МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ ДОНЕЦЬКИЙ
НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ІМЕНІ ВАСИЛЯ СТУСА

ОЛЕКСІЄНКО ВАЛЕНТИН ВІТАЛІЙОВИЧ

Допускається до захисту:
завідувач кафедри
інформаційних технологій
д.т.н., доцент
_____Т.В. Нескородєва
«___»_____2022 р.

«ДОСЛІДЖЕННЯ КОНТЕНТУ САЙТУ ТУРИСТИЧНОГО АГЕНСТВА НА ПІДСТАВІ МЕТОДІВ DATA SCIENCE»

Спеціальність 122 Комп'ютерні науки

Кваліфікаційна (магістерська) робота

Науковий керівник:
Є.Є. Федоров, професор кафедри
інформаційних технологій,
д.т.н., професор

_____
(підпис)

Оцінка: _____/_____/_____
(бали/за шкалою ЄКТС/за національною шкалою)

Голова ЕК: _____
(підпис)

Вінниця - 2022

ЗМІСТ

АНОТАЦІЯ

Олексієнко В.В. Дослідження контенту сайту туристичного агенства на підставі методів Data Science. ДОСЛІДЖЕННЯ КОНТЕНТУ САЙТУ ТУРИСТИЧНОГО АГЕНСТВА НА ПІДСТАВІ МЕТОДІВ DATA SCIENCE. Спеціальність 122 «Комп'ютерні науки», Освітня програма «Комп'ютерні науки (Data Science)» (СО «Магістр»). Донецький національний університет імені Василя Стуса, Вінниця, 2022.

У кваліфікаційні роботі досліджено процеси автоматичного реферування тексту, розглянуто методи та системи автоматичного реферування тексту, розгянуто бібліотеки JavaScript, React, Type Script, тощо. У роботі спроектовано веб-сайт туристичного агенства, який базується на мові програмування Java Script, спроектована база даних Mock API, а також можливість додавання опису певного туристичного туру за допомогою методів Data Science - автоматичного реферування. Запропонована система впроваджена у веб-сайт туристичного агенства. Ключові слова: автоматичне реферування тексту(summarization), Lex-rank alghoritm, туристичні агенства, великі дані.

ANNOTATION

Oleksiienko V.V. Study of the content of the travel agency website based on Data Science methods. RESEARCH OF THE CONTENT OF THE TOURIST AGENCY SITE BASED ON DATA SCIENCE METHODS. Specialty 122 "Computer science", Educational program "Computer science (Data Science)" (SO "Master"). Vasyl Stus Donetsk National University, Vinnytsia, 2022.

In the qualification work, the processes of automatic text abstracting were investigated, the methods and systems of automatic text abstracting were considered, JavaScript, React, Type Script libraries were accelerated, etc. In the work, a travel agency website based on the Java Script programming language was designed, a Mock API database was designed, as well as the possibility of adding a description of a certain tourist tour using Data Science methods - automatic abstracting. The proposed system is implemented in the website of the travel agency. Keywords: automatic text summarization, Lex-rank algorithm, travel agencies, big data.

ВСТУП

Сьогодні туристичним компаніям необхідно мати якогомога зручніший та привабливий по формі веб-сайт, щоб пропонувати свої послуги людям, які хочуть мандрувати.

Для цього туристичним фірмам потрібно використовувати веб-сайт задля того щоб у зрозумілій формі показати які послуги може забезпечити туристична компанія і скільки це буде коштувати користувачу.

Так на сьогоднішній день в цій сфері присутня сувора конкуренція туроператорів, єдиним засобом втриматись на ринку туристичних послуг необхідно мати в наявності оптимальний веб-ресурс з інформацією, зображення туристичного курорту та зручний інтерфейс для користувача.

Неодмінною складовою туристичного сайту є пошук потрібного туру, щоб користувач не марнував години на пошук та замовлення турів, білетів, готелів, екскурсій тощо. На сьогоднішній час релевантний сайт містить динамічні дані, приємний інтерфейс та має на меті принести користь кожному відвідувачу. Такі сучасні та складні системи рідко створюються одноосібно. Найчастіше розробка ділиться на три основні етапи: – створення дизайну; – втілення статичної кодової реалізації дизайну; – перетворення статичної кодової реалізації у динамічну веб-систему.

Перший етап виконує дизайнер, другий етап – Front-end розробник, третій – Back-end розробник. Серверна частина сайту лишається невидимою для звичайного користувача, але вона є основою яка закладає фундамент, на якому в подальшому веб-система буде розвиватися та розширюватися. Темою дипломного проекту є розробка веб-сайту для туроператора, з використанням методів Data Science.

Метою магістерської роботи є дослідження контенту сайту туристичного агенства на підставі методів Data Science та розробка подібного сайту.

Досягнення поставленої мети передбачало розв‘язання таких задач:

- розгляд методів та систем дослідження контенту сайтів;
- проектування системи дослідження контенту сайту туристичного агенства на підставі методів Data Science;
- розляд аналогів веб-сайтів туристичних агенств
- моделювання сайту UML-засобами
- розробка сайту за допомогою програмних інстументів JavaScript, React тощо
- розгляд методів автоматичного реферування тексту
- розроблення програмного інструментарію побудови
- UX веб-сайту туристичних агенцій
- системи дослідження контенту сайту туристичного агенства на підставі методів Data Science.

Об’єктом виступають процеси дослідження контенту сайтів.

Предметом є методи Data Science дослідження контенту сайту.

При проведенні досліджень та вирішенні вищезазначених задач використовувались: методи автоматичного реферування, методи глибокого машинного навчання, інстументри та бібліотеки з них: JS, React, TypeScript, Bootstrap, NodeJS, MokeAPI, тощо.

Наукова новизна роботи полягає у розробці системи дослідження контенту сайту туристичного агенства на підставі методів Data Science, що дозволила скоротити час на обробку вхідної текстової інформації та збільшити точність проведення її аналізу.

Практичне значення одержаних результатів полягає в тому, що розроблені теоретичні та методичні положення, а також практичні рекомендації щодо дослідження контенту сайту туристичного агенства на підставі методів Data Science.

РОЗДІЛ 1. ПОСТАНОВКА ЗАДАЧІ ТА ОГЛЯД АНАЛОГІВ, МЕТОДІВ ТА СИСТЕМ

1.1 Актуальність проблеми

Останнім часом на розвиток туристичного бізнесу великий вплив мають Інтернет-технології і все частіше у всесвітній павутині можна знайти різноманітні сайти присвячені розвитку туристичної індустрії, туристичним фірмам, агентствам, а також санаторіям, пансіонатам, базам відпочинку та готелям. Саме тому необхідною частиною будь-якого сучасного туристичного підприємства є наявність сайту, який може відгравати роль звичайної реклами у мережі Інтернет, а може надавати послуги пошуку, замовлення та довідки про місця та тури, які цікавлять клієнта.

Сайт туристичної компанії має бути зручним та зрозумілим для будь-якого користувача, мати добре спроектовану навігацію, доцільну інформацію у зручному вигляді, оскільки до туристичних компаній звертаються саме ті люди, які не хочуть бути заклопотаними організацією власного відпочину, тому сайт має бути максимально простим та легким для сприйняття.

Вдало спроектований інтерфейс сайту «приємний для ока» та викликає довіру у користувача, інакше складається думка що туристична компанія вже не працює або вона зовсім не дбає про свою репутацію. Ще одною важливою річчю для туристичної компанії є можливість інформувати своїх клієнтів поза сайтом, адже користувачам незручно кожен день заходити на сторінку компанії для отримання новин та про цікаві і вигідні пропозиції. Тому на сайті має бути спосіб поширення новин поза сайтом. Одним з таких способів є поштова розсилка.

Але останнім часом більшість користувачів частіше читають стрічку новин у соціальних мережах ніж перевіряють пошту на наявність нових листів. Тому ведення групи у «Facebook» або «Instagram» є необхідною умовою для інформування клієнтів туристичної компанії.

Метою є розробка сайту для туристичної компанії.

Перед подорожжю бувають кілька запитань, які виникають у нашому мозку, перш ніж ми навіть думаємо про подорож кудись. Не кожен може спланувати ідеальний відпочинок. Люди, які менше подорожують, не мають уявлення про те, як спланувати свій маршрут, до того ж, плануючи відпустку, можна почуватися безпорадним і неосвіченим.

Хтось завжди хотів поїхати в закордонну подорож, але був не впевнений, який напрямок для нього найкращий або які місця він/вона хочете відвідати, завжди простіше скористатися послугами туристичного агентства. За допомогою туристичних агентств туристи не переживають через домовленості та важливу інформацію, що може заощадити багато часу та зусиль. Вони чудово впораються з усіма необхідними для вас подорожами та подбають про те, щоб ваша подорож була безпечною.

Чому слід віддати перевагу туристичному агентству, щоб забронювати наступну відпустку:

1) Експертне керівництво

Ніхто не може зробити аналіз подорожей краще, ніж туристичне агентство, оскільки у них є експерти з належною освітою та підготовкою. Досвід і знання, накопичені роками, роблять їх кращими в усьому, що вони роблять. Вони можуть надати експертні пропозиції та попередження на основі свого особистого досвіду, щоб контролювати ваш бажаний відпочинок. Вони мають велику мережу, щоб надати вам інформацію, яку ви не можете знайти на веб-сайтах.

2) Економія часу

Уникайте головного болю та дозвольте вашому турагенту керувати всіма вашими потребами. Багато туристів тепер відчувають, що у них не так

багато часу на подорожі. Проводити години часу в Інтернеті та оглядати пам’ятки для ідеальної подорожі може зайняти багато днів. Туристичні агенції можуть провести дослідження та спланувати всю вашу відпустку, вони знають, як виконати роботу, зважаючи на ваші специфікації та дати.

3) Зручність

Туристичні агенції – це єдине рішення для ваших потреб у подорожах, вони можуть впоратися з усіма функціями вашої відпустки – від перельотів, готелів до огляду визначних пам’яток, оренди автомобілів та інших заходів. Ви можете вони подбати про те, щоб ви подорожували без будь-яких труднощів і насолоджувалися відпусткою без проблем. Вони можуть координувати та обговорювати деталі поїздки, щоб уникнути трагедій у подорожі.

4) Цінність

Туристичні агенції мають розширені зв’язки з туристичними постачальниками по всьому світу та мають доступ до ексклюзивних пропозицій. Вони можуть домовитися про найкращі пропозиції для своїх клієнтів, щоб надати їм найкращу вартість. Вони також пропонують широкий вибір залежно від плану витрат.

5) Індивідуальне обслуговування

Турагентство/агенти працюють на клієнтів, а не на постачальників туристичних послуг. Якщо під час подорожі ви зіткнетеся з будь-якими проблемами або змінами в плані, Інтернет вам не допоможе, але турагент готовий вислухати вас і надати вам пораду. Це перевага, яку шукає кожен мандрівник, подорожуючи в іншу країну, де ви не розмовляєте однією мовою.

6) Платіть майже однаково

Найкраща перевага бронювання через турагентів полягає в тому, що ви можете заощадити багато грошей. Зазвичай вони співпрацюють із готелями, авіакомпаніями, компаніями з прокату автомобілів, які можуть пропонувати різноманітні знижки або спеціальні тарифи, до яких ви не можете отримати доступ. Крім того, їхня плата за обслуговування майже безкоштовна або дуже мала для клієнтів, оскільки вони заробляють на комісійних продажах.

7) Правильні документи

Під час міжнародних подорожей людей часто плутають із візовою процедурою та документацією. Туристичні агенти також переконуються, що заявник добре поінформований про необхідні документи та направляють його через процедуру отримання візи.

1.2 Огляд методів автоматичного реферування тексту

В даний проект, буде добавлена можливість автоматичного реферування тексту для адміністрації сайту перед добавлянням якогось нового туру, для того щоб в користувача була можливість читати тільки необхідну йому інформацію. Адже по статистиці, люди в першу чергу дивляться на фото, а вже після того читають певну інформацію; якщо брати дану сферу, то це ціна, короткий опис про тур та що в цей тур входить.

Розберемо детально, що таке автоматичне реферування тексту.

Реферування тексту зазвичай використовується кількома веб-сайтами та програмами для створення стрічки новин і підсумків статей. Це стало для нас дуже важливим через нашу зайнятість. Ми віддаємо перевагу коротким резюме з усіма важливими моментами, а не читанню цілого звіту та самостійному його підсумкуванню. Тому було зроблено декілька спроб автоматизувати процес підведення підсумків.

Реферування — це техніка скорочення довгих текстів таким чином, щоб спрощення містило всі важливі моменти фактичного документа.

В основному є чотири типи спрощення:

- **Single Document Summary**: Спрощення окремого документа
- **Multi-Document Summary**: Спрощення з кількох документів
- **Query Focused Summary:** Підсумок конкретного запиту
- **Informative Summary:** Містить короткий виклад повної інформації.

Підходи до автоматичного реферування:

В основному існує два типи реферування:

- **Extraction-based Summarization:** Реферування на основі вилучення, підхід вилучення передбачає підбір найважливіших фраз і рядків із документа. Потім він об’єднує всі важливі рядки для створення спрощення. Отже, у цьому випадку кожен рядок і слово спрощення тексту фактично належать оригінальному документу, який підсумовується.
- **Abstraction-based Summarization:** Реферування на основі абстракції, абстрактний підхід передбачає реферування на основі глибокого навчання. Таким чином, він використовує нові фрази та терміни, відмінні від фактичного документа, зберігаючи ті самі пункти, як і те, як ми фактично підсумовуємо. Отже, це набагато складніше, ніж екстрактивний підхід.

По теорії та практиці є досвід, що екстрактивны реферування іноді працюють краще, ніж абстрактні, ймовірно тому, що екстрактивні не вимагають генерації природної мови та семантичних представлень.

Методи автоматичного реферування тексту:

1. **Evaluation methods - Методи оцінювання**

В даному методі пристуні два типи оцінок:

- Оцінка людини
- Автоматичне оцінювання

Оцінка людини: бали призначаються експертами-людьми на основі того, наскільки добре реферування тексту охоплює пункти, відповідає на запити та інші фактори, як граматичність і відсутність надлишків.

Автоматичне оцінювання:

ROUGE: ROUGE розшифровується як Recall-Oriented Understudy for Gisting Evaluation. Це метод, який визначає якість скорочення тексту, порівнюючи його з іншими текстами, зробленими людьми як довідковими. Щоб оцінити модель, існує ряд посилань, створених людьми, і згенерований машиною певний текст кандидата. Інтуїція, що стоїть за цим, полягає в тому, що якщо модель створює гарне резюме, то вона повинна мати спільні частини, що перекриваються з людськими посиланнями. Його запропонував Чін-Ю Лін з Каліфорнійського університету.

Поширені версії ROUGE:

ROUGE-n: вимірюється на основі порівняння результату, створеного машиною, та еталонного результату на основі n-грамів. N-грама — це безперервна послідовність із n елементів із заданого зразка тексту чи мови, тобто це просто послідовність слів. Біграми означають два слова, триграми означають 3 слова і так далі. Зазвичай використовуються біграми.

$$\text{ROUGE-}n = \frac{p}{q}$$

Рисунок 1.2.1 – ROUGE-n

«Де p — це «кількість загальних n-грамів між кандидатом і еталонним документом тексту», а q — «кількість n-грамів, виділених лише з еталонного документу з текстом»

ROUGE-L: Чим довша найдовша спільна підпослідовність у двох текстах,

тим вони схожі. Отже, він гнучкий, ніж n-грам. Він призначає бали на основі того, наскільки довгою може бути послідовність, яка є спільною для згенерованого машиною кандидата та еталонного зразка людини.

ROUGE-SU: Концепція пропуску біграм і уніграм. В основному він дозволяє або розглядає біграму, якщо між двома словами є інші слова, тобто біграми не обов'язково повинні бути послідовними словами

ROUGE-2 є найпопулярнішим:

$$ROUGE-2 = \frac{\sum_{s\in\{\text{RefSummaries}\}} \sum_{\text{bigrams } i\in S} \min(count(i,X), count(i,S))}{\sum_{s\in\{\text{RefSummaries}\}} \sum_{\text{bigrams } i\in S} count(i,S)}$$

Рисунок 1.2.2. – ROUGE-2

Де для кожної біграми «i» ми обчислюємо мінімальну кількість разів, коли вона з'явилася у створеному документі X і довідковому документі S, для всіх довідкових документів, поділену на загальну кількість разів, коли кожна біграма з'являється у всіх довідкових документах. Він базується на балах BLEU.

2. **Реферування на основі функцій**:

Розроблений Г. П. Луханом в IBM у 1958 році. У документі запропоновано, що важливість речення є функцією слів, які часто зустрічаються в документі. Якщо говорити детально, то алгоритм вимірює частоту слів і фраз у документі та визначає важливість речення, враховуючи слова в реченні та їх частоту. У ньому зазначено, чи є в реченні слова з більшою частотою, Це важливо, але тут ми не включаємо загальні слова, такі як «a», «the».

3. **Екстрактивне резюмування**

Метод екстракційного резюмування створюють резюме шляхом вибору підмножини речень в оригінальному тексті. **Extractive Summarizers** спочатку створюють проміжне представлення, головним завданням якого є виділення або видалення найважливішої інформації з тексту, який потрібно підсумувати на основі представлень. Існує два основних типи представлень:

- **Topic representations - Репрезентація тем**

Фокусується на представленні тем, представлених у текстах. Існує кілька видів підходів до отримання цього представлення. Нижче буде опис двох з них. Інші включають семантичний аналіз і моделі Байєса.

Frequency Driven Approaches (Підходи, орієнтовані на частоту): у цьому підході ми призначаємо ваги словам. Якщо слово пов'язане з темою, ми присвоюємо 1 або 0. Ваги можуть бути безперервними залежно від реалізації. **Word Probability (**Імовірність слова): використовується частота слів як показник важливості слова. Імовірність слова w визначається частотою появи слова f (w), поділеною на всі слова у вхідних даних, які містять N слів.

$$P(w) = \frac{f(w)}{N}$$

Рисунок 1.2.3 – Word Probability

- **Indicator Representations:**

Індекатори уявлення - тип уявлень, який залежить від особливостей речень і ранжує їх на основі ознак. Тут важливість речення залежить не від слів, які воно містить, а безпосередньо від особливостей речення. Існує два способи представлення цього типу:

1. Методи на основі графіків
2. Методи машинного навчання

1. Методи на основі графіків

Методи на основі графів були вперше представлені в статті Рада Міхалчеа та Пола Тарау з Університету Північного Техасу. Цей метод називається алгоритмом Text Rank, і на нього впливає алгоритм Page Rank від Google. Цей алгоритм насамперед намагається знайти важливість вершини в даному йому графі.

В цьому алгоритмі кожне речення представлено у вигляді вершини. Ребро, що з'єднує дві вершини або два речення, означає, що два речення подібні. Якщо подібність будь-яких двох речень перевищує певний поріг, вузли, що представляють речення, з'єднуються ребром.

Коли дві вершини з'єднані, це означає, що одна вершина віддає голос іншій. Чим більше кількість голосів за певний вузол (вершину чи речення), тим важливішим є цей вузол і, очевидно, представлене речення. Тепер голоси також певним чином зважені, кожен голос не має однакової ваги чи важливості. Важливість голосування також залежить від важливості вузла або речення, що голосує, чим вище важливість вузла, який голосує, тим вище важливість голосу. Отже, кількість голосів, поданих за речення, і важливість цих голосів визначають важливість речення. Це та сама ідея, що лежить в основі алгоритму рейтингу сторінок Google і того, як він вирішує та ранжує веб-сторінки, лише вузли представляють веб-сторінки.

Якщо у нас є абзац, ми розкладемо його на набір речень. Тепер, скажімо, ми представляємо кожне речення як вершину «vi», отже, ми отримуємо набір вершин V. Як обговорювалося, ребро з'єднує вершину з іншою вершиною того самого набору, тому ребро E можна представити як підмножину (V x V). У випадку орієнтованого графа, скажімо, In(V{i}) — це кількість вхідних ребер до вузла, а Out(v{j}) — це кількість вихідних ребер із даного вузла, а

також показник важливості вершина задана S{j}.

RageRank Alghoritm

Відповідно до алгоритму рейтингу сторінок Google:

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

Рисунок 1.2.4 – Google Rage Rank Algoritm

Де S(V{i}) є оцінкою досліджуваного вузла, а S(V(j)) представляє всі вузли, які мають вихідні ребра до V{i}. Тепер оцінка V{j} ділиться на ступінь відхилення від V{j}, що враховує ймовірність того, що користувач вибере саме цю веб-сторінку.

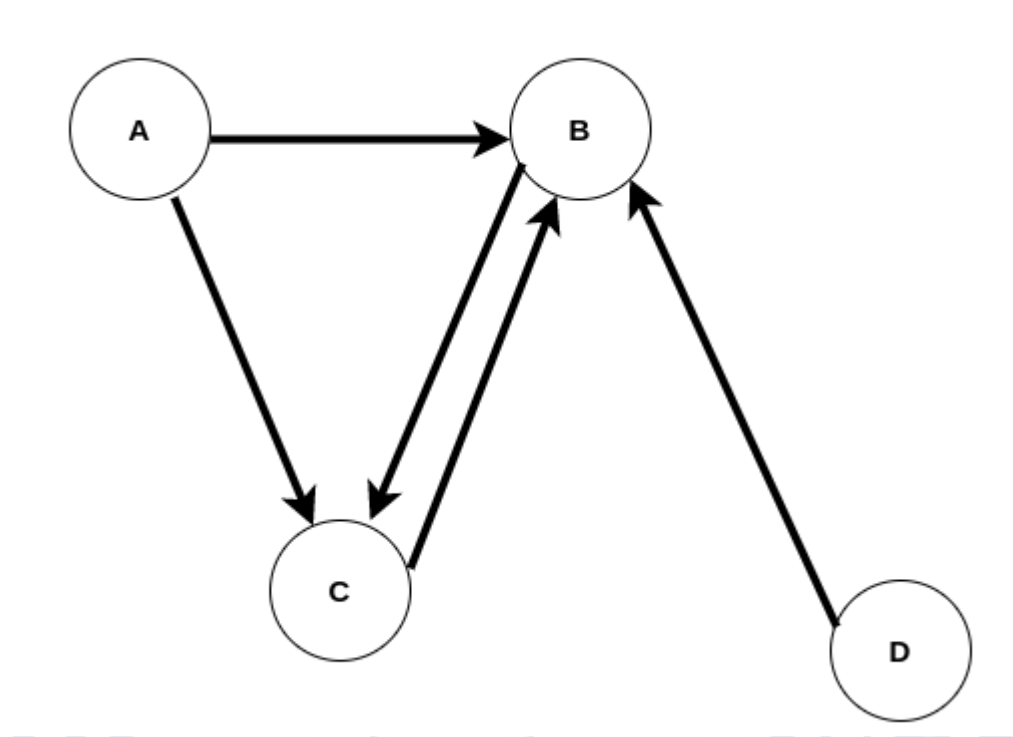


Рисунок 1.2.5 – Google Rage Rank Graphic

Точніше кажучи, якщо це графік, який стоїть на A, як користувач, я можу перейти як до B, так і до C, тому ймовірність того, що я перейду до C, дорівнює ½, тобто 1/(відступ від A). Фактор d називається коефіцієнтом

демпфування. В оригінальному алгоритмі рейтингу сторінок фактор d включає випадковість. 1-d означає, що користувач перейде на випадкову веб-сторінку, а не на підключені. Коефіцієнт зазвичай становить 0,85. Цей же алгоритм реалізовано в алгоритмі Text Rank.

Виникає питання, як отримуються бали?

Як ми бачимо вище, є 4 вершини, спочатку ми призначаємо випадкові оцінки всім вершинам, скажімо [0,8,0,9,0,9,0,9]. Потім ребрам призначаються оцінки ймовірності.

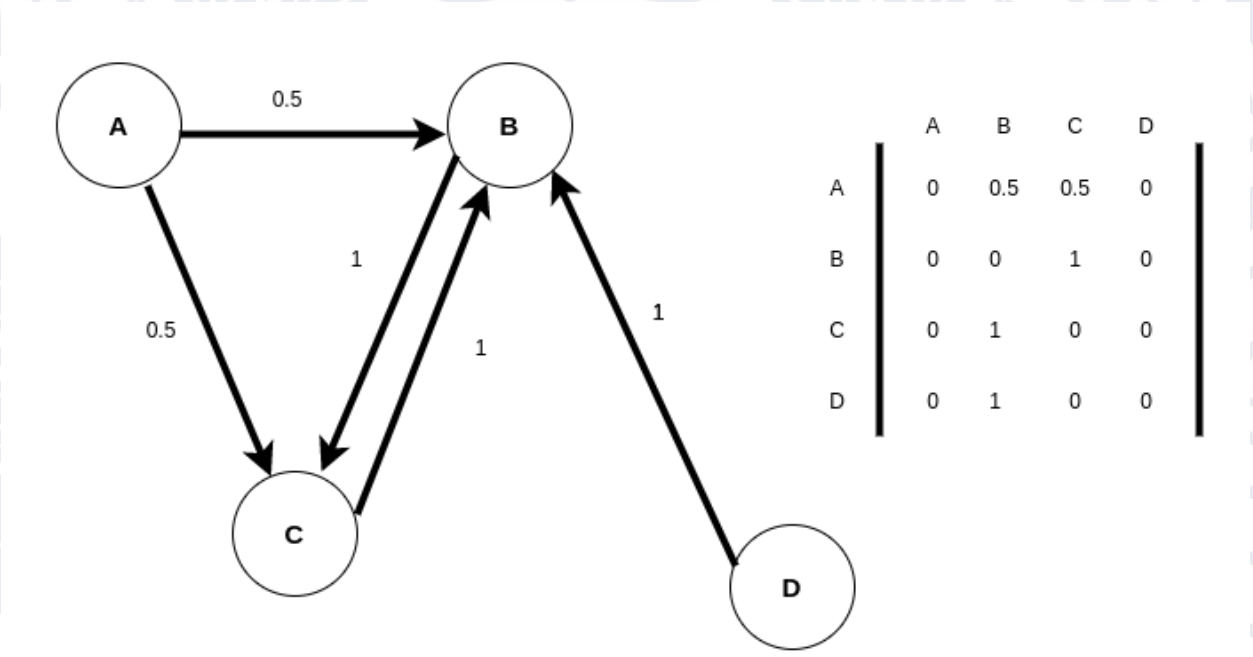


Рисунок 1.2.6 – Google Rage Rank графік та матриця

Матриця — це суміжна матриця графа. Можна помітити, що значення суміжних матриць є значеннями ймовірності, тобто 1/відступ від цього вузла або вершини. Отже, насправді графік рейтингу сторінок стає незваженим, оскільки рівняння містить лише термін, який дає вагу.

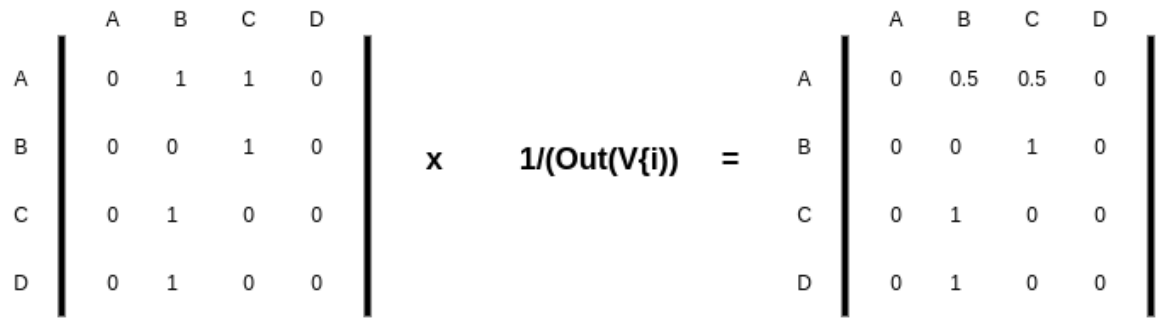


Рисунок 1.2.7 - Google Rage Rank матриця

Тепер усе рівняння виглядає так:

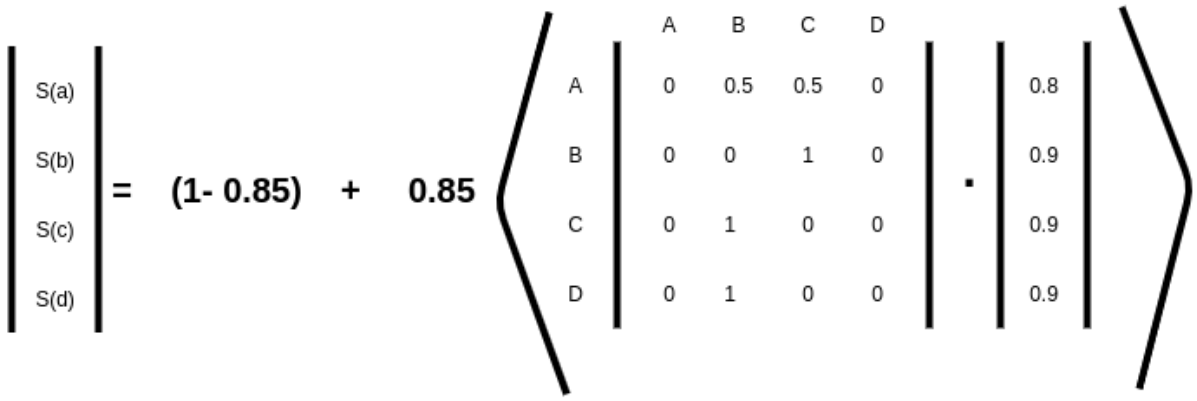


Рисунок 1.2.8 - Google Rage Rank

Abstraction-based Summarization

Абстрактні методи

Абстрактивні реферування називаються так тому, що вони не вибирають речення з початково поданого фрагмента тексту для створення реферування. Замість цього вони перефразують основний зміст даного тексту, використовуючи набір лексики, відмінний від оригінального документа. Підсумовуючи, це дуже схоже на те, що ми робимо як люди. Ми створюємо семантичне представлення документа в нашому мозку. Потім ми вибираємо слова з нашого загального словника (слів, які ми зазвичай використовуємо), які відповідають семантиці, щоб створити короткий підсумок, який представляє всі пункти фактичного документа. Розробка такого типу

підсумків може бути складною, оскільки буде необхідний Generation Natural Language. Розглянемо найбільш використовуваний підхід до вирішення проблеми.

Застосування RNN послідовності до послідовності

Цей підхід було запропоновано в статті Рамеша Наллапаті, Боуена Чжоу, Ціцерона дос Сантоса, Каглара Гулчехре, Бінг Сяна з IBM. Термін «моделі послідовності» використовується тому, що моделі створені для створення вихідної послідовності слів із вхідної послідовності слів. Вхідною послідовністю в розглянутому випадку є власне текстовий документ, а вихідною – скорочене резюме.

У статті пропонується модель, натхненна рекурентною моделлю кодера-декодера нейронної мережі, яка вперше була запропонована для машинного перекладу Дмитром Багданау, Університет Якоба, Німеччина.

Проблеми дуже різні. По-перше, для машинного перекладу нам потрібно, щоб переклад був без втрат, оскільки нам потрібно точне речення в перекладеній формі, але для генерації резюме нам потрібно стиснути оригінальний документ, щоб створити резюме, тому воно має бути трохи з втратами. По-друге, для створення резюме довжина резюме не залежить від оригінального тексту. Ці два пункти є ключовими викликами в проблемі, як задано проблемою.

Розглянемо мережі кодера та декодера та причину використання рівня уваги.

Виділення ключових слів на основі навчання з учителем

Для створення анотації у разі навчання з учителем застосовуються такі підходи машинного навчання:

– логічний (використовуються дерева рішень, наприклад, алгоритми ID 3, C4.5, CART) для бінарної класифікації;

- імовірнісний (наприклад, спрощений (наївний) алгоритм Байєса, логістична регресія, алгоритм максимальної ентропії) для бінарної класифікації;

- нейромережевий (наприклад, машина опорних векторів, багатошаровий персептрон) для бінарної класифікації ;

– метаевристичний (наприклад генетичний алгоритм) для ідентифікації значень параметрів умов евристичних правил.

Далі розглядаються найбільш популярні алгоритми та системи.

Формування безлічі основ словосполучень-кандидатів

Перед виділенням ключових слів відбувається формування багатьох основ словосполучень-кандидатів, що складається з трьох етапів.

Перший етап. Очищення вхідного тексту

Знаки пунктуації, дужки та номери замінюються межами речень; видаляються апострофи, дефіси та інші знаки, що не належать до літер.

Другий етап. Вибір словосполучень-кандидатів

Словосполучення-кандидати вибираються виходячи з таких міркувань:

– словосполучення-кандидати обмежуються деякою максимальною довжиною (зазвичай складаються не більше, ніж із трьох слів);

– словосполучення-кандидати не можуть бути власними іменами;

- словосполучення-кандидати не можуть починатися або закінчуватися стоп-словом;

– словосполучення-кандидати не можуть зустрічатися у статті лише один раз.

Список стоп-слів містить 425 слова з дев'яти синтаксичних класів (союзи,

артиклі, частки, прийменники, займенники, дієслова, прикметники, прислівники).

Наприклад, на основі речення «*the programming by demonstration method»* будуть згенеровані словосполучення - кандидати "*programming"* , "*demonstration", "method", "programming by demonstration", "demonstration method", "programming by demonstration method"* , оскільки "*the"* та "*by"* є стоп-словами.

Третій етап. Виділення основ словосполучень-кандидатів

Виділення ключових слів на основі алгоритму C4.5

1. Задається множина документів (статей) $A = \{a_i\}$ потужністю n із класифікованими словосполученнями-кандидатами (клас 1 – словосполучення є ключовим словом, клас 0 – словосполучення не є ключовим словом). Задається конкретна стаття a_0.

2. Створюється множина $T = \{t_j\}$ потужністю q, що складається з основ словосполучень-кандидатів статті a_0, у три етапи.

3. Розраховуються ознаки для кожної j-ї основи словосполучення-кандидата:

– кількість основ слів, з яких складається j-я основа словосполучення-кандидата;

- перше входження j-ї основи словосполучення-кандидата, що складається з основи одного слова, до статті a_0 (обчислюється у вигляді відношення позиції першого входження словосполучення-кандидата у статтю a_0 до загальної кількості слів у статті a_0, причому до загальної кількості включаються стоп-слова);

– перше входження j-ї основи словосполучення-кандидата до статті a_0 (обчислюється як ставлення позиції першого входження словосполучення-

кандидата у статтю a_0 до загальної кількості слів у статті a_0, причому у загальну кількість включаються стоп-слова);

- частота появи j-й основи словосполучення-кандидата що складається з основи одного слова, у статті a_0 (обчислюється у вигляді відношення кількості появ j-го словосполучення-кандидата у статті a_0 до загальної кількості слів у статті a_0, причому в загальну кількість включаються стоп-слова);

- частота появи j-й основи словосполучення-кандидата у статті a_0 (обчислюється у вигляді відношення кількості появ j-й основи словосполучення-кандидата у статті a_0 до загальної кількості слів у статті a_0 , причому в загальну кількість включаються стоп-слова);

– відносна довжина j-го словосполучення-кандидата (відповідає j-й основі) у статті a_0 (обчислюється у вигляді відношення кількості символів у j-му словосполученні-кандидаті у статті a_0 до середньої кількості символів у словосполученні-кандидаті).

3.3. Виконується дискретизація значень ознак

Для кожної j-ї основи словосполучення-кандидата значення ознак відносять до відповідних діапазонів значень, при цьому для кожного діапазону значень визначено на основі навчальних даних ймовірність попадання в нього значення ознаки ключового слова .

4. Класифікація кожної основи j-го слова-кандидата статті a_0 (клас 1 – словосполучення є ключовим словом, клас 0 – словосполучення не є ключовим словом) на основі алгоритму С4.5.

Виділення ключових слів на основі спрощеного (наївного) алгоритму Байєса (алгоритм KEA)

1. Задається множина документів (статей) $A = \{a_i\}$ потужністю n із класифікованими словосполученнями-кандидатами (клас 1 – словосполучення є ключовим словом, клас 0 – словосполучення не є ключовим словом). Задається конкретна стаття a_0.

2. Створюється множина $T = \{t_j\}$ потужністю q, що складається з основ словосполучень-кандидатів статті a_0, у три етапи.

3. Розраховуються дві ознаки для основ словосполучень-кандидатів

3.1. Обчислюється вага TF*IDF кожної j-й основи словосполучення-кандидата у статті a_0 у вигляді

$$TFIDF(t_j, a_0) = TF(t_j, a_0) IDF(t_j),$$

де вага TF (time frequency) j-й основи словосполучення-кандидата у статті a_0 найчастіше обчислюється як кількість появ j-й основи словосполучення-кандидата у статті a_0.

вага IDF (inverse document frequency) кожного j-й основи словосполучення-кандидата у статті найчастіше обчислюється у вигляді

$$IDF(t_j) = \begin{cases} \ln \dfrac{1}{(1/n) + P(t_j)}, & P(t_j) > 0 \\ \ln n, & P(t_j) = 0 \end{cases},$$

$$P(t_j) = \frac{|\{a_z \in A \,|\, t_j \in a_z\}|}{n},$$

де $P(t_j)$ – ймовірність появи j-ї основи словосполучення-кандидата у статті, що обчислюється як відношення кількості статей з j-ї основою словосполучення-кандидата до загальної кількості статей.

3.2. Обчислюється перше входження кожної j-ї основи

словосполучення-кандидата до статті a_0 $FO(t_j, a_0)$ у вигляді відношення позиції першого входження основи словосполучення-кандидата у статтю a_0 до загальної кількості слів у статті a_0.

3.3. Виконується дискретизація значень обох ознак

Для кожної j-ї основи словосполучення-кандидата значення ознак $TFIDF(t_j, a_0)$ і $FO(t_j, a_0)$ відносять до відповідних діапазонів значень, при цьому для кожного діапазону значень визначена на основі навчальних даних можливість попадання в нього значення ознаки ключового слова. Тому $TFIDF(t_j, a_0)$, $FO(t_j, a_0)$ замінюються $P(TFIDF(t_j, a_0)|1), P(FO(t_j, a_0)|1)$ відповідно, де 1 – номер класу, відповідний ключовому слову.

4. Класифікація кожної j-ї основи словосполучення-кандидата статті a_0 (клас 1 – словосполучення-кандидат є ключовим словом, клас 0 – словосполучення-кандидат не є ключовим словом) на основі спрощеного (наївного) алгоритму Байєса у вигляді

$$P(1|\mathbf{x}_j) = \frac{P(\mathbf{x}_j|1)P(1)}{P(\mathbf{x}_j|1)P(1) + (1 - P(\mathbf{x}_j|1))(1 - P(1))},$$

$$P(0|\mathbf{x}_j) = \frac{(1 - P(\mathbf{x}_j|1))(1 - P(1))}{P(\mathbf{x}_j|1)P(1) + (1 - P(\mathbf{x}_j|1))(1 - P(1))},$$

$$P(\mathbf{x}_j|1) = P(TFIDF(t_j, a_0)|1)P(FO(t_j, a_0)|1),$$

де $P(1)$ – ймовірність появи словосполучення, що входить у анотацію. $P(1)$ визначена на основі навчальних даних

Виділення ключових слів на основі машини опорних векторів

1. Задається множина документів (статей) $A = \{a_i\}$ потужністю n із класифікованими словосполученнями-кандидатами (клас 1 – словосполучення

є ключовим словом, клас 0 – словосполучення не є ключовим словом). Задається конкретна стаття a_0.

2. Створюється множина $T = \{t_j\}$ потужністю q, що складається з основ словосполучень-кандидатів статті a_0, у три етапи.

3. Розраховуються ознаки для основ словосполучень-кандидатів

3.1. Обчислюється вага TF (time frequency) кожної j-ї основи словосполучення-кандидата у статті a_0. Вага TF найчастіше обчислюється як кількість появ j-й основи словосполучення-кандидата у статті a_0.

3.2. Обчислюється вага IDF (inverse document frequency) кожної j-ї основи словосполучення-кандидата у статті. Вага IDF найчастіше обчислюється у вигляді

$$IDF(t_j) = \begin{cases} \ln \dfrac{1}{(1/n) + P(t_j)}, & P(t_j) > 0 \\ \ln n, & P(t_j) = 0 \end{cases},$$

$$P(t_j) = \frac{|\{a_z \in A \,|\, t_j \in a_z\}|}{n},$$

де $P(t_j)$ – ймовірність появи j-ї основи словосполучення-кандидата у статті, що обчислюється як відношення кількості статей з j-ї основою словосполучення-кандидата до загальної кількості статей.

3.3. Обчислюється перше входження кожної j-ї основи словосполучення-кандидата до статті a_0 $FO(t_j, a_0)$ у вигляді відношення позиції першого входження словосполучення-кандидата у статтю a_0 до загальної кількості слів у статті a_0.

3.4 Формування вектора ознак для кожної j-й основи словосполучення-кандидата у вигляді

$$\mathbf{x}_j = (TF(t_j, a_0), IDF(t_j), FO(t_j, a_0)).$$

4. Бінарна класифікація кожної j-й основи словосполучення-кандидата статті a_0 на основі машини опорних векторів.

5. Якість класифікації оцінюється на основі *F-міри подібності (міри на основі гармонійного середнього)* у вигляді

$$F = \frac{2RP}{R+P} \to \min,$$

де P – відношення ключових слів, виділених вручну та автоматично, до кількості ключових слів виділених автоматично,

R – відношення ключових слів, виділених вручну та автоматично, до кількості ключових слів, виділених вручну.

Виділення ключових слів на основі генетичного алгоритму (алгоритм GenEx)

1. Задається множина документів (статей) $A = \{a_i\}$ потужністю n із класифікованими словосполученнями-кандидатами (клас 1 – словосполучення є ключовим словом, клас 0 – словосполучення не є ключовим словом). Задається конкретна стаття a_0.

2. Створюється множина $T = \{t_j\}$ потужністю q, що складається з основ словосполучень-кандидатів статті a_0, у три етапи. У цьому довжина основи кожного слова зі словосполучення має перевищувати SL, $SL \notin [1,10]$.

3. Розраховуються ознаки для кожного j-го словосполучення-кандидата:

– позиція першого входження основи j-го словосполучення-кандидата, що складається з одного слова, у статтю a_0 $FO1(t_j, a_0)$;

– позиція перше входження основи j-го словосполучення-кандидата у статтю a_0 $FO2(t_j, a_0)$;

- частота появи основи першого слова j-го словосполучення-кандидата, що складається з одного слова, у статтю a_0 $TF1(t_j, a_0)$ (обчислюється у вигляді відношення кількості появ j-го словосполучення-кандидата у статті a_0 до загальної кількості слів у статті a_0, причому до загальної кількості включаються стоп-слова);

- частота появи основи j-го словосполучення-кандидата в статті a_0 $TF2(t_j, a_0)$ (обчислюється у вигляді відношення кількості появ j-го словосполучення-кандидата у статті a_0 до загальної кількості слів у статті a_0 , причому до загальної кількості включаються стоп-слова);

- відносна довжина основи j-го словосполучення-кандидата в статті a_0 (обчислюється у вигляді відношення кількості символів в основі j-м словосполучення-кандидата у статті a_0 до загальної кількості символів у статті a_0).

4. Обчислення оцінки основи кожного j-го словосполучення-кандидата, що складається з одного слова, у статті a_0 у вигляді

$$\text{cost}(t_j, a_0) = \begin{cases} TF1(t_j, a_0) \cdot FLF1, & TF1(t_j, a_0) < FLT \\ TF1(t_j, a_0), & FLT \le TF1(t_j, a_0) \le FHT \\ TF1(t_j, a_0) \cdot FHF1, & TF1(t_j, a_0) > FHT \end{cases},$$

де $FLT \in [1,1000]$, $FHT \in [1,4000]$, $FLF1 \in [1,15]$, $FHF1 \in [0.01,1]$.

5. Упорядкування основ словосполучень-кандидатів, які з одного слова, вибір їх NW перших, $NW \notin [25,75]$, і занесення в список словосполучень-кандидатів.

6. Обчислення оцінки основи кожного j-го словосполучення-кандидата, що складається з двох слів, у статті a_0 у вигляді

$$\text{cost}(t_j, a_0) = \begin{cases} TF1(t_j, a_0) \cdot FLF1 \cdot F2O, & TF1(t_j, a_0) < FLT \\ TF1(t_j, a_0), & FLT \le TF1(t_j, a_0) \le FHT, \\ TF1(t_j, a_0) \cdot FLF1 \cdot F2O, & TF1(t_j, a_0) > FHT \end{cases}$$

де $F2O \in [1,3]$.

Обчислення оцінки основи кожного j-го словосполучення-кандидата, що складається з трьох слів, у статті a_0 у вигляді

$$\text{cost}(t_j, a_0) = \begin{cases} TF1(t_j, a_0) \cdot FLF1 \cdot F3O, & TF1(t_j, a_0) < FLT \\ TF1(t_j, a_0), & FLT \le TF1(t_j, a_0) \le FHT, \\ TF1(t_j, a_0) \cdot FLF1 \cdot F3O, & TF1(t_j, a_0) > FHT \end{cases}$$

де $F3O \in [1,5]$.

7. Для основи кожного j-го словосполучення-кандидата зі списку словосполучень-кандидатів, сформованого на кроці 5, знайти найкращу за вартістю основу словосполучення-кандидата, що складається з двох слів, і найкращу за вартістю основу словосполучення-кандидата, що складається з трьох слів, і додати їх до списку словосполучень-кандидатів . Після додавання здійснюється впорядкування списку.

8. Якщо у списку словосполучень-кандидатів є три основи словосполучення-кандидата, що складаються з одного слова, що входять до складу однієї і тієї ж основи словосполучення-кандидата, що складається з трьох слів, то в списку залишається лише одне з них найкраще за вартістю. Якщо у списку словосполучень-кандидатів є дві основи словосполучення-кандидата, що складаються з одного слова, що входять до складу однієї й тієї ж основи словосполучення-кандидата, що складається з двох слів, залишається лише одне з них краще за вартістю.

9. Для основи кожного j-го словосполучення-кандидата зі списку словосполучень-кандидатів, знаходяться всі словосполучення, що містять цю основу, і з них вибирається найчастіше словосполучення . Це словосполучення

замінює у списку основу.

10. Якщо у статті a_0 словосполучення-кандидат зустрічається у різних регістрах, то в список словосполучень-кандидатів слід помістити всі регістри.

11. Обчислюється відносна довжина j-го словосполучення-кандидата (відповідна j-й основі) у статті a_0 $RL(t_j, a_0)$ (обчислюється у вигляді відношення кількості символів у j-му словосполученні-кандидаті у статті a_0 до середньої кількості символів у словосполученні-кандидаті).

12. Якщо $RL(t_j, a_0) < MLLR \wedge \text{cost}(t_j, a_0) \le MRLL$, то j-е словосполучення-кандидат видаляється зі списку, $MLLR \in [0.,3], MRLL \in [1,20]$.

13. У списку словосполучень-кандидатів залишають лише перших NP словосполучень-кандидатів, $NP \notin [5,15]$.

Генетичний алгоритм використовується для налаштування параметрів SL, FLT, FHT, $FLF1$, $F2O$, $F3O$, FHF, $MLLR$, $MRLL$, які розглядаються як гени хромосоми. Обсяг популяції – 50, кількість ітерацій – 1050.

Як фітнес-функція виступає F-міра (міра на основі гармонійного середнього) у вигляді

$$F = \frac{2RP}{R+P} \to \min,$$

де P – відношення ключових слів, виділених вручну та автоматично, до кількості ключових слів виділених автоматично,

R – відношення ключових слів, виділених вручну та автоматично, до кількості ключових слів, виділених вручну.

Створення анотації на основі навчання без вчителя

Для створення анотації у разі навчання без вчителя найчастіше застосовуються такі підходи:

– графовий (наприклад, алгоритми TextRank, LexRank, виділення абзаців);

– тест відношення правдоподібності;

– метричний (наприклад, міра подібності MMR);

- алгебраїчний (використовується факторизація матриць за допомогою алгоритмів PCA, SVD);

– семантичний (на основі лексичних ланцюгів та WordNet);

- поверхневий (наприклад, системи MEAD, LAKHAS, Едмундсона та Луна, алгоритм SumBasic).

Далі розглядаються найбільш популярні алгоритми та системи.

Створення анотації на основі тесту відношення правдоподібності (система SUMMARIST)

Застосовується для багатодокументного реферування.

1. Задаються кластери $A_k = \{a_{ki}\}$, $k \in \{1,...,c\}$, що містять n_k статей.

2. Для кожного k-го кластера створюється множина $T_k = \{t_{kj}\}$ потужністю q_k, що складається з центральних термінів статей k-го кластера (зазвичай з кожної статті береться перших 50-200 слів).

3. Для кожного k-го кластера речення статті, що входять до нього, об'єднуються в множину $S_k = \{s_{kl}\}$ потужністю L_k.

4. Для кожного k-го кластера обчислюється вага кожного j-го терміна у вигляді

$$w_{kj} = \begin{cases} 1, & -2\ln\lambda_{kj} > 10 \\ 0, & -2\ln\lambda_{kj} \le 10 \end{cases},$$

$$\lambda_{kj} = \frac{L(H_1)}{L(H_2)} = \frac{b(O_{kj11}; O_{kj11} + O_{kj12}, p_{kj})b(O_{kj21}; O_{kj21} + O_{kj22}, p_{kj})}{b(O_{kj11}; O_{kj11} + O_{kj12}, p_{kj1})b(O_{kj21}; O_{kj21} + O_{kj22}, p_{kj2})},$$

$$b(k; n, p) = \binom{n}{k} p^k (1-p)^{(n-k)},$$

де $b(k; n, p)$ - біномний розподіл,

H_1 – гіпотеза, що два розподіли мають однаковий параметр p_{kj} (релевантність статті не залежить від j-го терміну),

H_2 - гіпотеза, що два розподіли мають різні параметри p_{kj1} та p_{kj2} (релевантність статті залежить від j-го терміну),

$L(H_1), L(H_2)$ - функції правдоподібності для гіпотез H_1, H_2.

Тоді

$$-2\ln\lambda_{kj} = -2((O_{kj11} + O_{kj21})\ln p_{kj} + (O_{kj12} + O_{kj22})\ln(1-p_{kj}) -$$

$$-(O_{kj11}\ln p_{kj1} + O_{kj12}\ln(1-p_{kj1}) + O_{kj21}\ln p_{kj2} + O_{kj22}\ln(1-p_{kj2})))$$

,

$$p_{kj1} = \frac{O_{kj11}}{O_{kj11} + O_{kj12}},\ p_{kj2} = \frac{O_{kj21}}{O_{kj21} + O_{kj22}},$$

$$p_{kj} = \frac{O_{kj11} + O_{kj21}}{O_{kj11} + O_{kj12} + O_{kj21} + O_{kj22}},$$

де O_{kj11} – кількість входження j-го терміну до статей k-го кластера,

O_{kj12} – кількість входження j-го терміну до статей усіх кластерів, крім k-го кластера,

O_{kj21} - кількість входження всіх термінів, крім j-го терміну, у статті k-го кластера,

O_{kj22} – кількість входження всіх термінів, крім j-го терміну, у статті всіх кластерів, крім k-го кластера,

$O_{kj11} + O_{kj12} + O_{kj21} + O_{kj22}$– кількість входження всіх термінів до статей усіх кластерів.

Якщо $w_{kj} = 0$, $q_k = q_k - 1$то $T_k = T_k \setminus \{t_{kj}\}$.

5. Для кожного k-го кластера обчислюється оцінка (ранг) кожного l речення з кожної i статті k-го кластера у вигляді

$$Score(s_{kil}) = \sum_{t_{kj} \in s_{kil}} \frac{w_{kj}}{\left|\{t_{kj} \mid t_{kj} \in s_{kil}\}\right|},$$

5. Для кожного k-го кластера речення статті, що входять до нього, об'єднуються в множину $S_k = \{s_{kl}\}$ потужністю L_k та впорядковуються за оцінками (рангам).

6. Для кожного k-го кластера з ранжованої множини в k-ю анотацію вибираються перші (найкращі) речення (зазвичай 20%)

Створення анотації на основі міри подібності MMR

Застосовується для багатодокументного реферування.

1. Задаються кластери $A_k = \{a_{ki}\}$, $k \in \{1,...,c\}$, що містять n_k статей.

2. Для кожного k-го кластера створюється множина $T_k = \{t_{kj}\}$ потужністю q_k, що складається з центральних термінів статей k-го кластера (зазвичай з кожної статті береться перших 50-200 слів).

3. Для кожного k-го кластера вага IDF (inverse document frequency) кожного j-го терміна обчислюється найчастіше у вигляді

$$IDF(t_{kj}) = \begin{cases} \ln \dfrac{1}{(1/n_k) + P(t_{kj})}, & P(t_{kj}) > 0 \\ \ln n_k, & P(t_{kj}) = 0 \end{cases},$$

$$P(t_{kj}) = \frac{|\{a_{kz} \in A_k \mid t_{kj} \in a_{kz}\}|}{n_k},$$

де $P(t_{kj})$ – ймовірність появи j-го терміну у статті k-го кластера, що обчислюється як відношення кількості статей з j-м терміном до загальної кількості статей.

Якщо $IDF(t_{kj}) < \alpha$ (зазвичай α=3), то $T_k = T_k \setminus \{t_{kj}\}$, $q_k = q_k - 1$.

4. Для кожного k-го кластера обчислюється вага TF*IDF кожного j-го терміна l-го речення i-й статті у вигляді

$$TFIDF(t_{kj}, s_{kil}) = TF(t_{kj}, s_{kil}) IDF(t_{kj}),$$

де вага TF (time frequency) j-го терміну в l-му реченні k-го кластера найчастіше обчислюється як кількість появ j-го терміну в l-му реченні k-го кластера.

5. Для кожного k-го кластера речення статей, що входять до нього, об'єднуються в множину $S_k = \{s_{kl}\}$ потужністю L_k.

6. Для кожного k-го кластера речення обчислюється центроїд

$$\mathbf{m}_k = \frac{1}{L_k} \sum_{l=1}^{L_k} \mathbf{x}_{kl},$$

$$\mathbf{x}_{kl} = (TFIDF(t_{k1}, s_{kl}), ..., TFIDF(t_{kj}, s_{kl}), ..., TFIDF(t_{kq_k}, s_{kl})).$$

7. В k-ю анотацію вибирається речення l^* на основі косинусної міри подібності

$$l^* = \arg\min_l \frac{\mathbf{x}_{kl} \cdot \mathbf{m}_k}{\|\mathbf{x}_{kl}\| \cdot \|\mathbf{m}_k\|}, \; l \in \{1, ..., L_k\},$$

$$\tilde{S}_k = \{s_{kl^*}\}, \; \tilde{L}_k = 1, \; S_k = S_k \setminus \{s_{kl^*}\}, \; L_k = L_k - 1.$$

8. В k-ю анотацію обирається чергова речення l^* на основі максимальної маргінальної релевантності (MMR) міри подібності

$$l^* = \arg\max_l \left(\lambda \frac{\mathbf{x}_{kl} \cdot \mathbf{m}_k}{\|\mathbf{x}_{kl}\| \cdot \|\mathbf{m}_k\|} - (1-\lambda) \max_u \frac{\mathbf{x}_{kl} \cdot \tilde{\mathbf{x}}_{ku}}{\|\mathbf{x}_{kl}\| \cdot \|\tilde{\mathbf{x}}_{ku}\|} \right),$$

$$\tilde{S}_k = \tilde{S}_k \cup \{s_{kl^*}\}, \; \tilde{L}_k = \tilde{L}_k + 1, \; S_k = S_k \setminus \{s_{kl^*}\}, \; L_k = L_k - 1,$$

$$\lambda = \begin{cases} 0.3, & 1 \le \tilde{L}_k < \Lambda / 3 \\ 0.5, & \Lambda / 3 \le \tilde{L}_k < 2\Lambda / 3, \text{ де } \Lambda \text{ зазвичай 20\% } L_k, \\ 0.7, & 2\Lambda / 3 \le \tilde{L}_k \le \Lambda \end{cases}$$

$u \in \{1,...,\tilde{L}_k\}$, $l \in \{1,...,L_k\}$.

9. Якщо $\tilde{L}_k < \Lambda$, то перехід 8.

Мережі кодера та декодера

Якщо розглядати загальний рівень LSTM (довготермінової короткострокової пам'яті), він виглядає приблизно так, як наведена нижче схема(Рисунок 1.2.9). Він або створює вихідні дані для кожного входу, або створює вектор ознак, який пізніше використовується шарами щільної нейронної мережі для завдань класифікації із застосуванням шарів softmax. Наприклад, виявлення настрою, де ми пропускаємо все речення через RNN і використовуємо вектори ознак, які відповідають шарам softmax для отримання кінцевого результату.

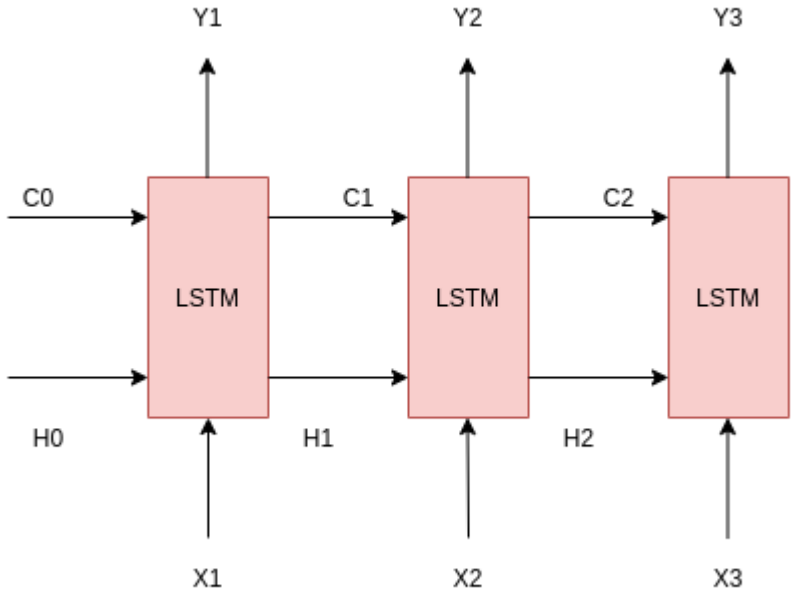


Рисунок 1.2.9 - LSTM

Але тут слід усвідомити одну річ: для поточної проблеми або проблем, подібних до цієї, включаючи машинний переклад, коли ми можемо сказати, що проблема є проблемою послідовності до послідовності, цей конкретний модельний підхід не можна застосувати. Основна причина полягає в тому, що розмір виведення не залежить від розміру вхідного, і обидва є послідовністю. Для вирішення цієї проблеми була представлена модель мережі кодер-декодер.

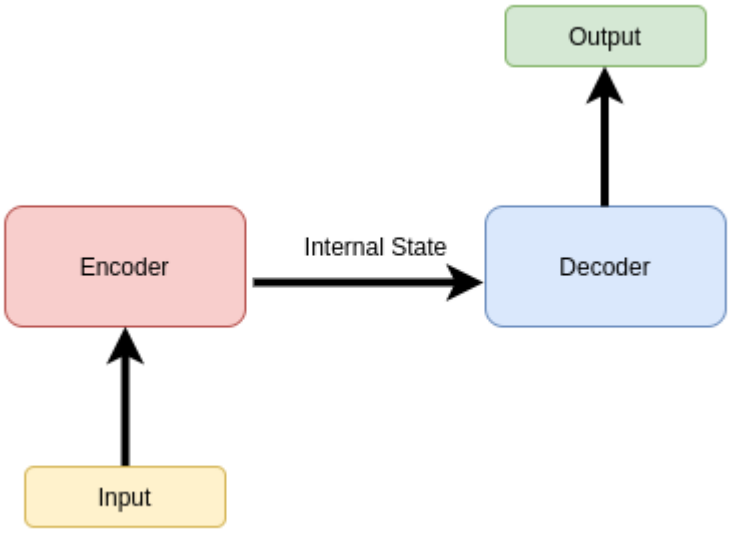


Рисунок 1.2.1.1 - Модель мережі кодер-декодер

Кодер відповідає за введення вхідного речення або оригінального документа та генерування кінцевого вектора стану (прихований стан і стан клітинки). Це представлено внутрішнім станом на діаграмі. Кодер може містити шари LSTM, RNN або GRU. Здебільшого використовуються шари LSTM через усунення проблеми розгортання та зникнення градієнта.

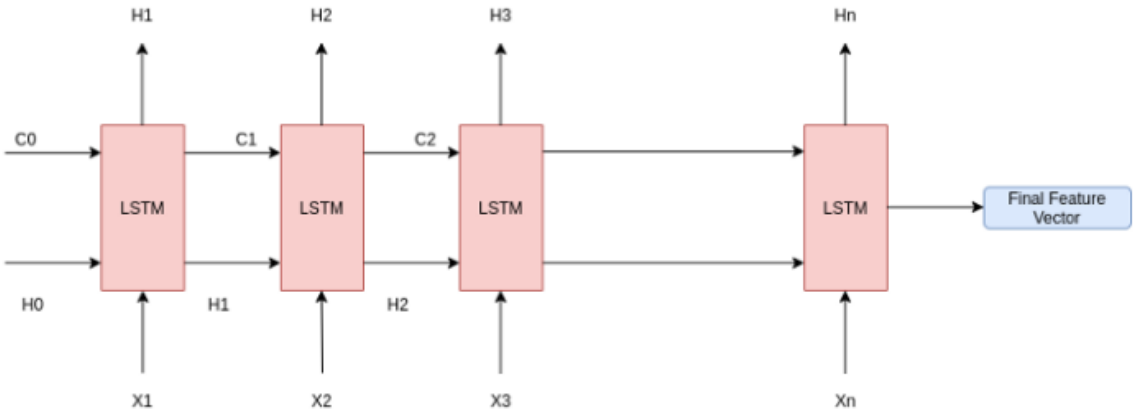


Рисунок 1.2.1.2

На діаграмі вище показано нашу мережу кодувальника. У мережі кодувальника одне слово подається за крок у часі, і, нарешті, після того, як n-те вхідне слово подається на рівень LSTM, прихований стан і стани комірки стають нашим кінцевим станом або вектором ознак. Стан комірки Cn і прихований стан Hn надсилаються до першого набору рівня LSTM декодера.

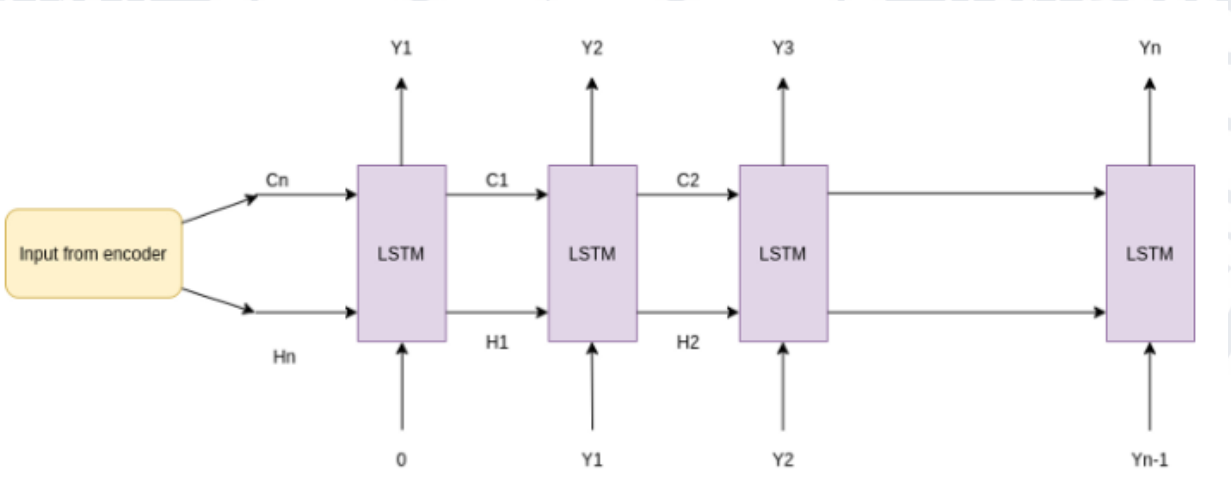


Рисунок 1.2.1.3

Ось так виглядає модель декодера. Тепер перший рівень отримує вхідні дані від кінцевих станів кодера, які є активаціями прихованих та клітинних станів. Модель декодера приймає вхідні дані та генерує передбачувані слова вихідної послідовності, враховуючи згенероване попереднє слово. Таким

чином, для LSTM на етапі часу 1 для декодера має 0 векторних вхідних даних, і Y1 є згенерованим прогнозованим словом, для кроку часу 2 Y1 подається на рівень LSTM як вхідний сигнал, а Y2 є згенерованим словом і так далі. Декодер генерує слова крок за кроком, доки не буде зустрічатися тег <end>.

Як генеруються слова? Модель кодера-декодера навчається на цільовому наборі або словниковому запасі слів. Тепер на кожному кроці декодера прихована активація LSTM надсилається через рівень softmax, який генерує ймовірності для кожного слова у словнику, яке буде передбачено як наступне слово. Слово з максимальною ймовірністю вибирається як результат на цьому кроці часу. Тепер, як модель дізнається, яке слово точно відповідає семантиці? Для цього модель тренується на наборі даних і перетворює проблему на контрольовану проблему класифікації. Крім того, моделі зазвичай використовують вбудовування слів у словник із добре відомих векторів вбудовування, таких як word2vec від Google або Glove від Standford NLP. Вбудовування слів допомагає отримати кілька уявлень про слово, наприклад, чи схоже дане слово на дане слово чи ні. Іноді векторизація TFIDF також використовується для створення значущості контекстних слів.

1.3 Аналіз сайтів аналогів туристичних агенцій

На сьогоднішній день існує безліч систем, що надають послуги з бронювання житлового фонду, при вивченні аналогів розроблюваної системи. Для постановки задачі на першому етапі роботи необхідно розглянути декілька аналогів для аналізу їх від’ємностей і недоліків. Це допоможе більш чітко сформувати основні вимоги до об’єкту розробки.

Першим аналогом буде розглянуто сайт туристичної компанії «Компас+» (Рисунок 1)

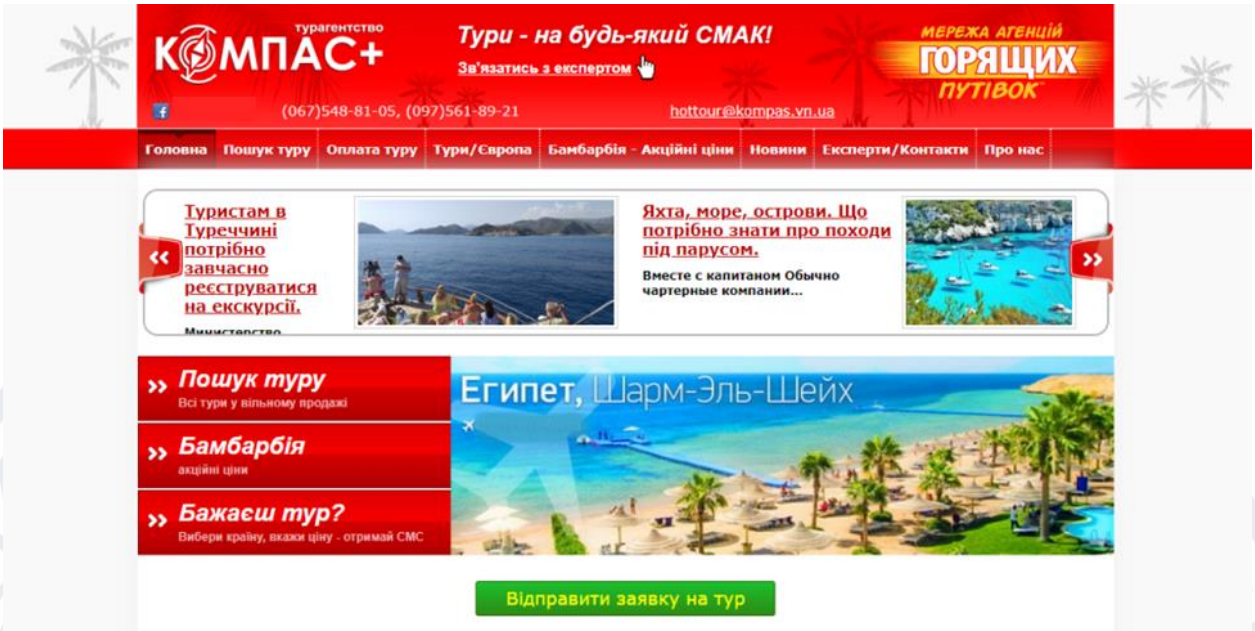


Рисунок 1 – Скріншот Головної сторінки сайту «Компас+»

Ця компаніє предоставляє для користувача більш менш зрозумілий інтерфейс, кнопки розташовані за тематикою, що є дуже вдалою ідеєю. На даному сайті є меню, у якого є пункти, які дозволяють побачити тури, пошук туру, акційні ціни, додаткову інформацію про країни, в які будуть виконуватися подорожі, тощо. Також на сайті присутні посилання на соціальні мережі, де можна побачити чітко і якісно тури які пропунує дана компанія, відгуки про них. Наявна форма підписки на новини за допомогою електронної пошти, а також вибравши країну та суму турист розраховує, команда експертів підбере місце відпочинку. На сайті реалізована функція пошуку турів за певними параметрами.

До переваг даного веб-сайту можна віднести наступне:

- онлайн-бронювання напрямків;
- наявність контактних телефонів;
- наявність соціальних мереж;
- пошук турів, включно з «гарячими пропозиціями»(акційними;

– наявність підписки за допомогою електронної пошти.

До недоліків необхідно віднести такі пункти:

– До недоліків слід віднести статичність сайту, відсутність мобільної версії, яка на даний час є дуже актуальною серед майбутніх мандрівників;

– відсутність ще декількох груп у соціальних мережах, окрім Facebook, потрібно мати хоча б ще 2 інші соціальні мережі, наприклад Instagram, Twitter.

– Доволі багато непотрібного тексту та мало фотографій турів, оскільки користувачі в першу чергу звертають увагу саме на фото.

Другим аналогом буде розглянуто сайт туристичної компанії «Янко» (Рисунок 2)

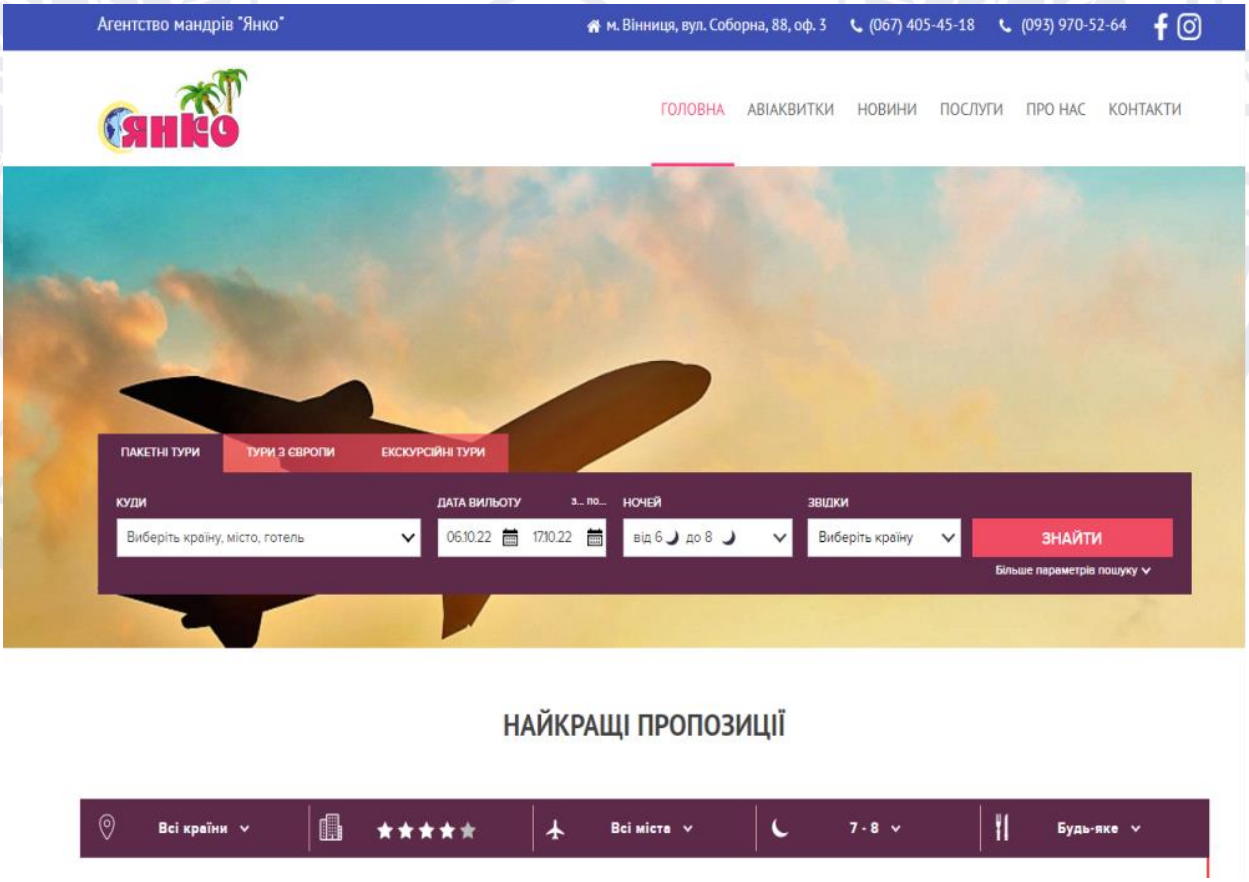


Рисунок 2 – Скріншот Головної сторінки сайту «Янко»

На даному сайті цієї туристичної компанії одразу в очі Пошук туру, в я кому присутній вибір міста, дати та кількість днів планування відпочинку, та вибір звідки саме буде відбуватись виліт. А також присутні акційни тури та тури з Європи.

В меню присутні такі розділи, як авіаквитки, новини, послуги компанії, інформація про неї, місцезнаходження та самі контакти для звязку. Також одною з переваг є присутність соціальних мереж, а також приємний інтерфейс.

Третім аналогом буде розглянуто сайт туристичної компанії «Dominika Travel» (Рисунок 3)

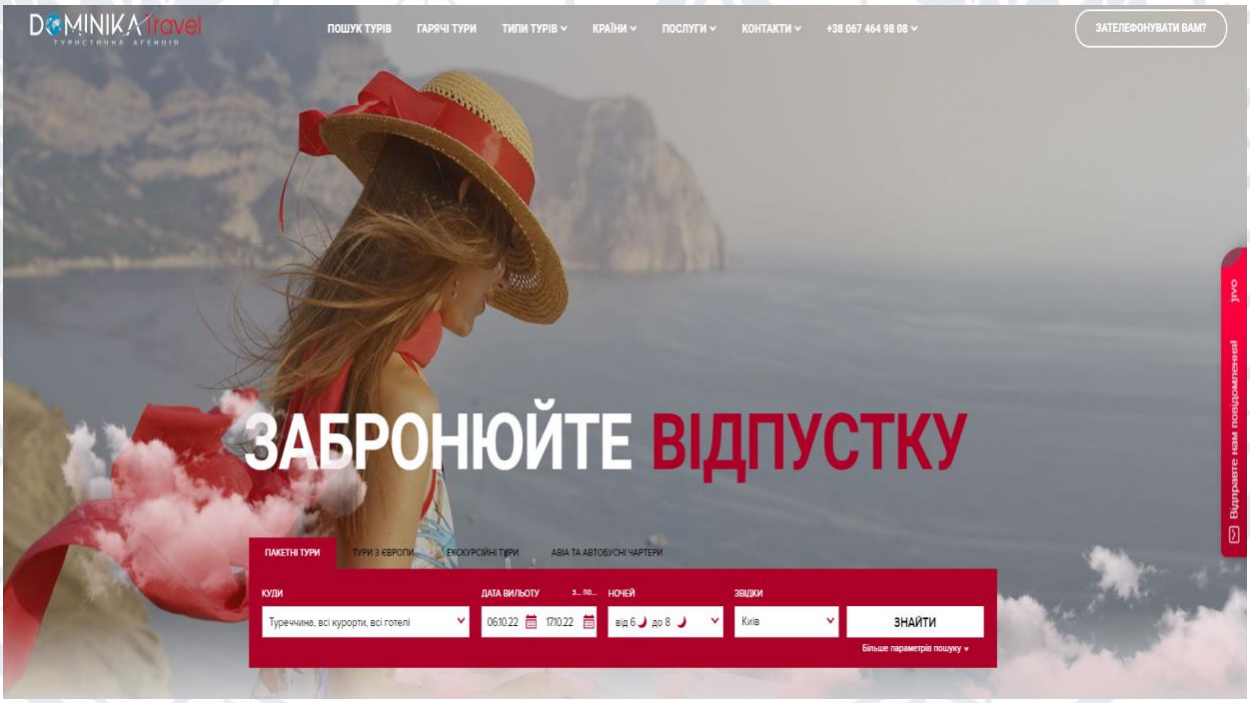


Рисунок 3 – Скріншот Головної сторінки сайту «Dominica Travel»

Одразу після переходу на сторінку цієї туристичної агенції, помітно інтуїтивно зрозумілий та приємний для перегляду сайт.

Переваги даного веб-сайту:

- Наявність додаткової інформації (блог, фотогалерея тощо);

- Пошук туру;
- Спеціальні акційні пропозиції, гарячі тури;
- Можливість залишити заявку, щоб вам зателефонували, для більш детальної інформації;
- Можливість продивитись тури в вибраній країні;
- Наявність груп, у соціальних мережах;
- Контактна інформація тур оператора;
- Елемент «календар», по якому легко орієнтуватися у датах гарячих турів;

1.4 Постановка задачі

- розгляд методів та систем дослідження контенту сайтів;
- проектування системи дослідження контенту сайту туристичного агенства на підставі методів Data Science;
- розляд аналогів веб-сайтів туристичних агенств
- моделювання сайту UML-засобами
- розробка сайту за допомогою програмних інстументів JavaScript, React тощо.
- розгляд методів автоматичного реферування тексту
- розроблення програмного інструментарію побудови системи дослідження контенту сайту туристичного агенства на підставі методів Data Science.

Висновки до розділу 1

В першому розділу ми розібрали актуальність вебсайту для туристичної агенції, та його переваги. Також детально розглянули методи Екстраційні методи реферування та Абкстрактні методи реферування тексту. Було розглянуто аналоги продукту, який буде розроблятись, а також їх плюси та мінуси. Поставили завдання для проектування та розробки інтерфейсу веб-ресурсу.

РОЗДІЛ 2. ПРОЕКТУВАННЯ ОБ'ЄКТУ РОЗРОБКИ

2.1 Опис предметної області

Сайт туристичної компанії має бути зручним та зрозумілим для будь-якого користувача, мати добре спроектовану навігацію, доцільну інформацію у зручному вигляді, оскільки до туристичних компаній звертаються саме ті люди, які не хочуть бути заклопотаними організацією власного відпочину, тому сайт має бути максимально простим та легким для сприйняття.

Вдало спроектований інтерфейс сайту «приємний для ока» та викликає довіру у користувача, інакше складається думка що туристична компанія вже не працює або вона зовсім не дбає про свою репутацію. Ще одною важливою річчю для туристичної компанії є можливість інформувати своїх клієнтів поза сайтом, адже користувачам незручно кожен день заходити на сторінку компанії для отримання новин та про цікаві і вигідні пропозиції. Тому на сайті має бути спосіб поширення новин поза сайтом. Одним з таких способів є поштова розсилка.

Але останнім часом більшість користувачів частіше читають стрічку новин у соціальних мережах ніж перевіряють пошту на наявність нових листів. Тому ведення групи у «Facebook» або «Instagram» є необхідною умовою для інформування клієнтів туристичної компанії.

Метою є розробка сайту для туристичної компанії.

З методів Data Science буде використовуватись Автоматичне реферування тексту за допомогою Lexrank Algorithm.

2.2 Моделювання UML-засобами

UML, скорочення від Unified Modeling Language, — це стандартизована мова моделювання, що складається з інтегрованого набору діаграм, розроблених для допомоги розробникам систем і програмного забезпечення у визначенні, візуалізації, конструюванні та документуванні артефактів програмних систем, а також для бізнес-моделювання та інші непрограмні

системи. UML представляє збірку найкращих інженерних практик, які виявилися успішними в моделюванні великих і складних систем. UML є дуже важливою частиною розробки об'єктно-орієнтованого програмного забезпечення та процесу розробки програмного забезпечення. UML використовує переважно графічні нотації для вираження дизайну проектів програмного забезпечення. Використання UML допомагає командам проектів спілкуватися, досліджувати потенційні проекти та перевіряти архітектурний дизайн програмного забезпечення.

Метою UML є надання стандартної нотації, яка може використовуватися всіма об'єктно-орієнтованими методами, а також вибір та інтеграція найкращих елементів нотацій-попередників. UML був розроблений для широкого діапазону програм. Таким чином, він забезпечує конструкції для широкого діапазону систем і дій (наприклад, розподілених систем, аналізу, проектування та розгортання системи).

Оскільки стратегічна цінність програмного забезпечення зростає для багатьох компаній, галузь шукає методи автоматизації виробництва програмного забезпечення та покращення якості та скорочення вартості та часу виходу на ринок. Ці методи включають технологію компонентів, візуальне програмування, шаблони та фреймворки. Підприємства також шукають методи управління складністю систем у міру того, як вони збільшуються в масштабі та масштабі. Зокрема, вони визнають необхідність вирішення повторюваних архітектурних проблем, таких як фізичний розподіл, паралелізм, реплікація, безпека, балансування навантаження та відмовостійкість. Крім того, розробка Всесвітньої павутини, спрощуючи деякі речі, загострила ці архітектурні проблеми. Уніфікована мова моделювання (UML) була розроблена для відповіді на ці потреби. Пейдж-Джонс у своїй книзі «Фундаментальний об'єктно-орієнтований дизайн в UML» узагальнює основні цілі розробки UML так:

- Надання користувачам готову до використання виразну

мову візуального моделювання, щоб вони могли розробляти значущі моделі та обмінюватися ними.

- Забезпечення механізмами розширення та спеціалізації для розширення основних концепцій.
- Бути незалежним від конкретних мов програмування та процесів розробки.
- Забезпечення формальноъ основи для розуміння мови моделювання.
- Заохочування зростання ринку орієнтованих інструментів.
- Підтримка концепцій розробки вищого рівня, таких як співпраця, фреймворки, шаблони та компоненти.
- Інтегрування найкращих практик.
-

У мові UML є кілька стандартних видів відношень між акторами і варіантами використання:

– асоціації (association relationship);

– включення (include relationship);

– розширення (extend relationship);

– узагальнення (generalization relationship).

Відношення асоціації – одне з фундаментальних понять у мові UML і в тій чи іншій мірі використовується при побудові всіх графічних моделей систем у формі канонічних діаграм. Включення (include) у мові UML – це різновид відношення залежності між базовим варіантом використання і його спеціальним випадком. При цьому відношенням залежності (dependency) є таке відношення між двома елементами моделі, при якому зміна одного елемента (незалежного) приводить до зміни іншого елемента (залежного)

Для даної магістерської роботи роботи було створено діаграми прецедентів, які зображені на рисунках 2.2.1 – 2.2.2:

Рисунок 2.2.1 – Діаграма прецедентів для клієнтів

Рисунок 2.2.2 – Діаграма прецедентів для Турагенції

2.3 Логічна структура сайту

При розробці логічної структури сайту потрібно враховувати доступість важливих розділів для користувачів. Відвідувачі не повинні шукати необхідні розділи, вони (розділи) повинні бути завжди на виду. Користувач не повинен здійснювати більше двох-трьох кліків мишкою, щоб потрапити в потрібний розділ. Крім того, рівень вкладеності не глибше третього спростить індексацію сайту пошуковими системами. Google регулярно вдосконалює свої алгоритми і в пошуковій видачі представляє найрелевантніші сторінки (які максимально відповідають запиту користувача).

Структура сайту – це логічна побудова всіх сторінок ресурсу. Схема, за якою розподіляється шлях до папок, категоріям, підкатегоріями тощо. З технічної точки зору, навігація ресурсу являє собою набір URL, логічно вибудуваних в певній послідовності. Структура взаємопов'язана з семантичним ядром (ключовими словами сторінок). Тому, зібравши семантику, вже можна зробити начерки схеми побудови кожного майбутнього url.

Пошукові роботи проводять проходять по посиланнях сайту. І чим правильніше, простіше буде складена структура сайту, тим менше пошуковий робот витрачає свої ресурси і швидше проведе обхід ресурсу. Це призведе до найбільш швидкої індексації сайту.

Отже, структура безпосередньо впливає на ранжування. Чим вона краще, тим швидше відбудеться індексація. Це є невід'ємним фактором для SEO-просування. Пошукова система проводить аналогію між сайтом і ставленням користувачів до нього. Чим привабливіший ресурс для споживачів, тим він привабливіший і для пошукових систем. Якщо алгоритм Google бачить, що сайт має низький час перебування відвідувача на ньому, поганий показник коефіцієнта кліків, то він сприйме ресурс, як неякісний, що дуже погано позначиться на просуванні

і пошуковій видачі веб-сайту.

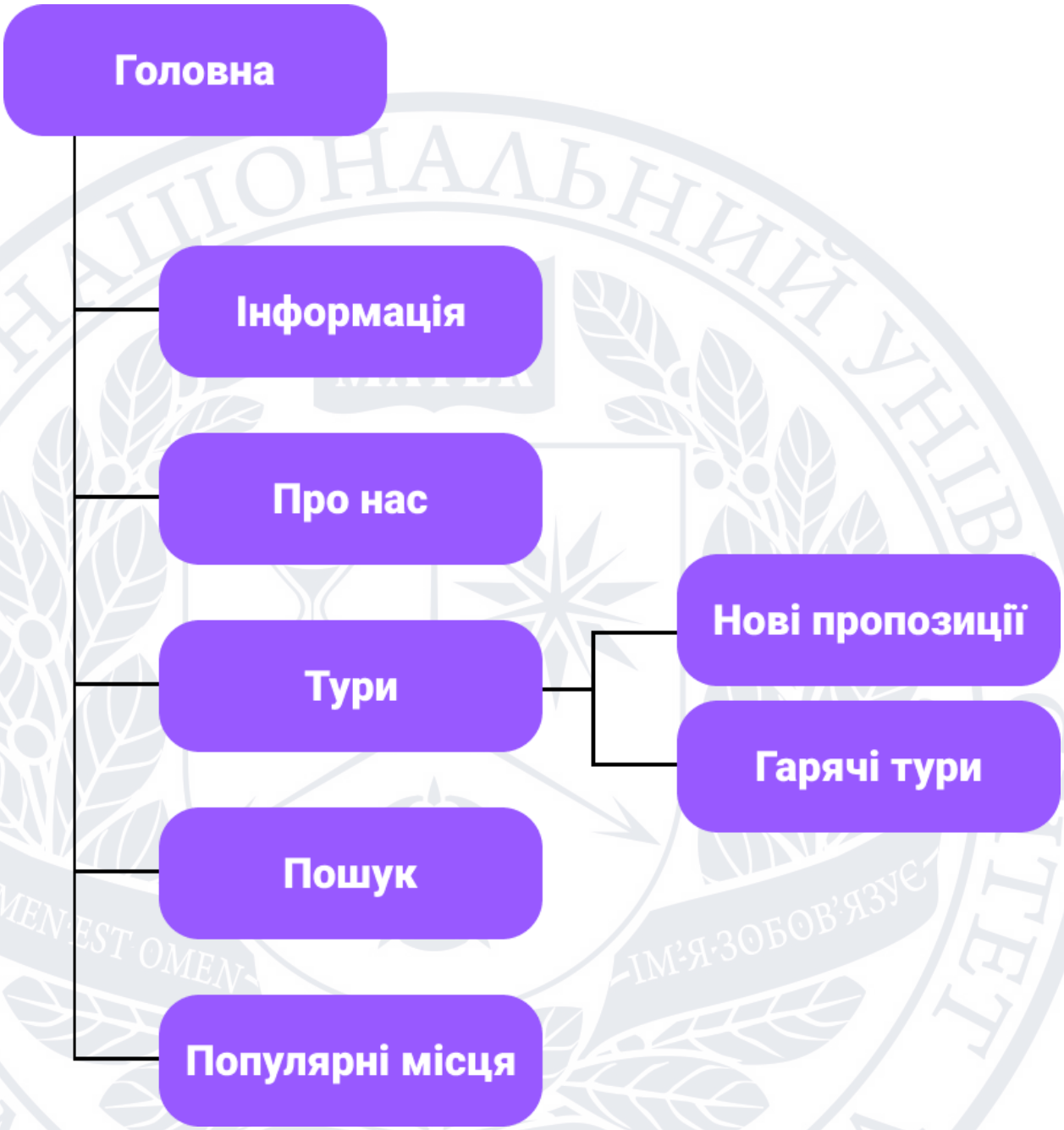


Рисунок 2.3.1 – Логічна структура сайту

2.4 Вибір метода автоматичного реферування тексту

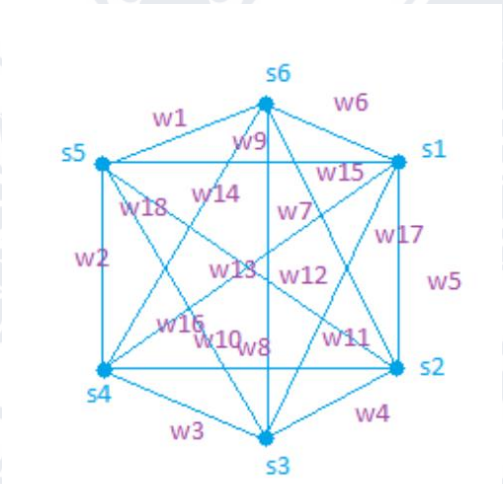
В даній роботі було обрано LexRank Alghoritm. Він використовує підхід на основі графіків для автоматичного підсумовування тексту. Розберемо, для прикладу, концепцію LexRank і різні методи її реалізації в Python.

LexRank — це неконтрольований підхід на основі графіків для автоматичного підсумовування тексту. Оцінка речень здійснюється графічним методом. LexRank використовується для обчислення важливості речення на основі концепції центральності власного вектора в представленні речень на графі.

У цій моделі ми маємо матрицю зв'язності, засновану на косинусній подібності внутрішнього речення, яка використовується як матриця суміжності представлення речень на графі. Це виділення речень здебільшого обертається навколо набору речень з однаковою метою, тобто вибирається центроїдне речення, яке працює як середнє для всіх інших речень у документі. Потім речення ранжуються за схожістю.

Графічний підхід:

- На основі центральності власного вектора.
- У вершинах графів розміщені речення
- Вага на ребрах обчислюється за допомогою косинусної метрики подібності



Де Si — речення у вершинах відповідно, а Wij — ваги на ребрах.

Рисунок 2.4.1 – Графічний підхід

Обчислення косинусної подібності:

Щоб визначити подібність, модель сумки слів використовується для представлення N-вимірних векторів, де N є кількістю всіх можливих слів у словах певної мови. Для кожного слова, яке зустрічається в реченні, значенням відповідного виміру у векторному представленні речення є кількість входжень слова в реченні, помножена на ідентифікатор слова.

$$\text{idf-modified-cosine}(x, y) = \frac{\sum_{w \in x,y} \text{tf}_{w,x} \text{tf}_{w,y} (\text{idf}_w)^2}{\sqrt{\sum_{x_i \in x} (\text{tf}_{x_i,x} \text{idf}_{x_i})^2} \times \sqrt{\sum_{y_i \in y} (\text{tf}_{y_i,y} \text{idf}_{y_i})^2}}$$

Де $\text{tf}_{w,s}$ — кількість входжень слова w у реченні s.

Рисунок 2.4.2 – Обчислення косинусної подібності

Центральність власного вектора та LexRank:

Внесок кожного вузла розраховується для визначення загальної центральності. Але не обов'язково, щоб у кожній графічній мережі всі вузли вважалися однаково важливими. Якщо є непов’язані документи з дуже важливими реченнями, ці речення автоматично отримають високі оцінки центральності. Цієї ситуації можна уникнути шляхом розгляду вузлів походження. Таким чином, простий метод обчислення центральності:

$$\mathbf{p} = \mathbf{B}^{\mathbf{T}}\mathbf{p}$$

$$\mathbf{p}^{\mathbf{T}}\mathbf{B} = \mathbf{p}^{\mathbf{T}}$$

Де матриця B — це матриця суміжності, отримана з графа подібності шляхом ділення кожного елемента на відповідну суму рядків. pT — лівий власний вектор матриці B з відповідним власним значенням 1.

Алгоритм:

Алгоритм розрахунку балів LexRank такий:

```
1.MInputAn array S of n sentences, cosine threshold t output: An array L of LexRank scores
2.Array CosineMatrix[n][n];
3.Array Degree[n];
4.Array L[n];
5.for i ← 1 to n do
6.     for j ← 1 to n do
7.          CosineMatrix[i][j] = idf-modified-cosine(S[i],S[j]);
8.          if CosineMatrix[i][j] > t then
9.               CosineMatrix[i][j] = 1;
10.               Degree[i] + +;
11.          end
12.          else
13.               CosineMatrix[i][j] = 0;
14.          end
15.     end
16.end
17.for i ← 1 to n do
18.     for j ← 1 to n do
19.          CosineMatrix[i][j] = CosineMatrix[i][j]/Degree[i];
20.     end
21.end
22. L = PowerMethod(CosineMatrix,n,ϱ);
23. return L;
```

У перших 4 етапах ми ініціалізуємо необхідні масиви для речень, порогового значення, балів LexRank, матриці косинусів, ступеня та власних значень відповідно.

Етапи з 5 по 9 ініціалізують нашу матрицю модифікованими значеннями tf-idf для заданих речень. Якщо значення перевищують порогове значення idf, тоді елемент замінюється на значення 1. Якщо воно менше необхідного порогу, значення замінюється на 0. Таким чином ми створюємо типову таблицю або матрицю tf-idf на цих кроках.

На етапах 17-21 кожне значення матриці косинусів ділиться на ступінь кожного вузла. Ступінь центральності тут - це відповідний ступінь кожного вузла.

За допомогою методу Power iteration ми обчислюємо остаточну оцінку.

Power Iteration method:

Степенева ітерація або метод степеня в основному використовується для обчислення значення найбільшого власного вектора матриці. Для матриці

A, яку можна діагоналізувати, алгоритм створить число лямбда, яке є найбільшим (за абсолютною величиною) власним значенням A, і ненульовий вектор v, який є відповідним власним вектором лямбда, тобто Av=lambda v.

Метод має рекурентний зв'язок:

$$b_{k+1} = \frac{Ab_k}{\|Ab_k\|}$$

Де b_0 є наближенням домінантного власного вектора. Таким чином, на кожній ітерації вектор b множиться на матрицю A і нормалізується.

Відмінності між LexRank і TextRank

TextRank:

- Використовує типовий підхід PageRank
- Використовується для підсумовування одного документа

LexRank:

- Використовуйте для підсумовування кількох документів
- Враховує позицію та довжину речень
- Окрім підходу pageRank, він використовує показники подібності

Висновки до розділу 2

В даному розділу було зроблено опис предметної області. Було зроблено моделювання UML-засобами, опис самого URL. Описано алгоритм автоматичного реферування тексту, на прикладі мові Python, який буде використаний в даній роботі а саме LexRank Algoritm, та його порівняння з TextRank Algoritm. Також було розроблена логічна структура сайту.

РОЗДІЛ 3. ПРОГРАМНА РЕАЛІЗАЦІЯ ВЕБ-СТОРІНКИ ТУРИСТИЧНОГО АГЕНСТВА

3.1 Опис середовища та засобів реалізації

Реалізація даного продукта була за допомогою таких засобів:

JavaScript - динамічна, об'єктно-орієнтована мова програмування. Реалізація стандарту ECMAScript. Найчастіше використовується як частина браузера, що надає можливість коду на стороні клієнта (такому, що виконується на пристрої кінцевого користувача) взаємодіяти з користувачем, керувати браузером, асинхронно обмінюватися даними з сервером, змінювати структуру та зовнішній вигляд web-сторінки.

Мова JavaScript також використовується для програмування на стороні серверу (подібно до таких мов програмування, як Java і C#), розробки ігор, стаціонарних та мобільних додатків, сценаріїв в прикладному ПЗ (наприклад, в програмах зі складу Adobe Creative Suite), всередині PDF-документів тощо.

JavaScript класифікують як прототипну (підмножина об'єктно-орієнтованої), скриптову мову програмування з динамічною типізацією. Окрім прототипної, JavaScript також частково підтримує інші парадигми програмування (імперативну та частково функціональну) і деякі відповідні архітектурні властивості, зокрема: динамічна та слабка типізація, автоматичне керування пам'- яттю, прототипне наслідування, функції як об'єкти першого класу

HTML - Мова розмітки гіпертексту, або HTML, є стандартною мовою розмітки для документів, призначених для відображення у веб-браузері. Цьому можуть допомогти такі технології, як каскадні таблиці стилів (CSS) і мови сценаріїв, такі як JavaScript.

Веб-браузери отримують HTML-документи з веб-сервера або з

локального сховища та передають документи на мультимедійні веб-сторінки. HTML семантично описує структуру веб-сторінки та початково містить підказки для зовнішнього вигляду документа.

CSS (каскадна або блочна верстка) прийшла на заміну табличній верстці web-сторінок. Головна перевага блочної верстки – розділення змісту сторінки (даних) та їхньої візуальної презентації.

SASS – (англ. Syntactically Awesome Stylesheets – Синтаксично приголомшливі таблиці стилів) – скриптова метамова, яка інтерпретується в каскадні таблиці стилів (CSS). Спроектована Гемптоном Кетліном та розроблена Наталі Вейзенбаум.

SASS призначений для підвищення рівня абсракції коду та спрощення файлів CSS. Мова SASS має два синтаксиси:

- ✓ SASS (оригінальний) – відрізняється відсутністю фігурних дужок, в ньому вкладені елементи реалізовані за допомогою відступів, а правила відокремлюються переведенням рядка;
- ✓ SCSS (новий) – використовує фігурні дужки (подібно до CSS). Файли SASS-синтаксису мають розширення .sass, SCSS-синтаксису – .scss

SASS розширює CSS, надаючи кілька механізмів, доступних в більш традиційних мовах програмування, зокрема об'єктно-орієнтованих мовах, але недоступних для CSS. Інтерпретатор Sass транслює SassScript у блоки правил CSS.

React (також відомий як React.js або ReactJS) — це безкоштовна інтерфейсна бібліотека JavaScript із відкритим кодом[3] для створення інтерфейсів користувача на основі компонентів інтерфейсу користувача. Він підтримується Meta (раніше Facebook) і спільнотою окремих розробників і

компаній.

React можна використовувати як основу для розробки односторінкових, мобільних або серверних додатків із такими фреймворками, як Next.js. Однак React займається лише керуванням станом і відтворенням цього стану в DOM, тому створення додатків React зазвичай потребує використання додаткових бібліотек для маршрутизації, а також певної функціональності на стороні клієнта.

Node.js - це кросплатформне бекендове середовище виконання JavaScript з відкритим вихідним кодом, яке працює на JavaScript Engine (тобто V8) і виконує код JavaScript поза веб-браузером, розробленим для створення масштабованих мережевих програм. Node.js дозволяє розробникам використовувати JavaScript для написання інструментів командного рядка та для створення сценаріїв на стороні сервера — виконання сценаріїв на стороні сервера для створення динамічного вмісту веб-сторінки перед тим, як сторінка надсилається у веб-браузер користувача. Отже, Node.js представляє парадигму «JavaScript всюди», об’єднуючи розробку веб-додатків навколо однієї мови програмування, а не різних мов для серверних і клієнтських сценаріїв.

Node.js має керовану подіями архітектуру з можливістю асинхронного введення-виведення. Ці варіанти дизайну спрямовані на оптимізацію пропускної здатності та масштабованості у веб-додатках із багатьма операціями введення/виведення, а також для веб-додатків у реальному часі (наприклад, програм для спілкування в реальному часі та браузерних ігор).

Проект розподіленої розробки Node.js раніше керував Node.js Foundation, а тепер він об'єднався з JS Foundation, щоб сформувати OpenJS Foundation, якій сприяє програма Спільних проектів Linux Foundation.

MockAPI - це простий інструмент, який дозволяє легко створювати макети API, генерувати власні дані та попередньо виконувати над ними операції за допомогою інтерфейсу RESTful. MockAPI призначений для використання як інструмент для прототипування/тестування/навчання.

TypeScript — це безкоштовна мова програмування з відкритим вихідним кодом, розроблена та підтримується Microsoft. Це суворий синтаксичний набір JavaScript і додає необов'язкову статичну типізацію до мови. Він розроблений для розробки великих додатків і транспілюється на JavaScript. Оскільки це надмножина JavaScript, існуючі програми JavaScript також є дійсними програмами TypeScript.

TypeScript можна використовувати для розробки програм JavaScript як для виконання на стороні клієнта, так і на стороні сервера (як у випадку з Node.js або Deno). Для транспіляції доступні кілька варіантів. Можна використовувати типовий компілятор TypeScript або можна викликати компілятор Babel для перетворення TypeScript у JavaScript.

TypeScript підтримує файли визначення, які можуть містити інформацію про тип існуючих бібліотек JavaScript, подібно до того, як файли заголовків C++ можуть описувати структуру існуючих об'єктних файлів. Це дозволяє іншим програмам використовувати значення, визначені у файлах, як якщо б вони були статично типізованими сутностями TypeScript. Існують сторонні файли заголовків для популярних бібліотек, таких як jQuery, MongoDB і D3.js. Також доступні заголовки TypeScript для базових модулів Node.js, що дозволяє розробляти програми Node.js у TypeScript

3.2 Опис інтерфейсу програми

1. **Головна сторінка**

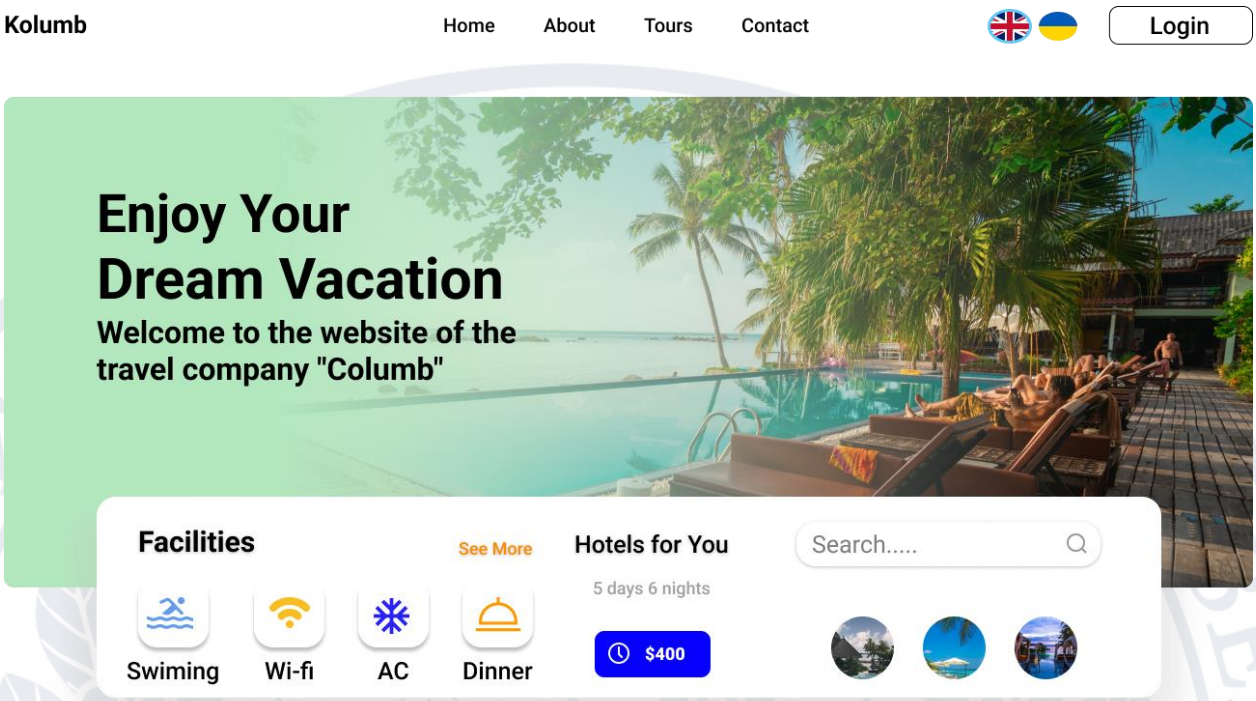


Рисунок 3.2.1 – Головна сторінка

Перше, що користувач побачить зайшовши на сторінку туристичної агенції, це Головну сторінку.

На ній можна скористуватись одразу навігацією сайту, а саме

Про нас – продивитись інформацією турагенства

Тури – перехід на сторінку турів, які пропонує туристична компанія

Контактну інформацію

Також присутній вибір мови, між англійською та українською

Присутня кнопка для авторизації користувача або адміністрації сайту

А, також на головній сторінці присутній пошук, де користувач одразу зможе ввести та знайти цікаву для нього пропозицію.

2. Пропозиція турів, по країні яку обрав користувач

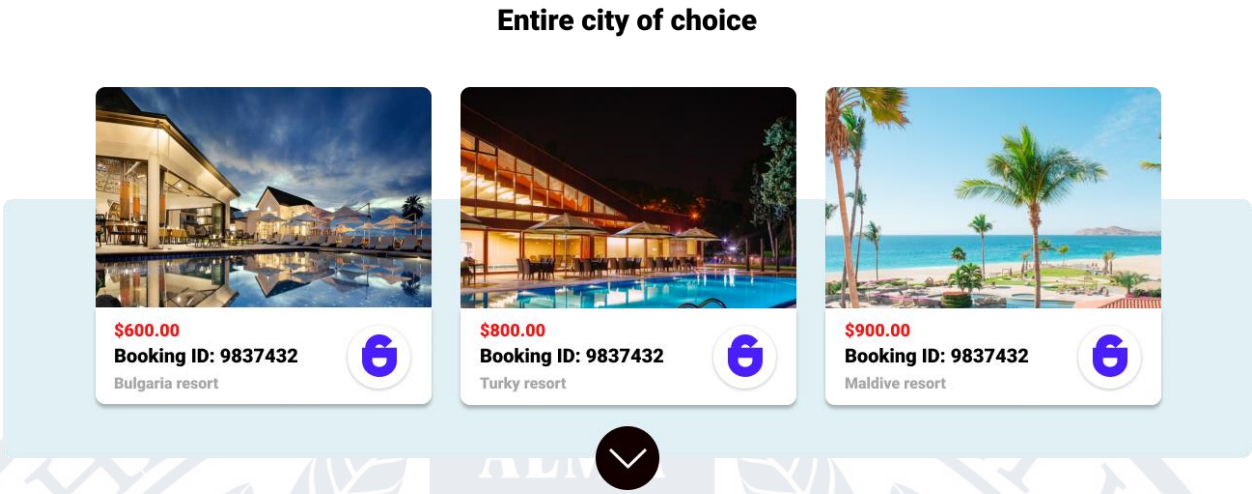


Рисунок 3.2.2 – Пропозиції турів, по країні (Головна сторінка)

Користувач обравши, наприклад, Турцію перейде на сторінку зі всіма курортами, які пропонує туроператор, в цій країні. Також можна побачити середню ціну турів, в даній країні.

3. Активність на відпочинку

Рисунок 3.2.3 – Активність на відпочинку (Головна сторінка)

В цьому пункті користувач може прочитати, та подивитись коротеньку інформацію/відео про сюрпризи, пляжи, активність тощо, на відпочинку.

4. Інформація від агенції

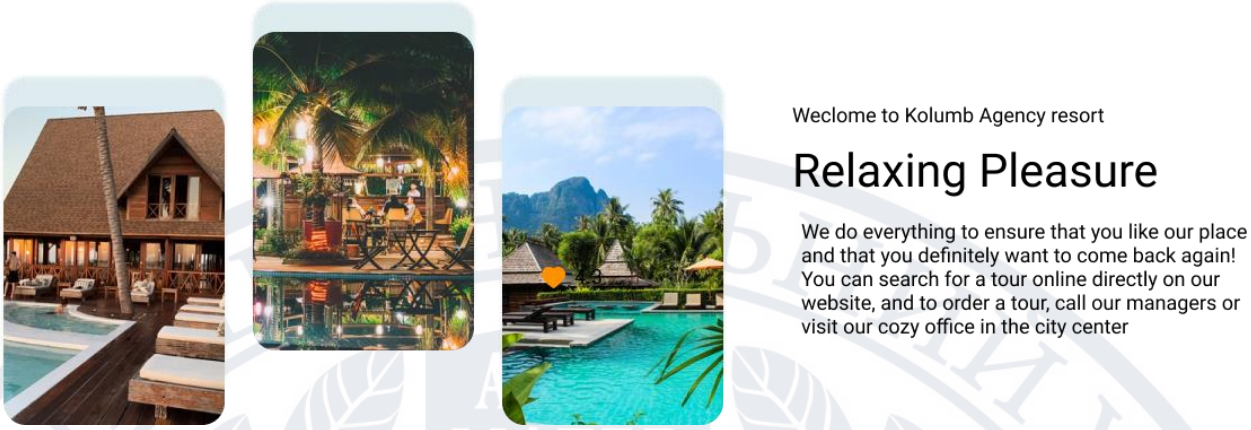


Рисунок 3.2.4 – Інформація від агенції (Головна сторінка)

5. Авіалінії з якими співпрацює туристичне агенство

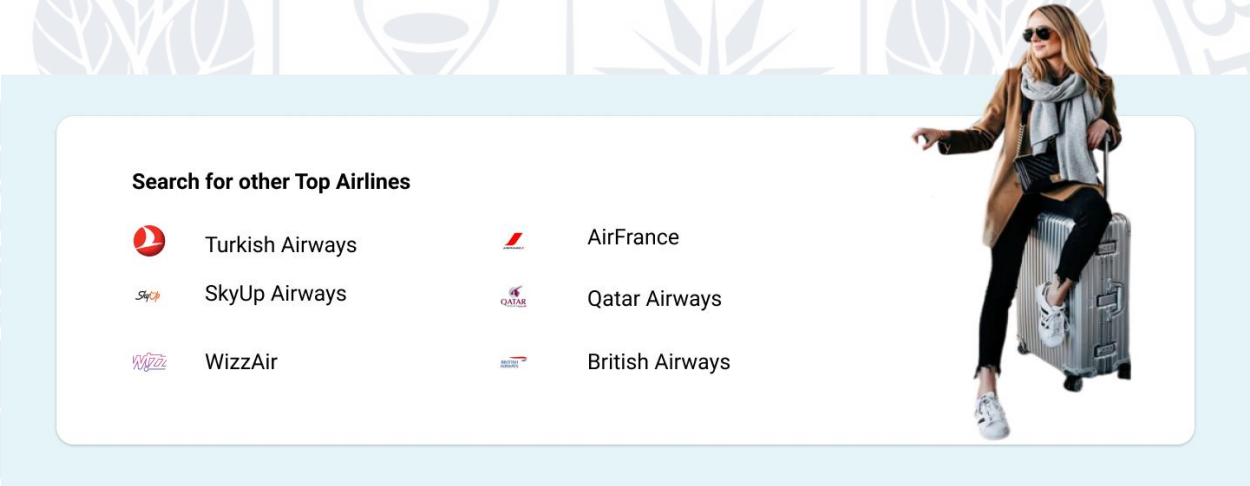


Рисунок 3.2.5 – Авіалінії з якими співпрацює туристичне агенство (Головна сторінка)

Для більшості людей важливо знати, з якими авіалініями співпрацює туристичне агенство. В основному для того щоб політ був комфортний. Тому було добавлений контейнер з вмістом цієї інформації.

6. Футер

Рисунок 3.2.6 – Футер (Головна сторінка)

7. Меню входу

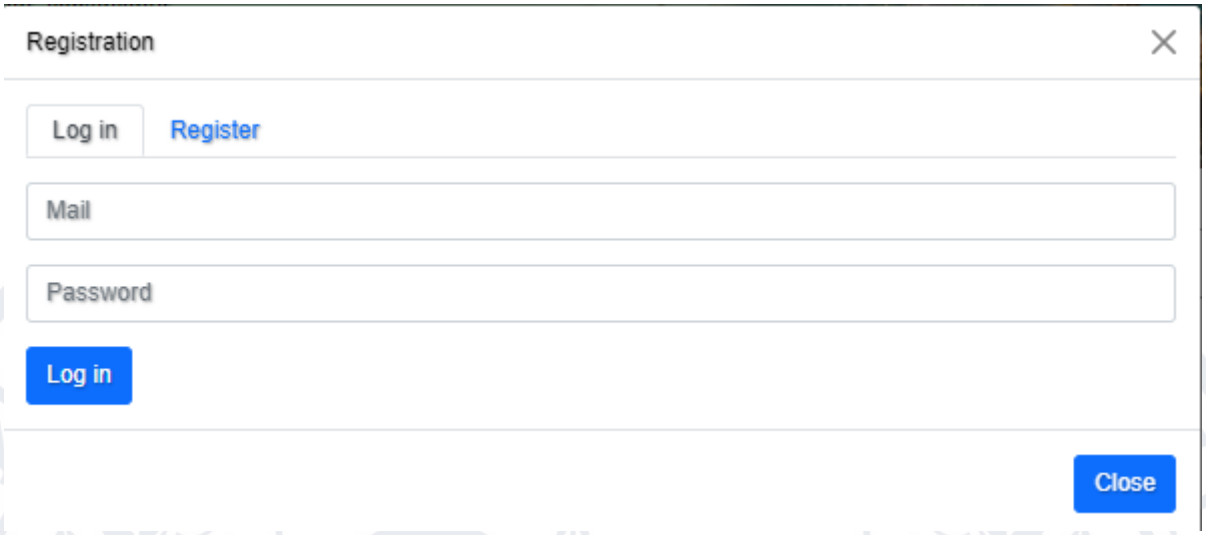


Рисунок 3.2.7 – Меню входу

Натиснувши кнопку “Login”, для користувача або адміністратора сайту відкривається модальне вікно для входу.

Щоб ввійти в спочатку потрібно ввести email та пароль, та натиснути кнопку Log in

8. Меню регістрації

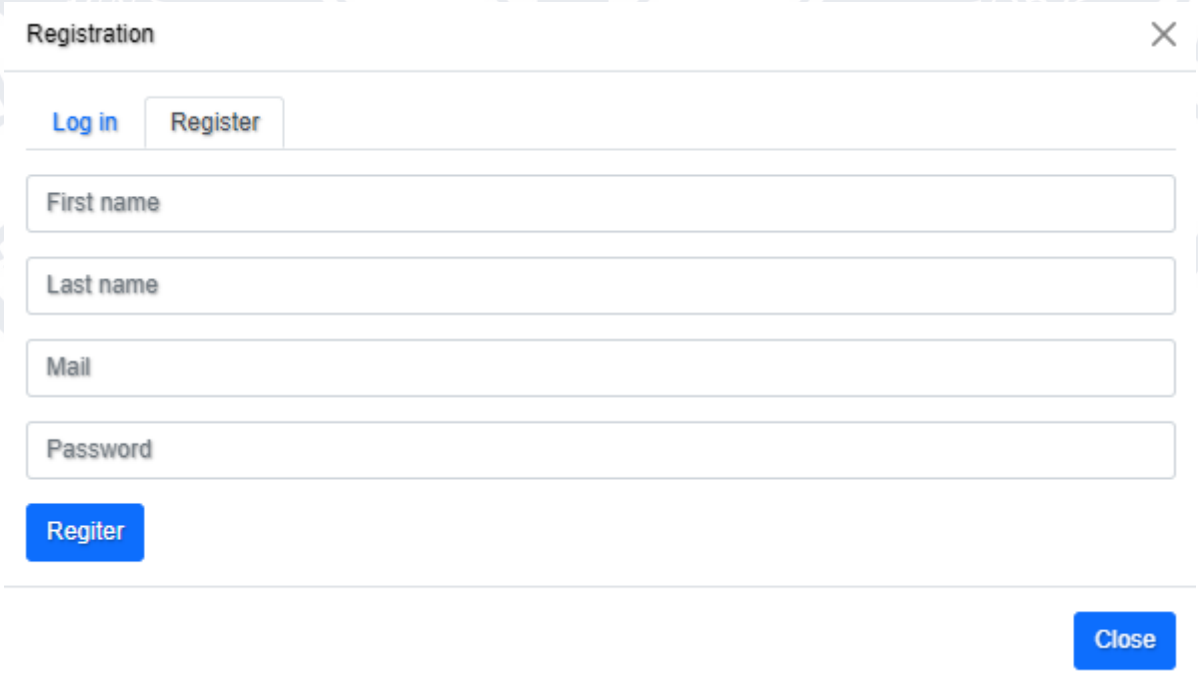


Рисунок 3.2.8 – Меню регістрації

Як і в 7 пункті, Натиснувши кнопку “Login”, для користувача відкривається меню входу, де він може переключитись на меню регістрації, якщо до того не був зареєстрований.

В даному меню користувачу потрібно ввести:

- Імя
- Прізвище
- E-mail
- Пароль

Після введеня інформації, потрібно натиснути кнопку Register, для продовження користування веб-сайтом.

9. Меню адміністратора

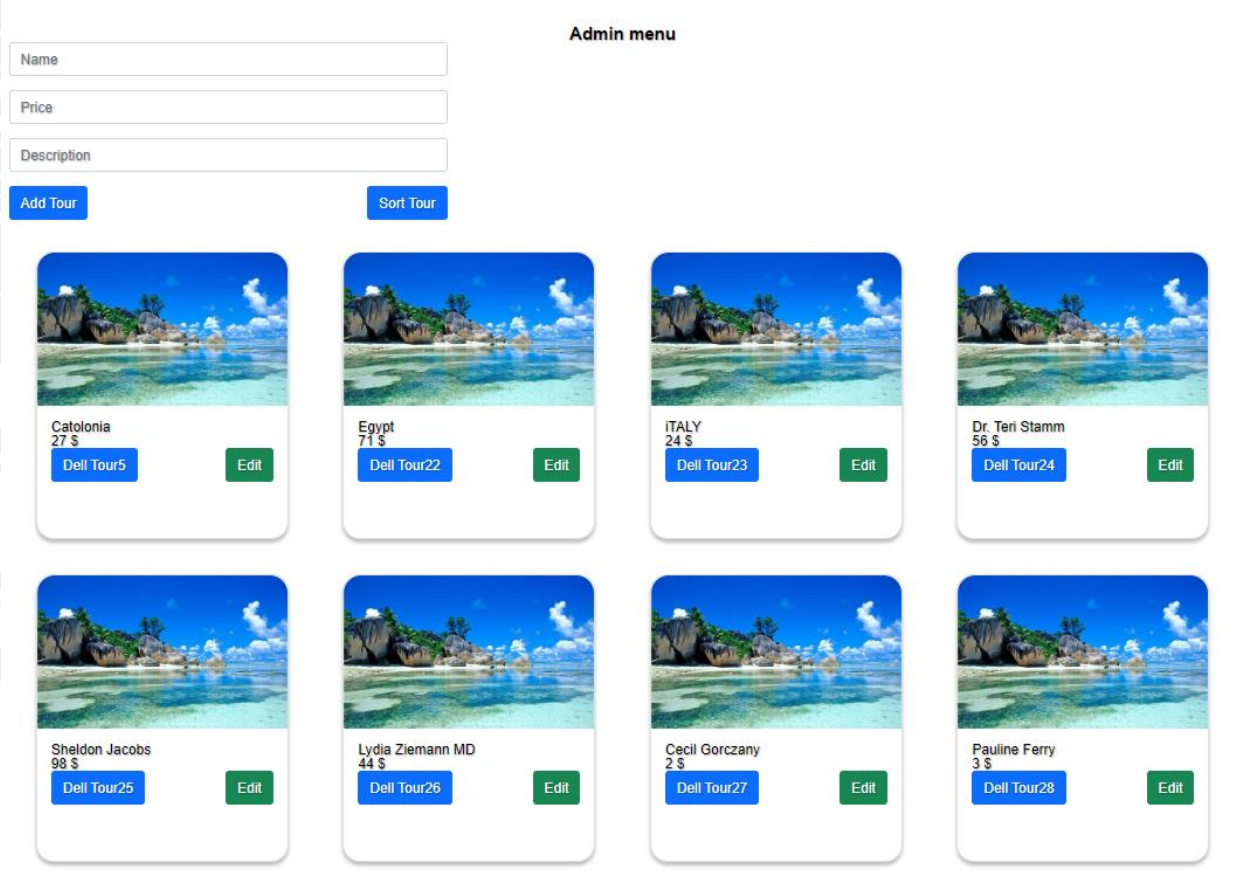


Рисунок 3.2.9 – Меню адміністратора

Меню адміністратора призначене для добавлення, редагування, або видалення турів.

Щоб добавити тур адміністратору потрібно ввести Назву туру, Його ціну, та написати або вставити готовий файл з текстом про даний тур. Файл з інформацією про тур буде оброблятись автоматичним реферуванням тексту, для більш кращої подачі інформації

10. Пошук туру

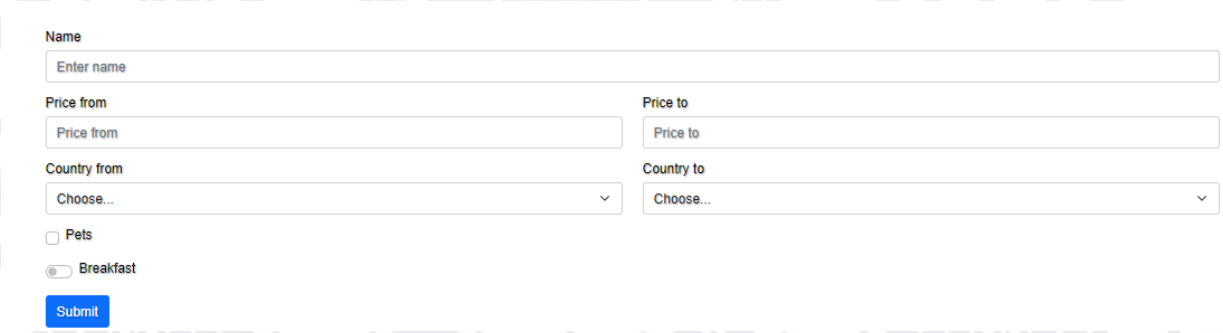


Рисунок 3.2.1.1 – Пошук

Коли користувач на головній сторінці, в меню навігації натискає кнопку Tours, відбувається перехід на сторінку зі всіма турами в якому присутній Пошук з фільтрами. В пошуку користувач може написати/обрати:

- Назву країни
- Обрати ціну від/до
- З якої країни він буде робити переліт і до якої
- Чи буде з ним домашній улюбленець
- Та обрати чи присутнє включення сніданку/вечері в турі

Після чого потрібно натиснути кнопку Submit, для перегляду турів з обраними параметрами.

11. Історія турів

History

Tour Name	Admin Name	Data	Type
name 1	adminName 1	2022-06-08T15:31:50.069Z	1
name 2	adminName 2	2022-06-09T03:25:45.274Z	2
name 3	adminName 3	2022-09-17T14:31:44.467Z	90
name 4	adminName 4	2022-09-18T06:05:21.590Z	69
name 5	adminName 5	2022-09-17T21:39:56.842Z	84
name 6	adminName 6	2022-09-17T22:52:41.707Z	5
name 7	adminName 7	2022-09-17T18:11:37.201Z	93
name 8	adminName 8	2022-09-18T01:54:30.076Z	39
name 9	adminName 9	2022-09-18T10:28:28.506Z	1

Рисунок 3.2.1.2 – Історія турів

Дане меню, доступне адміністратору веб-сторінки, для перегляду інформації, який тур буд добавлений, або видалений тощо

12. Відгуки

Рисунок 3.2.1.3 – Відгуки

Користувач, може залишити свій e-mail, для того щоб робітник турагенції звязався з ним для надання певного фідбеку від нього.

13. Сторінка про компацію

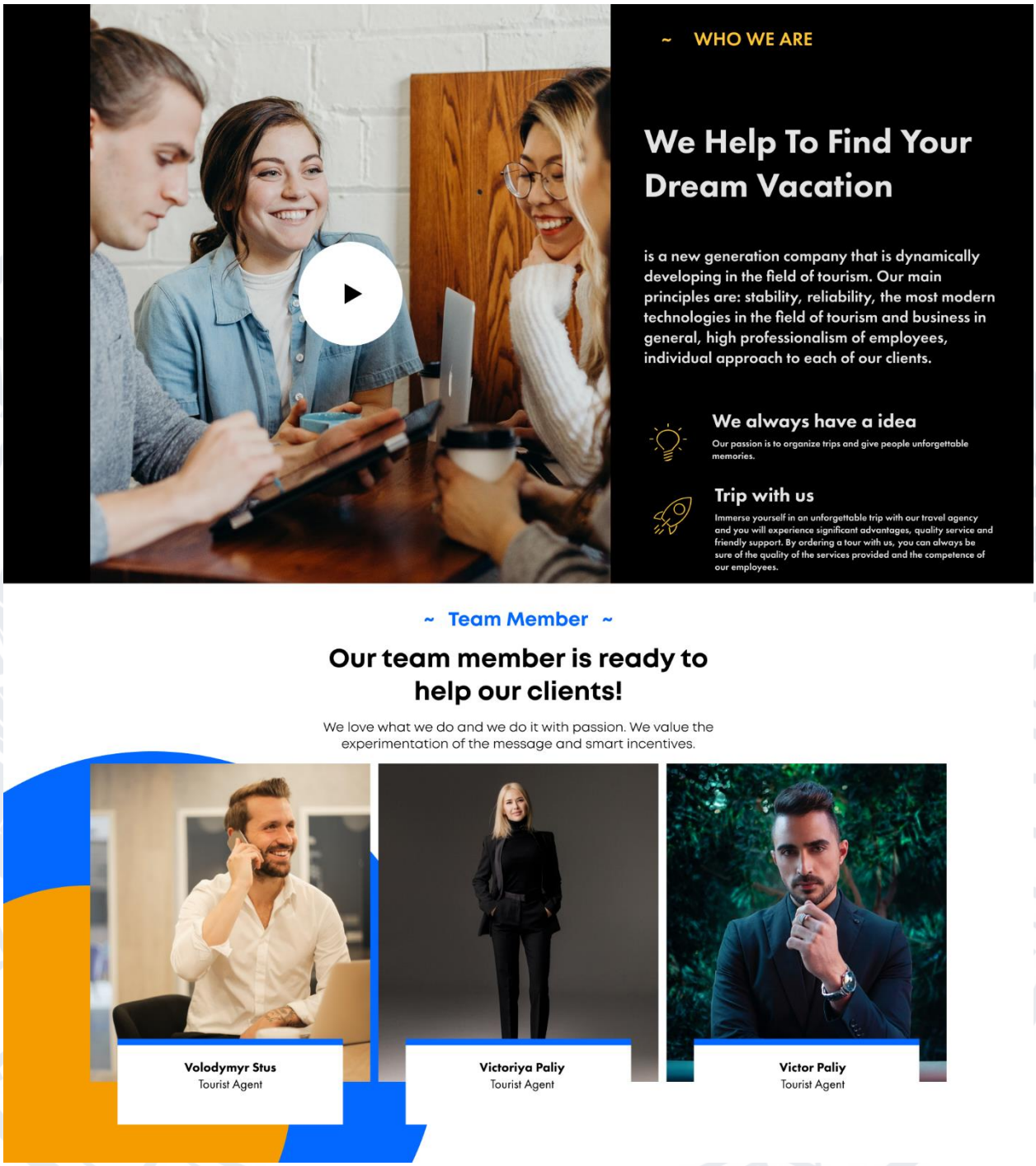


Рисунок 3.2.1.4 – Сторінка про компацію

Коли користувач на головній сторінці, в меню навігації натискає кнопку About, відбувається перехід на сторінку з інформацією про турагенцію

Тут можна подивитись певне відео з командою робітників, прочитати

певну інформацію, а також побачити команду туроператорів.

14.Пошук на головній сторінці

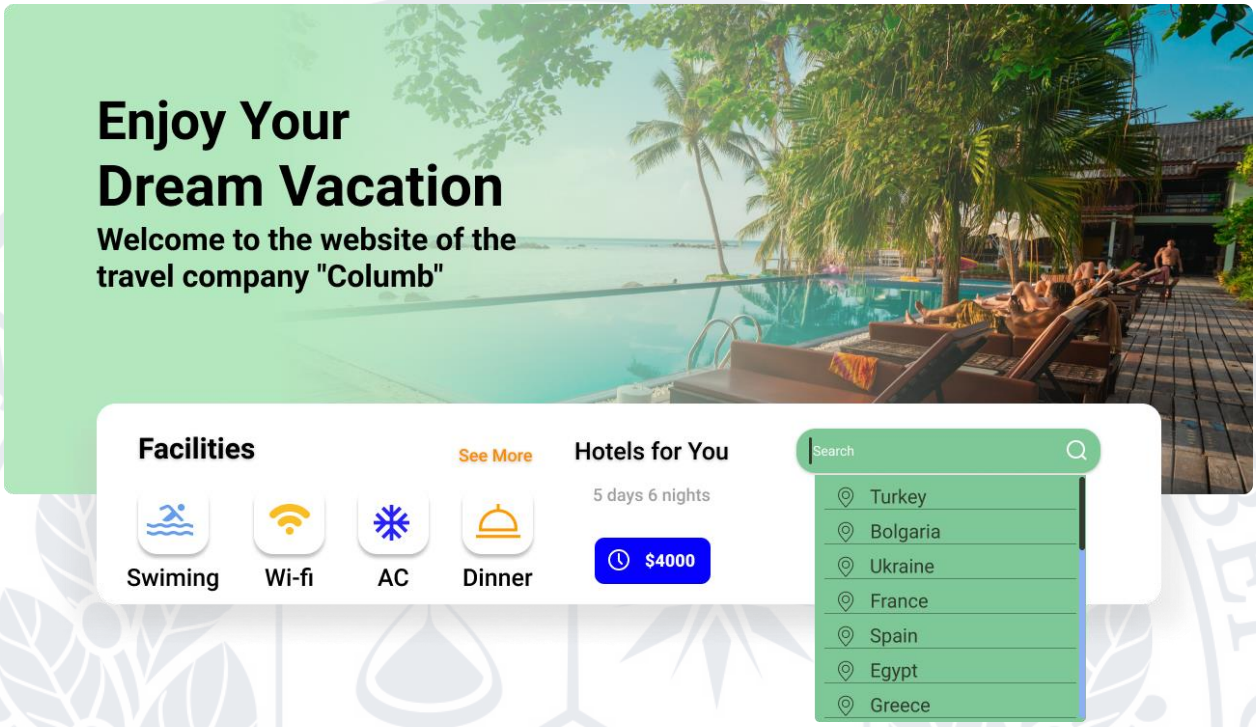


Рисунок 3.2.1.4 – Пошук на головній сторінці

Натиснувши на поле пошуку користувачу робиться активне вікно пошуку де він може ввести країну для подорожі, а також випадає список з підсказками по країнам в які присутні тури.

3.3 Опис використаної бази даних

В даній роботі використана база даних Mock API

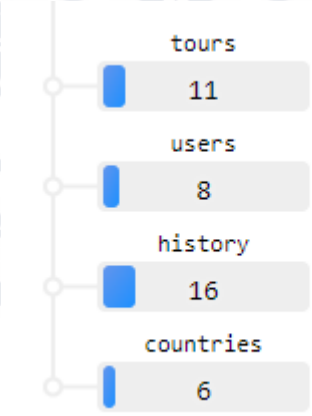


Рисунок 3.3.1

Вміст бази даних tours

Типи даних:


```
[
  {
    "createdAt": "2022-09-18T01:11:44.624Z",
    "name": "Catolonia",
    "price": 27,
    "description": "description 5",
    "pets": false,
    "from": "from 5",
    "to": "to 5",
    "discount": 23,
    "photo": "https://encrypted-
tbn0.gstatic.com/images?q=tbn:ANd9GcQYnJUMmW92Hvf_fl2RjidMRgyCTVBhxRMzT
```

W-4GdBpYHQDX2Ae86Oq4uHr8jVKKJQInJg&usqp=CAU",
    "owner": "Andre",
    "id": "5"
  },
  {
    "createdAt": "2022-09-17T16:32:06.924Z",
    "name": "Egypt",
    "price": 71,
    "description": "description 22",
    "pets": false,
    "from": "from 22",
    "to": "to 22",
    "discount": 27,
    "photo": "https://www.planetware.com/wpimages/2022/02/egypt-top-rated-resorts-four-seasons-resort-sharm-el-sheikh.jpg",
    "owner": "Aniya",
    "id": "22"
  },
  {
    "createdAt": "2022-09-18T09:01:26.242Z",
    "name": "Italy",
    "price": 24,
    "description": "description 23",
    "pets": false,
    "from": "from 23",
    "to": "to 23",
    "discount": 19,
    "photo": "https://luxuryhotel.com/wp-content/uploads/2021/05/luxury-island-resorts.jpg",
    "owner": "Lyric",
    "id": "23"
  },
  {
    "createdAt": "2022-09-18T08:56:36.054Z",
    "name": "Spain",
    "price": 56,
    "description": "description 24",
    "pets": false,
    "from": "from 24",
    "to": "to 24",
    "discount": 73,
    "photo": "photo 24",
    "owner": "Marta",
    "id": "24"
  },
  {

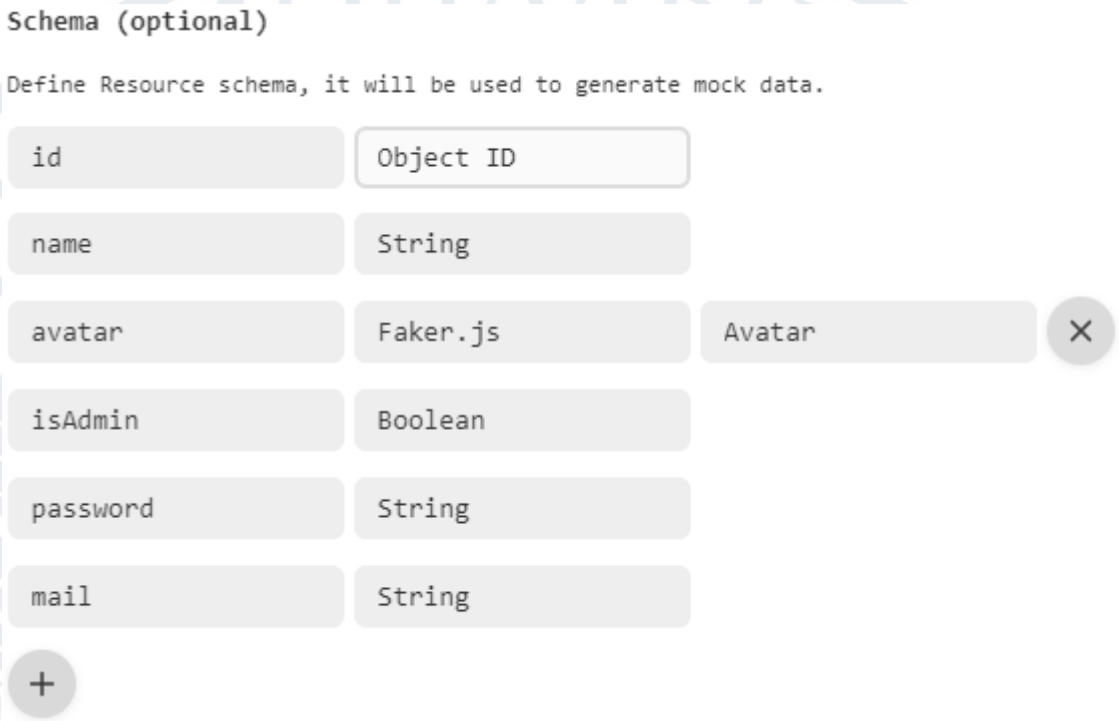
```
  "createdAt": "2022-09-17T13:43:08.612Z",
  "name": "Ukraine",
  "price": 98,
  "description": "description 25",
  "pets": false,
  "from": "from 25",
  "to": "to 25",
  "discount": 3,
  "photo": "photo 25",
  "owner": "Fernando",
  "id": "25"
},
{
  "createdAt": "2022-09-18T05:27:22.376Z",
  "name": "Maldiv",
  "price": 44,
  "description": "description 26",
  "pets": false,
  "from": "from 26",
  "to": "to 26",
  "discount": 60,
  "photo": "photo 26",
  "owner": "Ignacio",
  "id": "26"
},
{
  "createdAt": "2022-09-18T00:33:14.513Z",
  "name": "France",
  "price": 2,
  "description": "description 27",
  "pets": false,
  "from": "from 27",
  "to": "to 27",
  "discount": 81,
  "photo": "photo 27",
  "owner": "Laurianne",
  "id": "27"
},
{
  "createdAt": "2022-09-17T14:47:23.895Z",
  "name": "Pauline Ferry",
  "price": 3,
  "description": "description 28",
  "pets": false,
  "from": "from 28",
```

```
  "to": "to 28",
  "discount": 12,
  "photo": "photo 28",
  "owner": "Brenna",
  "id": "28"
},
{
  "createdAt": "2022-09-17T18:25:16.337Z",
  "name": "Odesa",
  "price": 999,
  "description": "TourGG",
  "pets": false,
  "from": "from 29",
  "to": "to 29",
  "discount": 58,
  "photo": "photo 29",
  "owner": "Bettie",
  "id": "29"
},
{
  "createdAt": "2022-10-13T04:16:22.022Z",
  "name": "",
  "price": 0,
  "description": "",
  "pets": false,
  "from": "from 30",
  "to": "to 30",
  "discount": 11,
  "photo": "photo 30",
  "owner": "Bobby",
  "id": "30"
},
{
  "createdAt": "2022-10-13T02:40:49.803Z",
  "name": "",
  "price": 0,
  "description": "",
  "pets": false,
  "from": "from 31",
  "to": "to 31",
  "discount": 86,
  "photo": "photo 31",
  "owner": "Godfrey",
  "id": "31"
}
```

]

Вміст бази даних users

Типи даних:



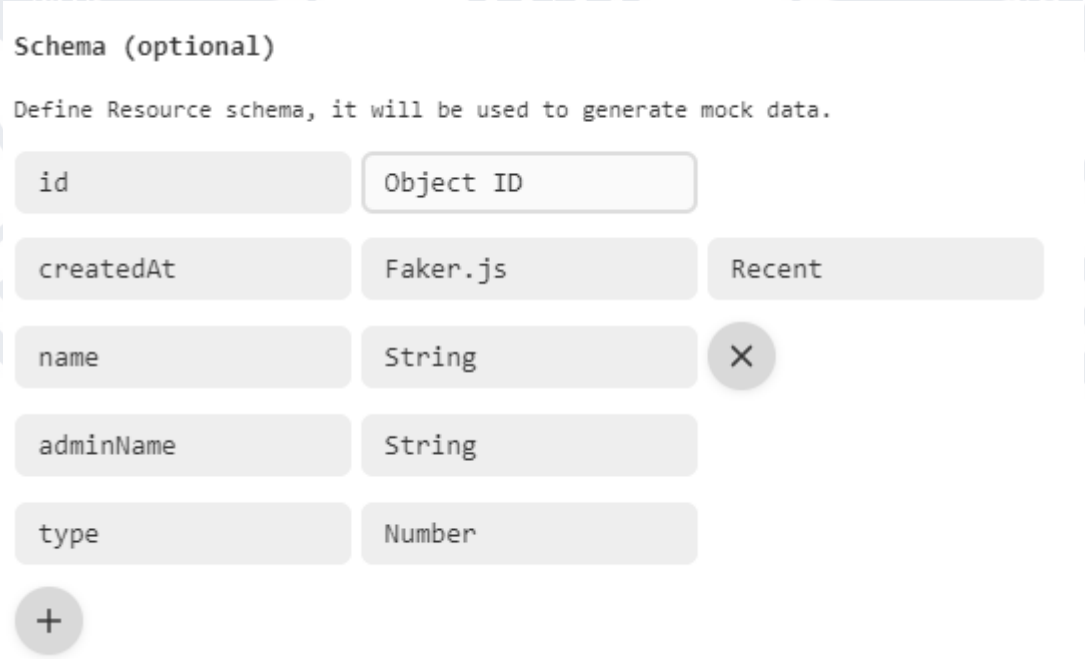

```
[
 {
  "name": "admin",
  "avatar": "https://cloudflare-ipfs.com/ipfs/Qmd3W5DuhgHirLHGVixi6V76LhCkZUz6pnFt5AJBiyvHye/avatar/279.jpg",
  "isAdmin": true,
  "password": "*******",
  "mail": "admin",
  "id": "1"
 },
 {
  "name": "maks",
  "avatar": "https://cloudflare-ipfs.com/ipfs/Qmd3W5DuhgHirLHGVixi6V76LhCkZUz6pnFt5AJBiyvHye/avatar/343.jpg",
  "isAdmin": true,
  "password": "*******",
```

```
 "mail": "maks",
 "id": "2"
 },
 {
 "name": "name 3",
 "avatar": "https://cloudflare-
ipfs.com/ipfs/Qmd3W5DuhgHirLHGVixi6V76LhCkZUz6pnFt5AJBiyvHye/avatar/213.jpg
",
 "isAdmin": false,
 "password": "password 3",
 "mail": "mail 3",
 "id": "3"
 },
 {
 "name": "oleg",
 "avatar": "https://cloudflare-
ipfs.com/ipfs/Qmd3W5DuhgHirLHGVixi6V76LhCkZUz6pnFt5AJBiyvHye/avatar/448.jpg
",
 "isAdmin": false,
 "password": "*******",
 "mail": "oleg",
 "id": "4"
 },
 {
 "name": "name 5",
 "avatar": "https://cloudflare-
ipfs.com/ipfs/Qmd3W5DuhgHirLHGVixi6V76LhCkZUz6pnFt5AJBiyvHye/avatar/86.jpg",
 "isAdmin": false,
 "password": "*******",
 "mail": "mail 5",
 "id": "5"
 },
 {
 "name": "name 6",
 "avatar": "https://cloudflare-
ipfs.com/ipfs/Qmd3W5DuhgHirLHGVixi6V76LhCkZUz6pnFt5AJBiyvHye/avatar/434.jpg
",
 "isAdmin": false,
 "password": "*******",
 "mail": "mail 6",
 "id": "6"
 },
 {
 "name": "name 7",
 "avatar": "https://cloudflare-
```

```
ipfs.com/ipfs/Qmd3W5DuhgHirLHGVixi6V76LhCkZUz6pnFt5AJBiyvHye/avatar/552.jpg
",
  "isAdmin": false,
  "password": "*******",
  "mail": "mail 7",
  "id": "7"
 },
 {
  "name": "Roma",
  "avatar": "https://cloudflare-
ipfs.com/ipfs/Qmd3W5DuhgHirLHGVixi6V76LhCkZUz6pnFt5AJBiyvHye/avatar/1217.jp
g",
  "isAdmin": true,
  "password": "*******",
  "mail": "romatest",
  "id": "8",
  "firstName": "Roma",
  "secondName": "Test"
 }
]
```


Вміст бази даних history

Типи даних:

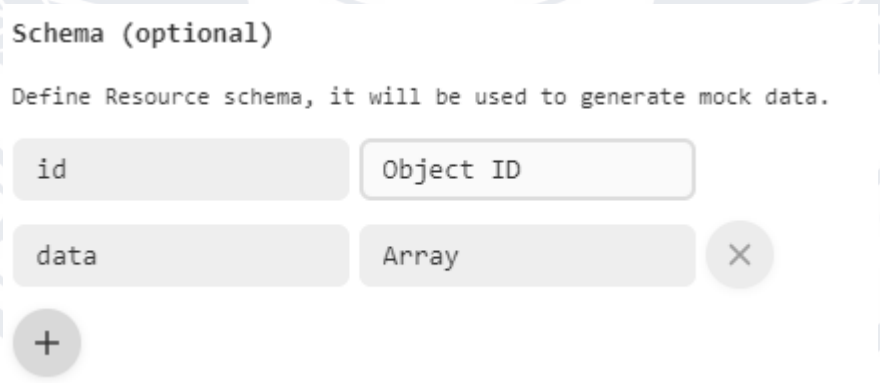


```
[
 {
  "createdAt": "2022-06-08T15:31:50.069Z",
  "name": "name 1",
  "adminName": "adminName 1",
  "type": 1,
  "id": "1"
 },
 {
  "createdAt": "2022-06-09T03:25:45.274Z",
  "name": "name 2",
  "adminName": "adminName 2",
  "type": 2,
  "id": "2"
 },
 {
  "createdAt": "2022-09-17T14:31:44.467Z",
  "name": "name 3",
  "adminName": "adminName 3",
  "type": 90,
  "id": "3"
 },
 {
  "createdAt": "2022-09-18T06:05:21.590Z",
  "name": "name 4",
  "adminName": "adminName 4",
  "type": 69,
  "id": "4"
 },
 {
  "createdAt": "2022-09-17T21:39:56.842Z",
  "name": "name 5",
  "adminName": "adminName 5",
  "type": 84,
  "id": "5"
 },
 {
  "createdAt": "2022-09-17T22:52:41.707Z",
  "name": "name 6",
  "adminName": "adminName 6",
  "type": 5,
  "id": "6"
 },
 {
```

```
 "createdAt": "2022-09-17T18:11:37.201Z",
 "name": "name 7",
 "adminName": "adminName 7",
 "type": 93,
 "id": "7"
},
{
 "createdAt": "2022-09-18T01:54:30.076Z",
 "name": "name 8",
 "adminName": "adminName 8",
 "type": 39,
 "id": "8"
},
{
 "createdAt": "2022-09-18T10:28:28.506Z",
 "name": "name 9",
 "adminName": "adminName 9",
 "type": 1,
 "id": "9"
},
{
 "createdAt": "2022-09-18T04:58:31.067Z",
 "name": "name 10",
 "adminName": "adminName 10",
 "type": 58,
 "id": "10"
},
{
 "createdAt": "2022-09-17T22:26:03.930Z",
 "name": "name 11",
 "adminName": "adminName 11",
 "type": 95,
 "id": "11"
},
{
 "createdAt": "2022-09-17T23:37:59.470Z",
 "name": "name 12",
 "adminName": "adminName 12",
 "type": 63,
 "id": "12"
},
{
 "createdAt": "2022-09-18T12:30:11.290Z",
 "name": "name 13",
 "adminName": "adminName 13",
```

```
 "type": 2,
 "id": "13"
 },
 {
 "createdAt": "2022-09-18T06:07:13.598Z",
 "name": "name 14",
 "adminName": "adminName 14",
 "type": 38,
 "id": "14"
 },
 {
 "createdAt": "2022-09-18T04:05:20.571Z",
 "name": "name 15",
 "adminName": "adminName 15",
 "type": 59,
 "id": "15"
 },
 {
 "createdAt": "2022-09-17T13:50:45.504Z",
 "name": "name 16",
 "adminName": "adminName 16",
 "type": 32,
 "id": "16"
 }
]
```


Вміст бази даних countries




```
[
 "Turkey",
 "Ukraine",
 "USA",
```

```
 "Poland",
 "Mexico",
 "Canada"
]
```

3.4 Опис основного коду програми

Основний код прогарми був написаний за допомогою інструментів середовища JavaScript: React та TypeScript. Для перегляду виконання коду, веб-сайт запускається за допоиогою Node JS. Основні елементи сайту, прописані в компонентах. Вміст веб-додатку змінюється як в самому коді програми, а також в базі даних MockApi. Детальний вміст коду середовища даного веб додатку можна побачити на Рисунок 3.4.1.

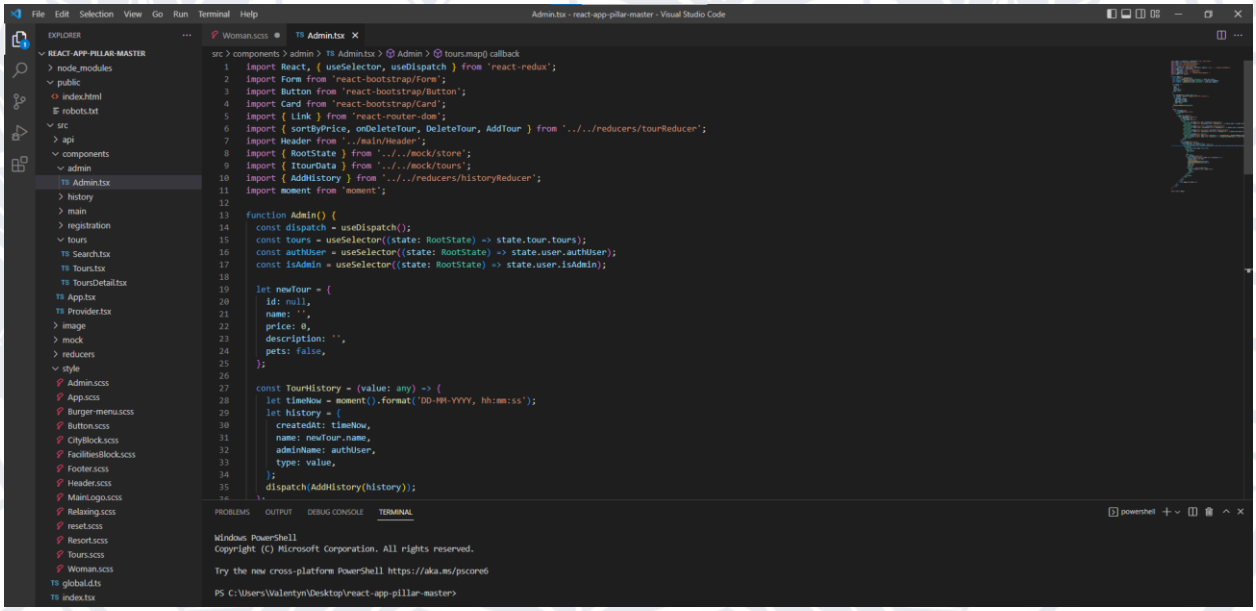


Рисунок 3.4.1 – Середовище розробленого продукту

Висновок до 3 розділу

У розділі 3 наведено програмну реалізацію веб-ресурсу туристичної агенції. Здійснено опис бази даних. Розроблено структуру програми.

Проведено опис інтерфейсу програми та викладено основний код програми

Висновок

В 1 розділі ми розібрали актуальність вебсайту для туристичної агенції, та його переваги. Також детально розглянули методи Екстраційні методи реферування та Абкстрактні методи реферування тексту. Було розглянуто аналоги продукту, який буде розроблятись, а також їх плюси та мінуси. Поставили завдання для проектування та розробки інтерфейсу веб-ресурсу.

В 2 розділі було зроблено опис предметної області. Було зроблено моделювання UML-засобами, опис самого URL. Описано алгоритм автоматичного реферування тексту, на прикладі мові Python, який буде використаний в даній роботі а саме LexRank Algoritm, та його порівняння з TextRank Algoritm. Розробили логічну структура сайту.

У розділі 3 наведено програмну реалізацію веб-ресурсу туристичної агенції. Здійснено опис бази даних. Розроблено структуру програми.

Проведено опис інтерфейсу програми та викладено основний код програми

Використанні джерела

1. Аннотирование и реферирование [Електронний ресурс] – Режим доступу: http://poznayka.org/s39440t1.html

2. Key Technology Trends Emerging in the Travel & Tourism Industry. [Електронний ресурс] – Режим доступу: https://www.revfine.com/technology-trends-travel-industry/

3. Vidal B. The New Technology and Travel Revolution. Електронний ресурс] – Режим доступу: https://www.wearemarketing.com/blog/tourism-and-technology-how-tech-isrevolutionizing-travel.html#

4. Utheim H. What is virtual tourism and when should you make use of

it? [Електронний ресурс] – Режим доступу: https://travelopment.com/what-is-virtual-tourism-and-when-should-youmake-use-of-it/

5. Интеллектуальные информационные технологии [Електронний ресурс] – Режим доступу: http://www.irkinfo.ru/intellektualnye-informatsionnye-tekhnol ogii-str34.html

6. Интеллектуальные информационные технологии [Електронний ресурс] – Режим доступу: http://www.irkinfo.ru/intellektualnye-informatsionnye-tekhnol ogii-str34.html

7. Автоматизированный анализ информационных ресурсов управления машиностроительным предприятием [Електронний ресурс] – Режим доступу: http://bibldis.narod.ru/2009/55.htm

8. Machine learning text feature extraction (tf-idf) [Електронний ресурс] – Режим доступу: http://blog.christianperone.com/2011/10/machine-learningtext-feature-extraction-tf-idf-part-ii/

9. A Comprehensive Introduction to Word Vector Representations [Електронний ресурс] – Режим доступу: https://medium.com/ai-society/jkljlj-7d6e699895c4

10. Моделирование текстов с использованием рекуррентных нейронных сетей/ И.Д. Чернобаев, А.С. Суркова, А.З. Панкратова, 2018

11. ROUGE: A Package for Automatic Evaluation of Summaries [Електронний ресурс] – Режим доступу: http://www.aclweb.org/anthology/W04-1013

12. GloVe: Global Vectors for Word Representation [Електронний ресурс] – Режим доступу: https://www.kaggle.com/rtatman/glove-global-vectors-forword-representation

13. . Single-document and multi-document summary evaluation using Relative Utility [Електронний ресурс] – Режим доступу: https://pdfs.semanticscholar.
org/5a2e/a7bd8cd14fc87a23bbe9540033aa0fb75682.pdf

14. bootstrap Tutorial [Електронний ресурс]. Режим доступу:

https://www.w3schools.com/bootstrap/bootstrap_ver.asp

15. jQueryTutorial [Електронний ресурс]. Режим доступу: https://www.w3schools.com/jquery/default.asp

16. Anushree R. Survey on Big Data Mining Algorithms. // International Journal for Research in Applied Science and Engineering Technology, 2019. N7. P. 1363-1367

17. Bathla G., Aggarwal H., Rani R. Evolution of Data Analytics Techniques: From Data Mining to Big Data Mining // International Journal of Advanced Science and Technology, 2019

18. Becerra-Bonache L., Bel-Enguix G., Jimenez-Lopez M.D., Martín-Vide C.: Formal Grammars and Languages. In R. Mitkov (Ed): The Oxford Handbook of Computational Linguistics (2nd Edition), Oxford University Press

19. Java Script Tutorial [Електронний ресурс]. Режим доступу: https://www.w3schools.com/js/default.asp

20. Класифікація текстів на природній мові за допомогою нейронної мережі / І.М. Шпінарева, О.А. Геренко, К.Ю. Морозова – Збірник наукових праць Військового інституту Київського національного університету ім. Т.Шевченка.

21. Short-Term Load Forecasting Using EMD-LSTM Neural Networks with a Xgboost Algorithm for Feature Importance Evaluation/ Huiting Zheng, Jiabin Yuan, Long Chen, 2017 [Електронний ресурс] – Режим доступу: http://www.mdpi.com/1996-1073/10/8/1168/htm

22. LexRank: Graph-based Lexical Centrality as Salience in Text Summarization [Електронний ресурс] – Режим доступу: https://www.cs.cmu.edu/afs/cs/project/jair/pub/volume22/erkan04a-html/erkan04a.html

Додаток А

```
import React, { useSelector, useDispatch } from 'react-redux';
import Form from 'react-bootstrap/Form';
import Button from 'react-bootstrap/Button';
import Card from 'react-bootstrap/Card';
import { Link } from 'react-router-dom';
import { sortByPrice, onDeleteTour, DeleteTour, AddTour } from
'../../reducers/tourReducer';
import Header from '../main/Header';
import { RootState } from '../../mock/store';
import { ItourData } from '../../mock/tours';
import { AddHistory } from '../../reducers/historyReducer';
import moment from 'moment';

function Admin() {
  const dispatch = useDispatch();
  const tours = useSelector((state: RootState) => state.tour.tours);
  const authUser = useSelector((state: RootState) => state.user.authUser);
  const isAdmin = useSelector((state: RootState) => state.user.isAdmin);

  let newTour = {
    id: null,
    name: '',
    price: 0,
    description: '',
    pets: false,
  };

  const TourHistory = (value: any) => {
```

```
  let timeNow = moment().format('DD-MM-YYYY, hh:mm:ss');
  let history = {
    createdAt: timeNow,
    name: newTour.name,
    adminName: authUser,
    type: value,
  };
  dispatch(AddHistory(history));
};

 return (
  <div className="App">
   <div className="App__first">
    <Header />
    {
     isAdmin && authUser ? (
      <div className="change-tour">
       <h1>Admin menu</h1>
       <div className="add-tour">
        <Form>
         <Form.Group className="mb-3 Name" controlId="formBasicEmail">
          <Form.Control type="text" placeholder="Name" onChange={(e) => {
newTour.name = e.target.value; }} />
         </Form.Group>
         <Form.Group className="mb-3 Price"
controlId="formBasicPassword">
          <Form.Control type="number" placeholder="Price" onChange={(e)
=> { newTour.price = parseInt(e.target.value); }} />
         </Form.Group>
```

```
            <Form.Group className="mb-3 Description"
controlId="formBasicCheckbox">
              <Form.Control type="text" placeholder="Description" onChange={(e)
=> { newTour.description = e.target.value; }} />
            </Form.Group>
            <Form.Group className="mb-3 add-sort">
             <Button variant="primary" type="reset" onClick={() => {
TourHistory(1), dispatch(AddTour(newTour)) }}>Add Tour</Button>
              <Button variant="primary" type="reset" onClick={() => {
dispatch(sortByPrice()); }}>Sort Tour</Button>
            </Form.Group>
           </Form>
          </div>
          <div className="dell-tours">
           {tours.map((tour: ItourData) => (
            <Card key={tour.id} style={{ width: '18rem' }}>
             <Card.Img variant="top" style={{ borderRadius: '20px 20px 0 0' }}
src="https://img.rozavitriv.com/3/540x370/00/00/16/63/166325.jpg" />
             <Card.Body>
              <Card.Title>{tour.name}</Card.Title>
              <Card.Text>
               {tour.price}
               {' '}
               $
              </Card.Text>
              <div className='admin_btn'>
               <Button variant="primary" type="reset" onClick={() => {
                newTour.name = tour.name,
                TourHistory(0),
                dispatch(onDeleteTour(tour.id)),
```

```
                    dispatch(DeleteTour(tour.id)) }}>
                    Dell Tour
                    {tour.id}
                  </Button>
                  <Link to={`../tours/${tour.id}`}>
                    <Button variant="success" type="reset">
                     Edit
                    </Button>
                  </Link>
                </div>
              </Card.Body>
            </Card>
          ))}
        </div>
      </div>
    ) : (<div>page not found</div>)
  }
  </div>
 </div>
 );
}

export default Admin;
```

Олексієнко Валентин Віталійович
Прізвище, ім'я по батькові

Інформаційних і прикладних технологій
Факультет

122 Комп'ютерні науки
Шифр і назва спеціальності

Комп'ютерні технологій обробки даних
Освітня програма

ДЕКЛАРАЦІЯ АКАДЕМІЧНОЇ ДОБРОЧЕСНОСТІ

Усвідомлюючи свою відповідальність за надання неправдивої інформації, стверджую, що подана кваліфікаційна (магістерська) робота на тему «ДОСЛІДЖЕННЯ КОНТЕНТУ САЙТУ ТУРИСТИЧНОГО АГЕНСТВА НА ПІДСТАВІ МЕТОДІВ DATA SCIENCE» є написаною мною особисто.

Одночасно заявляю, що ця робота:

- не передавалась іншим особам і подається до захисту вперше;
- не порушує авторських та суміжних прав, закріплених статтями 21-25 Закону України «Про авторське право та суміжні права»;
- не отримувалась іншими особами, а також дані та інформація не отримувалась у недозволений спосіб.

Я усвідомлюю, що у разі порушення цього порядку моя кваліфікаційна робота буде відхилена без права її захисту, або під час захисту за неї буде поставлена оцінка «незадовільно».

_____ _____
(дата) (підпис здобувача освіти)