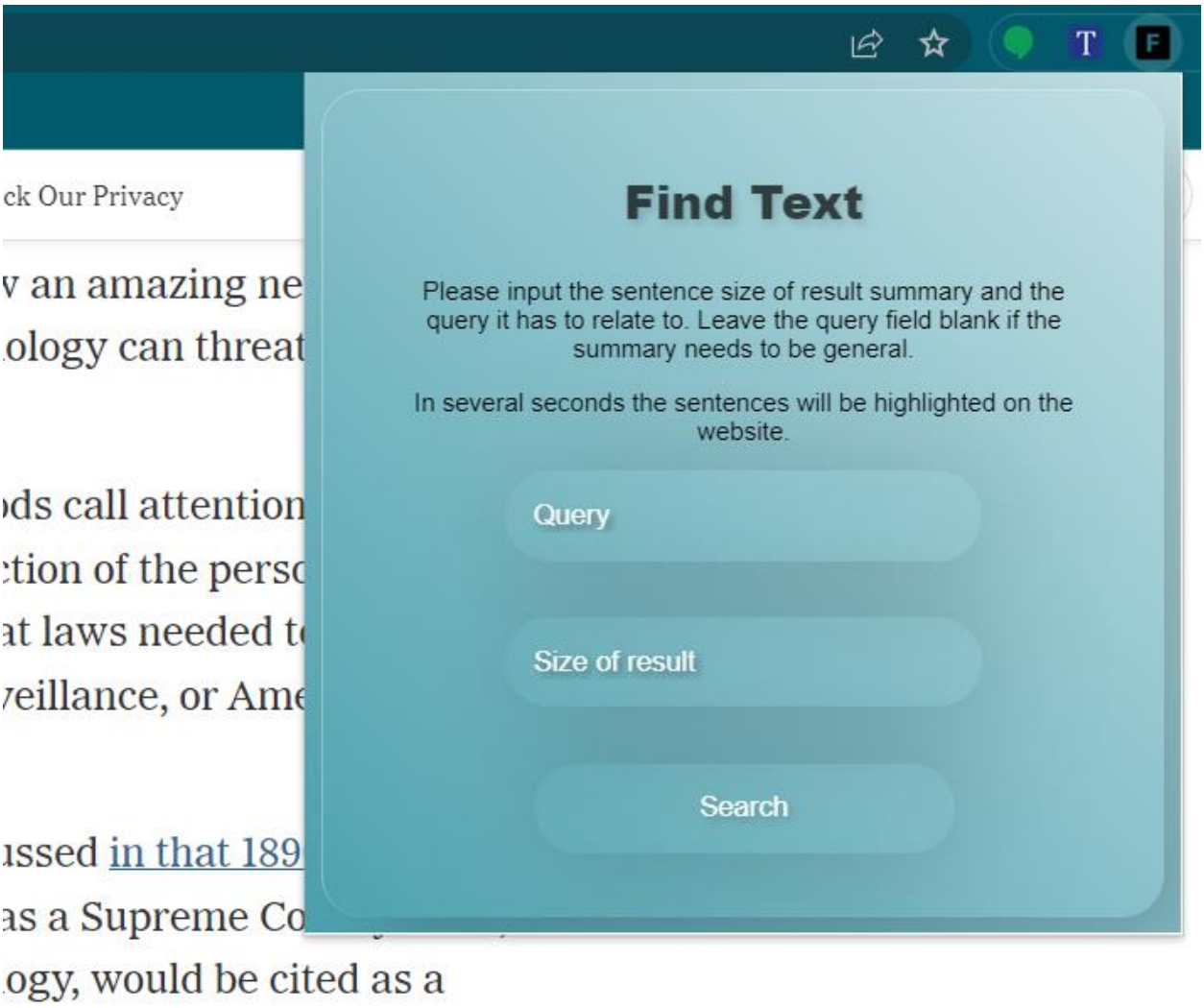
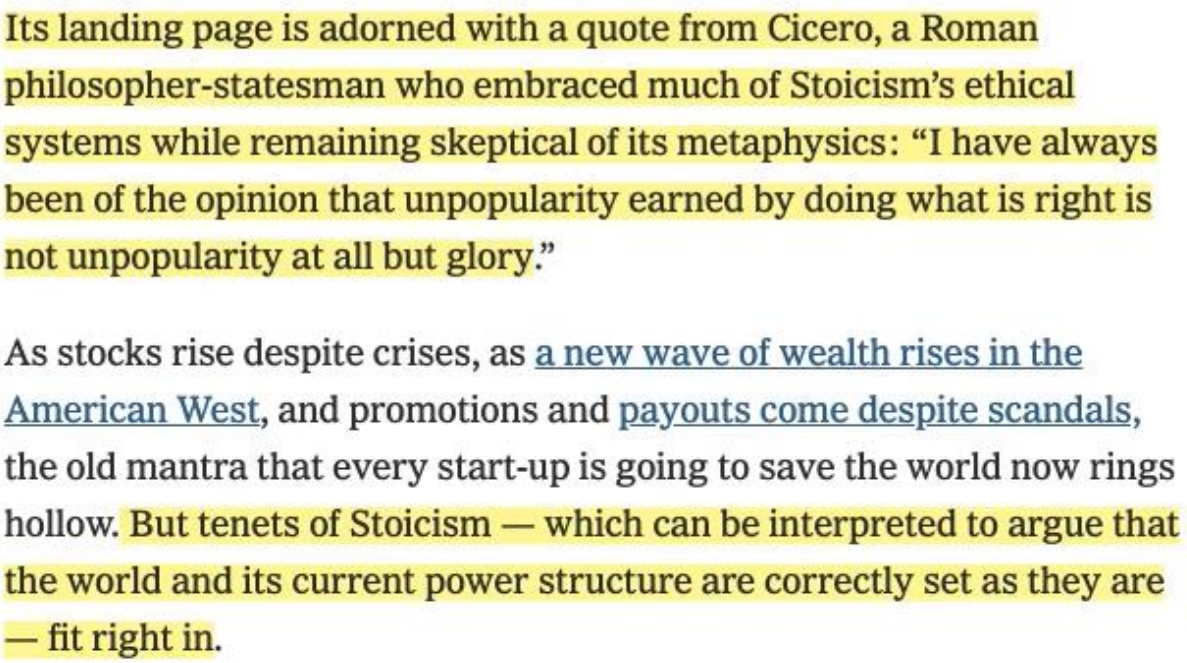


Рис. 3.2 Інтерфейс веб-розширення

Після натиснення на кнопку Search з’явиться сповіщення про те, що запит було надіслано. Через деякий час, коли функція завершить свою роботу, результат буде виділено на веб-сайті в залежності від попередніх налаштувань користувача. Якщо запит не було обрано попередньо, то буде відбуватися загальне реферування тексту.

Іноді кількість речень в результаті може виявитись меншою за кількість обрану користувачем в налаштуваннях, якщо кількість релевантних до запиту речень була менше, ніж бажаний розмір фрагмента в налаштуваннях користувача. Такий варіант також можливий якщо кількість обраних користувачем речень виявилась більшою, ніж кількість речень у всьому тексті. В такому випадку буде виділена менша кількість речень, ніж була обрана попередньо.


Its landing page is adorned with a quote from Cicero, a Roman philosopher-statesman who embraced much of Stoicism's ethical systems while remaining skeptical of its metaphysics: "I have always been of the opinion that unpopularity earned by doing what is right is not unpopularity at all but glory."

As stocks rise despite crises, as a new wave of wealth rises in the American West, and promotions and payouts come despite scandals, the old mantra that every start-up is going to save the world now rings hollow. But tenets of Stoicism — which can be interpreted to argue that the world and its current power structure are correctly set as they are — fit right in.

Рис. 3.3 Зразок результату виділеного тексту на веб-сайті

РОЗДІЛ 4 ОЦІНКА МОДЕЛІ

4.1 Час роботи алгоритму

Час роботи алгоритму при використанні Google Cloud функції залежить від того, чи була функція запущена вперше. Найперший запуск відбувається після того, як перший запит надійшов до Google Cloud функції після її розгортання на сервері. Для обробки цього запиту також відбувається

завантаження моделі з Google Cloud Storage. Після того, як цей запит було оброблено, модель залишається ініціалізованою для повторного використання у наступних запитах. Для всіх наступних запитів час завантаження буде зменшений за рахунок використання завантаженої в пам’ять моделі. В середньому, обробка тексту займає декілька секунд в залежності від розміру тексту та обраного методу загального реферування (5-10 секунд) або реферування за запитом (20-30 секунд) для веб-сторінок середнього розміру.

4.2 Тестування

Після того, як функція була розгорнута на Google Cloud Functions сервері, її було протестовано за допомогою секції тестування у Google Cloud Function Dashboard. Також було виконано тестування за допомогою бібліотеки Curl, що дозволило протестувати програмний продукт надсилаючи HTTP запити з різними тестовими даними. Це дозволило ізолювати тестування помилок серверної частини Google Cloud функції та Chrome веб-розширення.

Після тестування окремих частин програмного забезпечення (Google Cloud функції та Chrome веб-розширення) було виконане інтеграційне тестування. Було знайдено, що функції працюють коректно між собою: HTTP запит містить необхідну інформацію про налаштування користувача, був коректно отриманим та обробленим функцією. Результат був показаний на веб-сайті шляхом виділення тексту.

Google Cloud функція також використовувала бібліотеку google-cloud-logger для логування для перевірки коректності роботи функції у реальному часі та знаходження помилок.

ВИСНОВКИ

У цій роботі було розглянуто алгоритми для розв'язання задачі екстрактного реферування текстів. Був розроблений алгоритм для реферування тексту на основі запиту. Використовуючи цей алгоритм, був побудований програмний продукт у вигляді веб-розширення, що дозволяє знайти фрагмент тексту, який найбільш точно підсумовує текст поданий на вхід та відповідає даному запиту.

Можливі також шляхи покращення і модифікація інших алгоритмів екстрактного або абстрактного реферування для задачі реферування на основі запиту. Програмний продукт може бути розширений, щоб була можливість використання абстрактного реферування або наявності інших налаштувань для кастомізації отриманого результату. На разі веб-розширення працює з веб-сайтами лише англійською мовою, однак у майбутньому може бути розширене і на інші мови. Також, через те, що веб-розширення дозволяє застосовувати алгоритм реферування не лише для документа, а і веб-сторінки у вигляді HTML файлу, можна було б покращити алгоритм використовуючи заголовки або інформацію з посилань. Багато веб-сторінок також містять додаткову нерелевантну інформацію, наприклад рекламу або додаткові посилання, що не належать основному тексту.

Отже, в результаті реалізовано закінчений програмний продукт – веб-розширення, що працює коректно та відповідає поставленій меті. Після закінчення виконання програми проаналізовано її переваги та недоліки, на основі яких визначено можливі шляхи подальшого покращення.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Python Style Guide [Електронний ресурс] – Режим доступу: https://google.github.io/styleguide/pyguide.html
2. See, Abigail & Liu, Peter & Manning, Christopher. (2017). Get To The Point: Summarization with Pointer-Generator Networks. 1073-1083. 10.18653/v1/P17-1099.
3. Liu, Peter & Saleh, Mohammad & Pot, Etienne & Goodrich, Ben & Sepassi, Ryan & Kaiser, Lukasz & Shazeer, Noam. (2018). Generating Wikipedia by Summarizing Long Sequences.
4. Liu, Yang. (2019). Fine-tune BERT for Extractive Summarization.
5. TLDR This [Електронний ресурс] – Режим доступу: https://tldrthis.com/
6. Sumy [Електронний ресурс] – Режим доступу: https://github.com/miso-belica/sumy
7. AutoSummarizer [Електронний ресурс] – Режим доступу: https://autosummarizer.com/
8. Ozsoy, Makbule & Alpaslan, Ferda & Cicekli, Ilyas. (2011). Text summarization using Latent Semantic Analysis. J. Information Science. 37. 405-417. 10.1177/0165551511408848.
9. Strang, Gilbert. (2003). Introduction to Linear Algebra.
10. Kazemi, Ashkan & Pérez-Rosas, Verónica & Mihalcea, Rada. (2020). Biased TextRank: Unsupervised Graph-Based Content Extraction.
11. Mihalcea, Rada & Rada, & Tarau, Paul & Paul. (2004). TextRank: Bringing Order into Texts.
12. PL. Prabha, M. Parvathy. (2020). Extractive and Abstractive Text Summarization Techniques.
13. Wang, Changhu & Jing, Feng & Zhang, Lei. (2007). Learning Query-Biased Web Page Summarization. 555-562. 10.1145/1321440.1321518.

14. Dong, Yue. (2018). A Survey on Neural Network-Based Summarization Methods.
15. Lin, Chin-Yew. (2003). Automatic Evaluation of Summaries Using n-gram Co-occurrence Statistics. 71-78. 10.3115/1073445.1073465.
16. Lin, Chin-Yew. (2004). ROUGE: A Package for Automatic Evaluation of summaries. Proceedings of the ACL Workshop: Text Summarization Braches Out 2004. 10.
17. Cormen, T. R., C. E. Leiserson, and R. L. Rivest. (1989). Introduction to Algorithms
18. Reimers, Nils & Gurevych, Iryna. (2019). Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks.
19. Xu, Yumo & Lapata, Mirella. (2021). Text Summarization with Latent Queries.