МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ АВІАЦІЙНИЙ УНІВЕРСИТЕТ
ФАКУЛЬТЕТ КІБЕРБЕЗПЕКИ
КОМП'ЮТЕРНОЇ ТА ПРОГРАМНОЇ ІНЖЕНЕРІЇ
КАФЕДРА КОМП'ЮТЕРИЗОВАНИХ СИСТЕМ УПРАВЛІННЯ

ДОПУСТИТИ ДО ЗАХИСТУ
Завідувач кафедри

_____ Литвиненко О. Є.
"___" _____ 2020 р.

ДИПЛОМНА РОБОТА
(ПОЯСНЮВАЛЬНА ЗАПИСКА)

ВИПУСКНИКА ОСВІТНЬОГО СТУПЕНЯ
"МАГІСТР"

Тема: Комп'ютерна система надання послуг з обробки текстової інформації

Виконавець: Іванілов А. Р.

Керівник: Вавіленкова А. І.

Нормоконтролер: Тупота Є. В.

Київ 2020

НАЦІОНАЛЬНИЙ АВІАЦІЙНИЙ УНІВЕРСИТЕТ

Факультет кібербезпеки, комп'ютерної та програмної інженерії

Кафедра комп'ютеризованих систем управління

Спеціальність 123 «Комп'ютерна інженерія»
(шифр, найменування)

Освітньо-професійна програма «Системне програмування»

ЗАТВЕРДЖУЮ
Завідувач кафедри

Литвиненко О. Є.
«_____»_____ 2020 р.

ЗАВДАННЯ
на виконання дипломної роботи
Іванілова Антона Ростиславовича
(прізвище, ім'я, по батькові випускника в родовому відмінку)

1. Тема дипломної роботи: «Комп'ютерна система надання послуг з обробки текстової інформації»

затверджена наказом ректора від «27» серпня 2020 р. № 1203/ст

2. Термін виконання роботи: з 05 жовтня 2020 р. по 13 грудня 2020 р.

3. Вхідні дані до роботи: технічна документація, тестові дані, програмні продукти

4. Зміст пояснювальної записки:

1) Аналіз існуючих систем надання послуг та обробки текстової інформації

2) Функціональні особливості та алгоритми роботи комп'ютерної системи надання послуг з обробки текстової інформації

3) Результати та якісні показники функціонування комп'ютерної системи надання послуг з оробки текстової інформації

5. Перелік обов'язкового графічного (ілюстрованого) матеріалу:

1) Структура класичного маркетплейсу

2) Процес функціонування алгоритму визначення мови введеного тексту (схема функціональна)

3) Організаційна структура комп'ютерної системи надання послуг з обробки текстової інформації (схема структурна)

4) Інтерфейс розробленої системи надання послуг з обробки текстової інформації

5) Результат роботи програмного модулю системи надання послуг з обробки текстової інформації

6) Діаграма порівняння якості роботи функції синонімізації розробленої системи та сторонніх популярних систем

6. Календарний план-графік

№ пор.	Завдання	Термін виконання	Відмітка про виконання
1	Проаналізувати існуючі системи надання послуг та обробки текстової інформації	05.10.2020 – 10.10.2020	
2	Визначити особливості та принципи моделі маркетплейс	10.10.2020 – 14.10.2020	
3	Дослідити основні алгоритми обробки текстової інформації	14.10.2020 – 28.10.2020	
4	Реалізувати алгоритми та методи обробки натуральної мови у вигляді програмного коду	28.10.2020 – 14.11.2020	
5	Розробити комп’ютерну систему надання послуг з обробки текстової інформації	15.11.2020 – 01.12.2020	
6	Проаналізувати результати роботи та якісні показники функцій системи	01.12.2020 – 05.12.2020	
7	Оформити пояснювальну записку	05.12.2020 – 10.12.2020	
8	Оформити графічний та ілюстративний матеріал	10.12.2020 – 13.12.2020	

7. Дата видачі завдання: « 05 » жовтня 2020 р.

Керівник дипломної роботи _____ Вавіленкова А. І.
(підпис керівника) (П.І.Б.)

Завдання прийняв до виконання _____ Іванілов А. Р.
(підпис студента) (П.І.Б.)

РЕФЕРАТ

Пояснювальна записка до дипломної роботи «Комп’ютерна система надання послуг з обробки текстової інформації»: 104 сторінки, 39 рисунків, 8 таблиць, 27 використаних джерел.

ОБРОБКА ТЕКСТОВОЇ ІНФОРМАЦІЇ, АЛГОРИТМИ ОБРОБКИ ПРИРОДНОЇ МОВИ, МАРКЕТПЛЕЙС, КОМП’ЮТЕРНА СИСТЕМА.

Об’єкт дослідження дипломної роботи – процес обробки текстової інформації.

Предмет дослідження дипломної роботи – комп’ютерні системи надання послуг з обробки текстової інформації.

Мета дипломної роботи – створити та проаналізувати роботу комп’ютерної системи надання послуг з обробки текстової інформації з використанням нової моделі надання послуг – маркетплейсу.

Методи дослідження – алгоритми визначення тональності тесту, алгоритм виділення компонент електронного текстового документу, методи автореферування, методи об’єктно-орієнтованого програмування.

Здійснено аналіз існуючих сервісів обробки текстової інформації; досліджено методи створення маркетпейсів та автоматизованих систем обробки текстової інформації; вивчено алгоритми, що використовуються у функціях для обробки текстової інформації; реалізовано алгоритми для обробки текстової інформації у вигляді автоматизованої системи на основі сучасних мов програмування; проаналізовано результати та якість роботи створеної комп’ютерної системи та сервісів з аналогічним функціоналом.

Матеріали дипломної роботи рекомендується використовувати для автоматичної обробки електронних текстових документів, при проведенні наукових досліджень, у навчальному процесі фахівців з системного програмування.

Прогнозні припущення про розвиток об’єкту та предмету дослідження – новітній підхід до подачі інформації та необхідного функціоналу можна застосовувати для різних напрямків обробки текстової інформації.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ ТА СКОРОЧЕНЬ

HTML (HyperText Markup Language) – стандартизована мова розмітки веб-сторінок;

API (Application Programming Interface) – прикладний програмний інтерфейс;

XML (*Extensible Markup Language*) – розширена мова розмітки. Текстовий формат для обміну даними;

SQL (*Structured Query Language*) – мова структурованих запитів. Для взаємодії з базами даних;

UI (*User interface*) – графічний інтерфейс користувача;

MVC (*Model-View-Contoller*) – паттерн веб програмування. Складається з моделі сутності, візуального представлення та контроллеру виконання логіки;

SPA (*Single-page application*) – односторінковий додаток. Паттерн веб-програмування;

PHP (*Hypertext Preprocessor*) – мова веб-програмування;

JS (*Javascript*) – скриптова мова програмування;

Vue (*Vue.js*) – фреймворк мови програмування *Javascipt*;

Redis – утіліта зберігання даних у оперативній пам’яті;

LiqPay – інтерфейс для здійснення банківських операцій.

ВСТУП

Актуальність теми. З розвитком мережі Інтернет за останні п’ять років дедалі більше нових технлогій розвивається саме в цій сфері. Перехід від стаціонарного програмного забезпечення (ПО) – до онлайн-ПО виглядає цілком логічним з огляду на те, що користувачі вже не хочуть чекати встановлення програми на комп’ютер, розбиратися із налаштуваннями програми та неможливістю отримувати зворотній зв’язок у випадку недієздатності додатку. Набагато легшим виглядає використовувати додаток в мережі, використовуючі лише Браузер та Інтернет. До того ж Інтернет зараз є практично у кожного користувача, що має комп’ютер і смартфон. Також розвиток смартфонів підштовхує виробників розвивати онлайн-додатки, що виглядає і дешевшим і практичнішим рішенням, хоча б з огляду на те, що необхідність розробляти один і той самий додаток на декілька стаціонарних операційних систем та мобільних програє перед можливістю розробити один веб-додаток, що буде достійний з будь-якої платформи.

Комп’ютерні системи з обробки текстової інформації, що доступні користувачам зараз, мають цілу низку проблем. До неї входять і велика вартість програм, і застарілий інтерфейс, що змушує клієнта читати велику інструкцію по керуванню, і також розробка під дуже вузьку кількість платформ.

Тому на даний момент є актуальним створення системи надання послуг з оробки текстової інформації у веб-просторі за моделлю маркетплейс, що дозволить розширити кількість платформ, що підтримуються, тим самим підвищити задоволеність користувача системою та зібрати всі алгоритми з обробки текстової інформації в одному місці, прибираючи необхідність завантажувати та розбиратись у великій кількості додатків та їх інструкцій для отримання стандартних функцій.

Мета і завдання виконання дипломної роботи. Мета дипломної роботи – створити та проаналізувати роботу комп'ютерної системи надання послуг з обробки текстової інформації з використанням нової моделі надання послуг – маркетплейсу.

Для досягнення поставленої мети необхідно вирішити такі завдання:

– ознайомитися з існуючими сервісами обробки текстової інформації;
– дослідити методи створення маркетплейсів та автоматизованих систем обробки текстової інформації;
– вивчити алгоритми, що використовуються у функціях для обробки текстової інформації (визначення мови тексту, знаходження синонімів тощо);
– реалізувати алгоритми для обробки текстової інформації у вигляді автоматизованої системи з використанням мов програмування *PHP*, *Python* та *JavaScript*;
– створити комп'ютерну систему надання послуг з обробки текстової інформації, реалізовану у вигляді маркетплейсу;
– проаналізувати результати роботи створеної комп'ютерної системи та сервісів з аналогічним функціоналом.

Об'єкт і предмет дослідження. Об'єктом дослідження даної роботи є процес обробки текстової інформації. Предметом дослідження дипломної роботи є комп'ютерні системи надання послуг з обробки текстової інформації.

Методи дослідження. Було досліджено метод побудови ймовірнісної моделі за допомогою *N*-грам символів для визначення мови введеної текстової інформації; реалізовано у вигляді програмного коду ДСМ-метод для автоматичного породження гіпотез для визначення тональності тексту; запрограмовано алгоритм виділення компонент електронного тексту, що базується на регулярних виразах; досліджено метод виділення іменованих сутностей з електронного текстового документу за *BIOES*-схемою; реалізовано алгоритм *SumBasic* екстрактивного методу автореферування тексту.

Наукова новизна отриманих результатів. Новизною даної дипломної роботи є застосування моделі веб-додатку маркетплейс для надання послуг з

обробки текстової інформації. Реалізація зручного, частково безкоштовного додатку, що об’єднує в собі всі найбільш необхідні функції з обробки мови та має можливість розширюватися новими методами та можливостями.

Створено новий підхід для розробки та впровадження послуг з обробки текстової інформації, що має на меті не лише заробіток, оновлений інтерфейс, зручність та легкість у використанні, але й заохочення науковців до розвитку алгоритмів та нових можливостей у сфері обробки мови людини комп’ютерними алгоритмами.

Практичне значення отриманих результатів. Розроблено комп’ютерну систему надання послуг з обробки текстової інформації. Дана система буде корисною для користувачів, оскільки представляє новітній підхід до подачі інформації та необхідного функціоналу. Дана система також полегшує розробку нових, нині ще невинайдених алгоритмів обробки текствої інформації. Оскільки система має у центрі модель маркетплейс, розробники з легкістю можуть впровадити у вже існуючу систему свій власний алгоритм та отримувати кошти за його використання, що безумовно є легшим, ніж розробка власного додатку. Дана система також є крос-платформенною, оскільки розроблена у веб-просторі з використанням веб-технологій, що доречно вписується у сучасну тенденцію використання ПО.

Особистий внесок випускника. Всі результати, представлені у дипломній роботі, отримані випускником особисто. Автором дипломної роботи було здійснено аналіз існуючих існуючими сервісів обробки текстової інформації; досліджено методи створення маркетпейсів та автоматизованих систем обробки текстової інформації; вивчено алгоритми, що використовуються у функціях для обробки текстової інформації; реалізовано алгоритми для обробки текстової інформації у вигляді автоматизованої системи сучасними мовами програмування; створено комп’ютерну систему надання послуг з обробки текстової інформації, реалізованої у вигляді маркетплейсу; проаналізувано результати та якість роботи створеної

комп’ютерної системи та сервісів з аналогічним функціоналом, зроблено висновки якості результатів роботи функцій обробки текстової інформації на ідентичних вхідних даних на користь розробленої системи.

Прогнозні припущення про розвиток об’єкту та предмету дослідження. Новітній підхід до подачі інформації та необхідного функціоналу можна застосовувати для різних напрямків обробки текстової інформації.

РОЗДІЛ 1
АНАЛІЗ ІСНУЮЧИХ СИСТЕМ НАДАННЯ ПОСЛУГ ТА ОБРОБКИ ТЕКСТОВОЇ ІНФОРМАЦІЇ

1.1. Характеристики та класифікація маркетплейсів

Платформа *eBay* – 164 мільйони активних покупців на сайті, *Uber* – сервіс таксі в 60 країнах світу. Ці прибуткові торгівельні майданчики – яскраві приклади маркетплейсів. Вони допомагають постачальникам і споживачам відшукати один одного в мережі незалежно від запиту: чи то пошук рідкої речі або виклик таксі.

Коли в 1995 році П'єр Омідьяр запустив проект *eBay*, він, сам того не знаючи, став засновником маркетплейсу. На сайті він зібрав колекціонерів і тих, хто готовий був заплатити за рідкісні речі.

Маркетплейс (*marketplace*) – торгівельна площадка в інтернеті, яка допомагає продавцям і потенційним покупцям знаходити один одного онлайн і взаємодіяти між собою.

Складові елементи маркетплейсу представлені на рисунку 1.1.

Рис. 1.1. Складові елементів маркетплейсу

Перший продаж на *eBay* – зламана лазерна указівка за 14$. Чоловік, який запропонував лот, з дитинства збирав указівки. Йому захотілося обмінятися експонатами з ким-небудь, хто мав схоже хобі. Новий сервіс відповів на запит –

дозволив колекціонерові знайти покупця. Після цього на *eBay* не раз продавали дивні речі: запрошення на весілля, яке пережило пожежу, рукопис Шекспіра, курорт в США, а один відчайдушний дивак з Каліфорнії навіть поставив на аукціон власне життя. Якими б дивними іноді не здавалися пропозиції – кожне знаходило відгук у аудиторії. Звідси і слоган компанії: «Якщо це є в тебе на думці - це є на *eBay*».

Сама ж ідея – створити умови, щоб споживачі зустрілися з постачальниками, – стала фундаментом для визначення маркетплейсу.

Така модель сайту однаково актуальна і для продавців товарів, і для тих, хто пропонує послуги. На підтвердження – приклад компанії *Uber*.

Тревіс Каланік і Геррет Кемп марно намагалися зловити таксі в Парижі, коли місто замітало снігом. Тоді їм прийшла в голову ідея: як було б зручно натиснути кнопку на мобільному телефоні і побачити порожні машини поблизу. У березні 2009-го вони реалізували задум в Америці, в 2010 – вийшли на міжнародний рівень.

Uber став одним з популярних маркетплейсів на ринку послуг – він допомагає знаходити транспорт в різних містах світу (більш 450). Користувачі відстежують маршрут руху таксі і оплачують поїздки через мобільний додаток. За прикладом цієї компанії з'явилися і інші, де можна найняти няню дитині, знайти садівника для стрижки газону або ветеринара, який вилікує собаку.

Головний фактор, за яким ідентифікують маркетплейс, – безліч вендорів на одному сайті.

Вендор (*vendor, provider*) – компанія-постачальник товарів або послуг, яка виробляє і реалізує продукцію.

За рахунок великої кількості вендорів маркетплейс підкорює користувачів різноманітністю пропозицій. У клієнта завжди є можливість вибрати більше і краще. Зростаюча кількість зацікавлених покупців, в свою чергу, робить торгівельний майданчик привабливим для нових вендорів. Щоб ця схема працювала злагоджено, маркетплейс повинен виконати 3 функції:

– познайомити споживачів з постачальниками;
– спростити для них процес комунікації, обміну товарами, послугами і оплатами;

– забезпечити нормативно-правову базу, яка ляже в основу функціонування ринку.

Набираючи популярність, маркетплейси витісняють інтернет-магазини і централізують навколо себе торгівлю в інтернеті. Приклади популярних маркетплейсів, розділених за напрямами ринку зображені на рисунку 1.2.

Рис. 1.2. Приклади популярних маркетплейсів

Створення маркетплейсу може виявитися прибутковою ініціативою для бізнесу. Питання, з якими доведеться зіткнутися на початковому етапі: як вибрати напрям, визначити потребу в послузі, розробити і запустити маркетплейс.

Зазвичай торгівельні майданчики в інтернеті класифікуються за такими типами:

– за типом учасників (*C2C*, *B2C*, *B2B*);
– за типом продуктів, що пропонуються (*goods*, *services*);
– за типом кінцевих комунікацій (*online commerce*, *O2O*);
– за способами монетизації (продаж трафіку, лидів, реклами, додаткових послуг та ін.).

Маркетплейс створюється для великої кількості постачальників і споживачів. Учасники можуть взаємодіяти на рівних (коли обидві сторони – приватні особи або представники бізнесу) або вибудовувати ієрархію «підприємець-клієнт». Існує 3 базові моделі:

– *C2C* (*customer-to-customer*) – торгують або пропонують послуги звичайні люди, не підприємці;
– *B2C* (*business-to-customer*) – бізнес здійснює продажі людям;
– *B2B* (*business-to-business*) – бізнес купує у бізнесу.

Приклади популярних торгівельних майданчиків, розділених за даною класифікацією зображено на рисунку 1.3.

Рис. 1.3. Базові моделі взаємодії «підприємець-клієнт»

За типом продуктів маркетплейси можна поділити на такі тематичні категорії:

– товари (*goods and products*);
– послуги (*services*);
– інформація (*information*);
– інвестиції і краудфандінг (*investment and crowdfunding*).

Приклади маркетплейсів за типом продуктів зображено на рисунку 1.4.

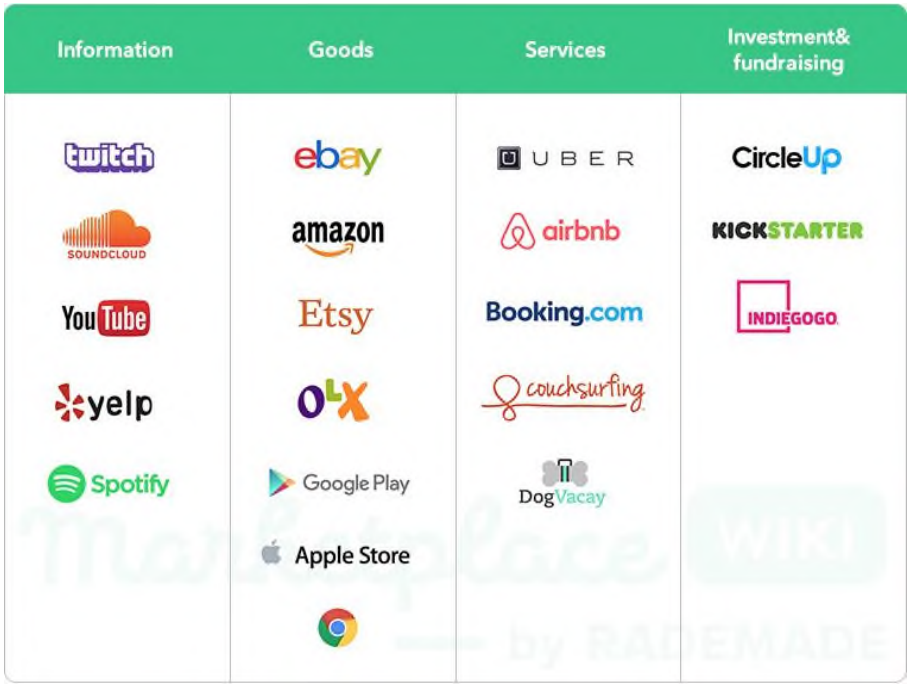


Рис. 1.4. Класифікація маркетплейсів за типом продуктів

Залежно від того, де клієнт отримує послугу (онлайн або офлайн), виділяють 2 типи маркетплейсів:

– *O2O: online to offline* – маркетплейс знаходить користувачів в мережі, а послуги користувач отримує офлайн. Наприклад, користувач вибрав товар в інтернеті, а забрав в магазині; або знайшов людину, яка погодилася вигуляти його собаку, і він приїхав до нього додому;
– *Online commerce* – все відбувається у мережі, офлайн відбувається тільки доставка товарів, але і її може не бути. Наприклад, якщо мова йде про маркетплейси з продажу квитків. Сесію можна завершити онлайн, отримавши на пошту електронні квитанції. Або дуже відомі приклади

GooglePlay і *AppleStore*: користувачі купують додатки онлайн і користуються ними теж в інтернеті.

Щоб отримати прибуток, компанія повинна монетизувати свій ресурс. Існують різні способи, які, в свою чергу, визначають види маркетплейсів. Монетизація через продаж:

– трафіку;
– лідів (заявок);
– дій;
– товарів і послуг;
– реклами;
– додаткових послуг.

1.1.1. Модель маркетплейсу *C2C*

Маркетплейси *C2C* припускають, що всі учасники рівні. Тому модель ще називають *P2P*, що має декілька тлумачень:

– *peer-to-peer* – ровесник ровеснику;
– *people-to-people* – люди людям;
– *person-to-person* (*human-to-human*) – людина людині.

Люди, які збираються на платформі, не є підприємцями. Вони схожі за інтересами, достатком, віком і заходять на сайт, щоб поділитися чимось.

Користувачі:

– домовляються на вигідних умовах – економлять;
– мають схожі інтереси – об'єднуються в співтовариства;
– обміняються ролями – сьогодні купують, завтра продають.

Ці 3 характеристики застосовні для всіх без винятку *C2C* маркетплейсів.

Торгівельні майданчики *C2C* виникають в соціальних сферах, тому що всі обожнюють економити. Вдалий приклад в цьому випадку – *ridesharing* маркетплейси, начебто сервісу *BlaBlaCar*. Він допомагає водіям знаходити

попутників і навпаки. У одних є вільне місце в машині, у інших – символічна сума, яку вони заплатять за бензин.

Економлять і ті, і інші:

– водій без пасажира здійснив би звичайний щоденний маршрут і нічого не заробив;
– пасажир заплатив би дорожче за послуги традиційних перевізників.

Про можливість заощадити такий сайт повідомляє на головній сторінці, як зображено на рисунку 1.5.

Рис. 1.5. Вигляд повідомлення про можливість заощадження

По-іншому це називається – шерінгова економіка, тобто витрати діляться навпіл. Аналогічний сервіс пошуку попутників *Lyft* обіцяє, що таким чином можна заощадити до 60% вартості поїздки.

Крім безпосередньої економії грошей, учасники *C2C* обміну отримують і інші вигоди – нові знайомства, ненудну поїздку, мобільність (адже час, маршрут і зупинки завжди можна обговорити з водієм).

Гроші – далеко не головний аргумент для користувачів *C2C* платформ. Для прикладу: сервіс *Couchsurfing* для мандрівників. Зареєстровані на ньому користувачі можуть зупинитися в будь-якому місті світу безкоштовно – місце для ночівлі і гарячу вечерю їм запропонує приймаюча сторона.

Незважаючи на колосальну економію грошей, це перевага відходить на другий план. Куди важливіше – познайомитись і поспілкуватись з місцевими жителями, поділитися досвідом подорожей. Саме тому головна сторінка онлайн-платформи не

згадує про економію, вона пропонує інші переваги: переночуй у місцевих, зустрінься з мандрівниками, як зображено на рисунку 1.6.

Рис. 1.6. Типова головна сторінка маркетплейсу моделі *C2C*

Всі *C2C* маркетплейси вибудовують спільноти за інтересами. Для *Couchsurfing* – це подорожі, для *Etsy* – речі ручної роботи, для *DogVacay* – домашні вихованці. Користувачі заходять на платформу, тому що вона співзвучна з їх захопленнями, і залишаються вірні сервісу.

Через те, що в бізнес-моделі *C2C* всі рівні, немає чіткої межі – ти пропонуєш послугу або хочеш скористатися. Учасники рівнозначні, і потрібно вміти цим користуватися.

Яскравий приклад – маркетплейс *SkillShare*, що пропонує вчитися онлайн. Користувачі переглядають відеоуроки, вибираючи тематику на свій розсуд. Якщо в якійсь сфері користувач є експертом – він легко може розмістити на сайті відеоролик.

На головній сторінці маркетплейсу є 2 лозунги:

– вивчай нову навичку кожен день;

– спробуй викладати і таким чином заробляти.

Основна сторінка сервісу зображена на рисунку 1.7.

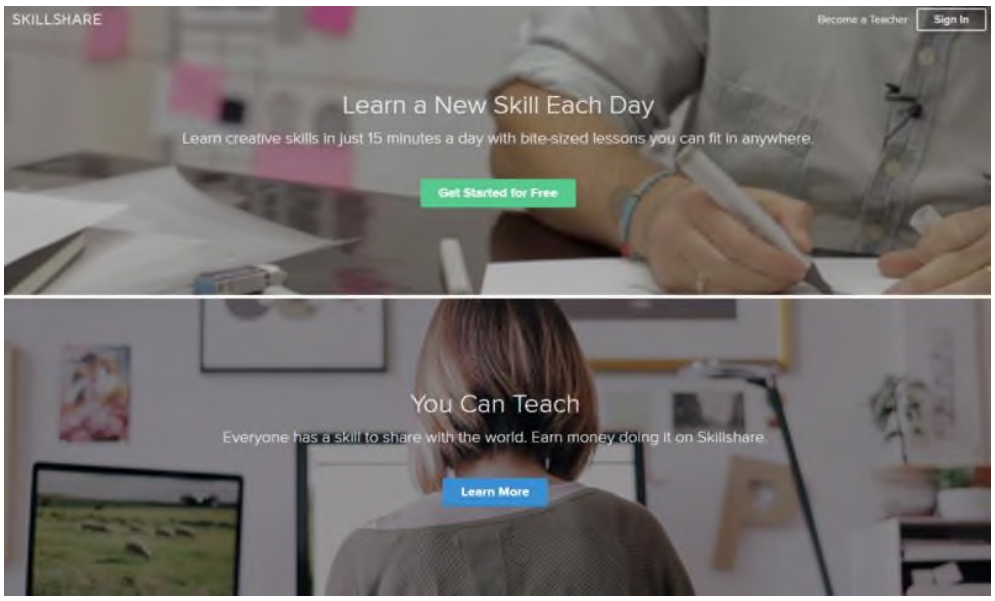


Рис. 1.7. Головна сторінка маркетплейсу *SkillShare*

Схожий підхід у маркетплейса з пошуку репетиторів *Preply*. Сервіс дозволяє вивчати по відеозв'язку іноземну мову і викладати рідну.

1.1.2. Модель маркетплейсу *B2C*

Інша маркетингова модель, *B2C* – відрізняється тим, що сюди залучений бізнес. Між собою взаємодіють підприємці та їхні клієнти. Користувачі зупиняються не у гостинних місцевих мешканців, а в хостелах і готелях *BedandBreakfast* або *Booking.com.*

Перевага таких майданчиків – великий вибір комерційних пропозицій на одному сайті. Це «*one stop shop*» – місце, де можна знайти все, що цікавить.

З моделлю *B2C* працює маркетплейс *AliExpress* – величезний торгівельний майданчик, на якій різні постачальники виставили свої товари. Платформу за місяць відвідують понад 525 млн. користувачів. Вони знаходять тут пропозиції компаній-виробників: від одягу і побутових приладів до автомобілів та мотоциклів.

На сайті кожен товар продається під власним брендом – очевидно, що сторону продавців представляє бізнес. Вигляд сторінки інтернет-майданчика зображено на рисунку 1.8.

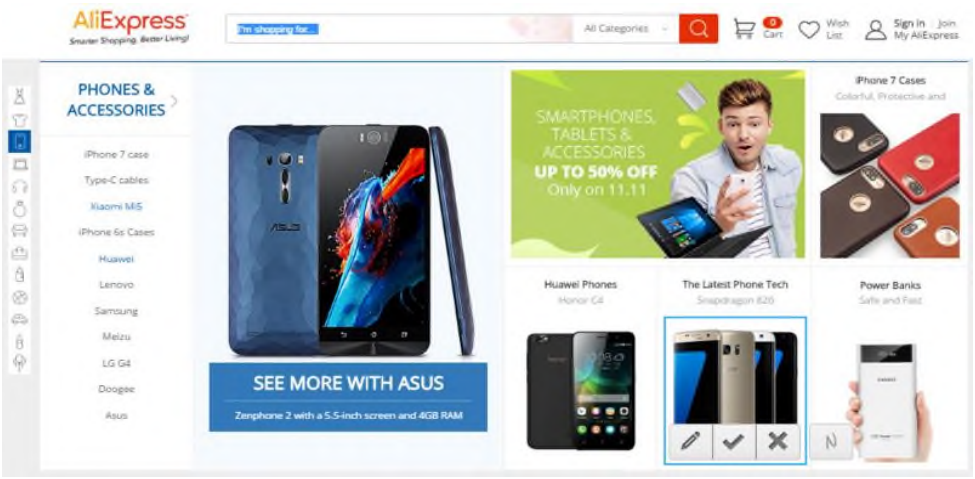


Рис. 1.8. Сторінка інтернет-майданчика *AliExpress*

Іноді бізнес-модель *B2C* стає похідною від *C2C*. Відбувається це з двох причин:

– на інтернет-майданчику *C2C* збирається багато людей і бізнесмени бачать, що утворився ринок з їх цільовою аудиторією. Вони домовляються про розміщення на майданчику своїх продуктів, і маркетплейс частково або повністю перетворюється в *B2C*;

– приватні особи (звичайні користувачі) отримують мінімальний прибуток від угод і вирішують далі розвивати свою справу як бізнес. Вони самі стають підприємцями.

Саме тому на сайті пошуку житла *Airbnb* можна зустріти не тільки приватні оголошення, а й пропозиції оренди від компаній. А на платформі *Etsy*, яка теж стартувала як абсолютний *C2C* маркетплейс, зараз товари виставляють компанії-виробники.

1.1.3. Модель маркетплейсу *B2B*

В свою чергу, дочірній проект *BookingB2B* – онлайн-технологія бронювання, що розрахована на тих, хто працює в туристичній сфері (туроператорів, агентів, перевізників та ін.). Всі користувачі платформи – малий і великий бізнес.

З одного боку знову представлені власники готелів, з іншого боку – компанії-туроператори, які хочуть заощадити на бронюванні, отримати оптові знижки. Вигляд сторінки сервісу *BookingB2B* представлено на рисунку 1.10.

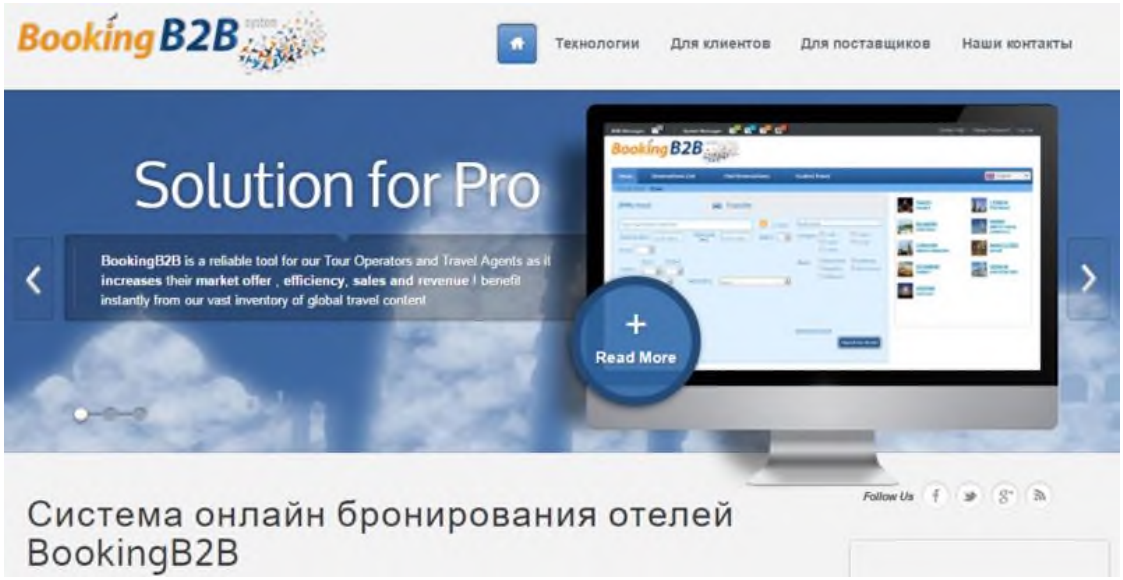


Рис. 1.10. Сторінка сервісу *BookingB2B*

Порівняння моделей маркетплейсу представлено у вигляді таблиці 1.1.

Таблиця 1.1

Порівняльна таблиця моделей маркетплейсу

Фактори	*C2C*	*B2C*	*B2B*
Вартість послуг	Десятки доларів	Десятки або сотні доларів	Тисячі або мільйони доларів
Ціни	Договірні	Ринкові	Узгоджуються разом з клієнтом, в залежності від замовлення
Тривалість процесу купівлі для користувача	Від пару годин до декількох днів	Від пари днів до декількох тижнів	Від кількох днів до декількох місяців

Закінчення табл. 1.1

Скільки людей приймає рішення про купівлю	Один або двоє	Один або двоє	Група людей, до дванадцяти чоловік
Складність процесу купівлі для користувача	Достатньо просто, обговорюється ціна, варіанти доставки та оплати	Відносно просто, також обговорюються ціна, варіанти доставки та оплати, але рідше	Складно, довготривалі перемовини з приводу ціни, обговорюється все, включаючи гарантії
Що мотивує покупців	Індивідуальні потреби або емоції	Індивідуальні потреби або емоції	Потреби бізнесу
Складність процесу реєстрації для провайдерів	Просто	Середнє	Складно
Основні задачі маркетплейсу	Познайомити з товаром або послугою; Презентувати своє рішення проблеми; Створити спільноту користувачів	Показати великий асортимент товарів або послуг; Ознайомити з брендами; Створити умови для конкуренції	Запропонувати рішення для бізнесу; Продавати великим “гравцям”; Бути першими в своїй ніші
Ніші, в яких представлені маркетплейси	Громадські блошині ринки, персональні послуги	Роздрібна торгівля	Прямі поставки, послуги для бізнесу, оптові продажі

Структура класичного маркетплейсу приведена на рисунку 1.11.

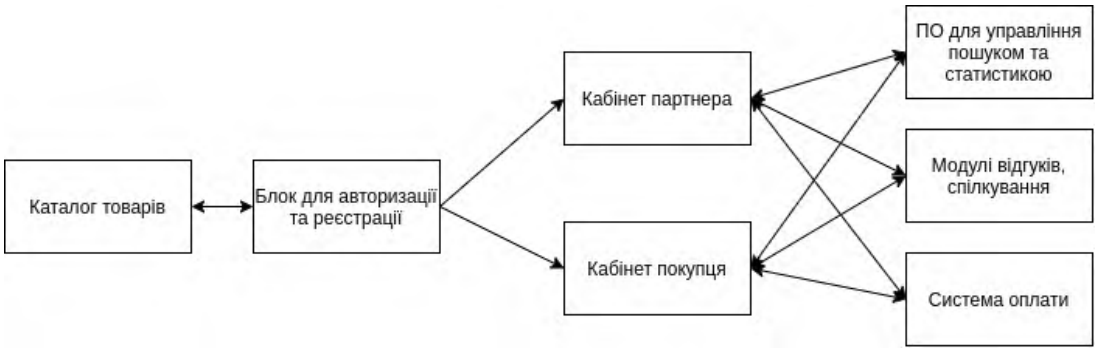


Рис. 1.11. Структура класичного маркетплейсу

– каталог товарів – схожий на вітрину звичайного інтернет-магазину, але з додаванням інформації про продукцію від постачальника;
– блок для реєстрації і авторизації покупців, а також продавців;
– кабінет партнера – в ньому продавець додає товари, стежить за обігом та отримує дані з продажу;
– кабінет покупця – працює як в онлайн-магазині;
– програмне забезпечення для управління пошуком і статистикою;
– система оплати;
– модулі відгуків, спілкування постачальників з клієнтами і врегулювання суперечок.

Особливості маркетплейсів:

– є сторонні продавці;
– один і той же товар можуть пропонувати різні фірми;
– для всіх учасників встановлені одні правила оплати і доставки;
– на торговому майданчику можуть розміщуватися товари і послуги;
– маркетплейс монетизуються за рахунок абонентської плати, комісії або відсотка від продажів.

Як вже було зазначено, інтернет-магазин пропонує продукцію лише одного постачальника. Маркетплейс, в свою чергу, нагадує біржу з різними типами товарів

від різних постачальників. В такому випадку у покупців буде великий вибір потрібних їм речей, а продавці отримають широку базу клієнтів.

У своєму інтернет-магазині власник самостійно диктує умови, весь ресурс з унікальним доменним ім'ям належить саме йому. Але і займатися створенням, просуванням, підтримкою торгівельного майданчика доведеться теж власнику.

Відмінності інтернет-магазину від маркетплейсу представлені у таблиці 1.2.

Таблиця 1.2

Таблиця відмінностей інтернет-магазину від маркетплейсу

Відмінності	Інтернет-магазин	Маркетплейс
Позиції в пошуковій видачі	Попадання наверх списку вимагає багато роботи і вкладень	Лідируючі
Географія	Складно масштабувати продажі за межі країни.	Великі маркетплейси мають налаштовані процеси міжнародної торгівлі
Обслуговування	Вимагає багато часу і сил. Проте можна підтримувати зв'язок з клієнтом за допомогою бонусних програм і акцій	Посередник ефективно обслуговує як покупця, так і продавця, формуючи додаткову вартість

Закінчення табл. 1.2

Співробітники	Для успішного запуску потрібні щонайменше програміст, веб-дизайнер і інтернет-маркетолог	Достатньо лише розмістити на торгівельному майданчику контент
Час розробки	Тиждень для шаблонного варіанту і від 2 місяців для унікального	Можна розмістити товари за один день
Які товари продавати	Залежить від безлічі факторів	Найкраще продаються дешеві і сезонні товари

Обидва формати мають свої особливості. Власний інтернет-магазин – це візитна картка, бренд. Маркетплейс – готовий інструментарій, вихід на новий ринок і розширення бази клієнтів.

Основні переваги таких сервісів для продавців:

– постійний потік цільової аудиторії за рахунок популярності торгівельного майданчика;
– можливість розширити географію продажів;
– зниження витрат на рекламу;
– швидкий запуск;
– відсутність необхідності створювати і просувати ресурс самостійно.

Серед недоліків маркетплейсів:

– висока конкуренція на майданчику;
– залежність від встановлених сервісом правил;
– менше можливостей для комунікації з потенційним клієнтом;

– проблематично підвищувати лояльність за рахунок акцій і особливих пропозицій.

1.2. Існуючі сервіси обробки текстової інформації

Існують різноманітні сервіси обробки текстової інформації, які пропонують великий спектр послуг за своєю якістю та функціональністю.

Дані сервіси реалізовані в багатьох представленнях: веб-додатки, комп'ютерні системи, надбудови браузера, програмні пакети мов програмування і т.д.

Зазвичай функції таких систем є платними і коштують дуже дорого, а дешеві або безкоштовні аналоги не дають гарних результатів та переповнені рекламою. Іншим їх недоліком є те, що дати випуску цих додатків сягають двадцяти років і більше, а дизайн систем не тільки важкозрозумілий пересічному користувачу, але й унеможливлює швидку роботу додатку, зважаючи на його застарілість.

Деякі системи з надання послуг обробки текстової інформації є настільки незрозумілими, що навіть знайти інформацію про придбання системи на офіційному сайті є неможливим, не кажучи вже про те, як встановити та налаштувати додаток. Аналізуючи сучасний ринок текстових аналізаторів постає питання: "Чому задля отримання послуг обробки тексту користувач має вишукувати годинами офіційний сайт продукту, вносити велику плату за доступ до функцій, яких він навіть не може знайти у інтерфейсі, а потім оплачувати кожне звернення до технічної підтримки? І це все у 2020 році."

Далі наведено короткий опис найпопулярніших існуючих додатків систем обробки текстової інформації.

1.2.1. *Cognitive Dwarf*

Синтаксичний аналізатор *Cognitive Dwarf* здійснює частковий синтаксичний розбір пропозицій, вирішуючи практичну задачу побудови кращої синтаксичної

інтерпретації для довільного вхідного ланцюжка слів. Вигляд робочої області програми представлений на рисунку 1.12.

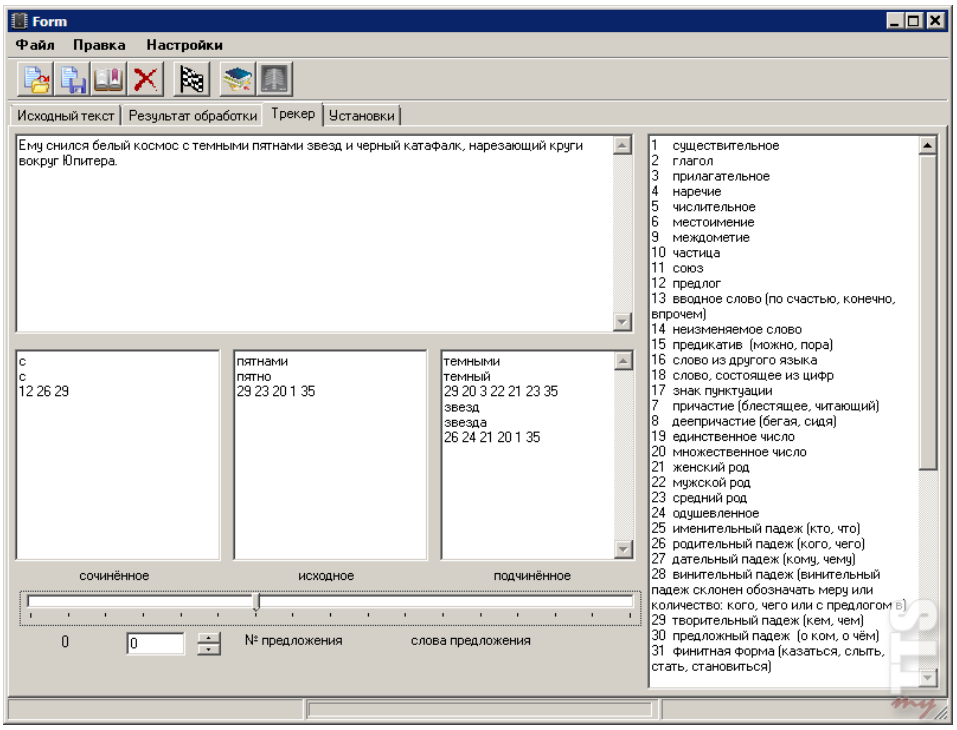


Рис. 1.12. Робоча область системи *Cognitive Dwarf*

В основі лінгвістичного опису системи *Cognitive Dwarf* лежить граматика залежностей. Дерево розбору представляється у вигляді набору бінарних спрямованих відносин залежності між словами. В кожному відношенні задаються слово-мати (головне слово) і слово-дочка (залежне слово).

Для того, щоб результуючий граф був деревом синтаксичного розбору, дотримуються наступних вимог:

– деревоподібна структура – кожне слово може мати довільну кількість слів-дочок, але тільки одне слово-мати;
– цілісність дерева – алгоритм розбору допускає розбиття вихідної речення на два і більше незалежних піддерева. Тому, щоб забезпечити цілісність результуючого дерева, перед реченням вводиться додаткове порожнє слово , яке відіграє роль слова-матері для слів, які є вершинами незв'язаних між собою піддерев;

– включення в дерево всіх слів без винятків – кожне слово в реченні повинно мати слово-мати.

Компоненти, що становлять мовну модель – лінгвістичні процесори, які один за одним обробляють вхідний текст. Вхід одного процесора є виходом іншого. Виділяються наступні компоненти:

– графематичний аналіз – виділення слів, цифрових комплексів, формул і т.д.;
– морфологічний аналіз – побудова морфологічної інтерпретації слів вхідного тексту;
– синтаксичний аналіз – побудова дерева залежностей на всі речення;
– семантичний аналіз – побудова семантичного графа тексту.

1.2.2. *TextAnalyst*

Змістовний аналізатор *TextAnalyst* – персональна система автоматичного аналізу тексту. Інтерфейс системи наведений на рисунку 1.13.

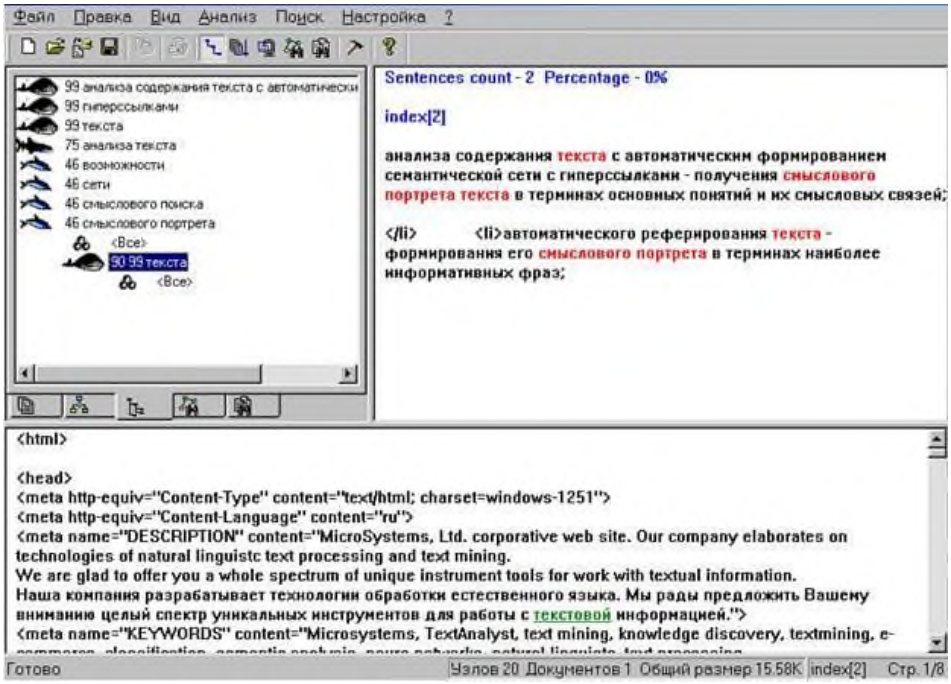


Рис. 1.13. Інтерфейс системи *TextAnalyst*

TextAnalyst розроблений як інструмент для аналізу змісту текстів, смислового пошуку інформації, формування електронних архівів, і надає користувачеві наступні основні можливості:

– аналіз змісту тексту з автоматичним формуванням семантичної мережі з гіперпосиланнями – отримання смислового портрету тексту в термінах основних понять та їх смислових зв'язків;
– аналіз змісту тексту з автоматичним формуванням тематичного древа з гіперпосиланнями – виявлення семантичної структури тексту у вигляді ієрархії тем і підтем;
– смисловий пошук з урахуванням прихованих смислових зв'язків слів запиту зі словами тексту;
– автоматичне реферування тексту – формування його смислового портрета в термінах найбільш інформативних фраз;
– кластеризація інформації – аналізу розподілу матеріалу текстів по тематичним класів;
– автоматичне індексації тексту з перетворенням в гіпертекст;
– ранжування всіх видів інформації про семантику тексту по "ступенем значущості" з можливістю варіювання детальності її дослідження;
– автоматичне формування повнотекстової бази знань з гіпертекстовою структурою і можливостями асоціативного доступу до інформації.

Семантична мережа і тематичне дерево представляються для дослідження користувачеві *TextAnalyst*, який за кожним поняттям і зв'язком бачить сенс, закладений в його моделі світу. При цьому користувач позбавляється від необхідності формування моделі тексту – за нього це робить *TextAnalyst*. Залишається тільки ознайомитися з цією моделлю.

Завершується аналіз автоматичним реферуванням. При цьому з тексту відбирається безліч фраз, що містять найбільш вагомі поняття і найсильніші зв'язки. Саме вони несуть максимальну інформацію про текст. Фрази представляються в порядку їх появи в початковому тексті. Звичайно, отриманий реферат вимагає

деякого шліфування: фрази треба підганяти один до одного, але це набагато комфортніше, ніж самому виловлювати ті ж самі фрази в тексті.

1.2.3. *IBM Intelligent Miner*

IBM Intelligent Miner – це інструментарій розробки програмного забезпечення для виявлення знань. Він містить інструменти для розробників, які хочуть створювати додатки для отримання ключової інформації з дуже великої кількості документів, електронних листів або веб-сторінок, що зберігаються в Інтернеті, без необхідності їх перечитувати. Завдяки інтелектуальному майстру для тексту можливо:

– впорядковувати документи за тематикою, знаходити переважаючі теми у колекції документів та узагальнювати їх;
– шукати відповідні документи за допомогою потужних та гнучких запитів.

Інтелектуальний майстер для тексту містить три основні компоненти:

– інструменти аналізу тексту *IBM* – інструмент ідентифікації мови, різноманітні інструменти кластеризації, інструмент категоризації тем, інструмент підбиття підсумків та засоби вилучення функцій. Ці інструменти ідентифікують мову документів, групують концептуально пов'язані документи, класифікують документи за змістом та витягують ключові елементи тексту;
– *IBM Text Search Engine* – всеосяжна пошукова система, яка налаштовується як на складний повнотекстовий пошук (включаючи функції видобутку тексту), так і на веб-налаштовані функції пошуку. Він вдосконалений за допомогою технологій *Java* та *Java Beans*, щоб допомогти створити програми для пошуку тексту та адміністративних функцій, доступних через браузер з підтримкою *Java*;
– пакет *IBM Web Crawler* – складається з готового веб-сканера та інструментарію *Web Crawler* для створення спеціалізованих веб-сканерів.

Вигляд системи зображений на рисунку 1.14.

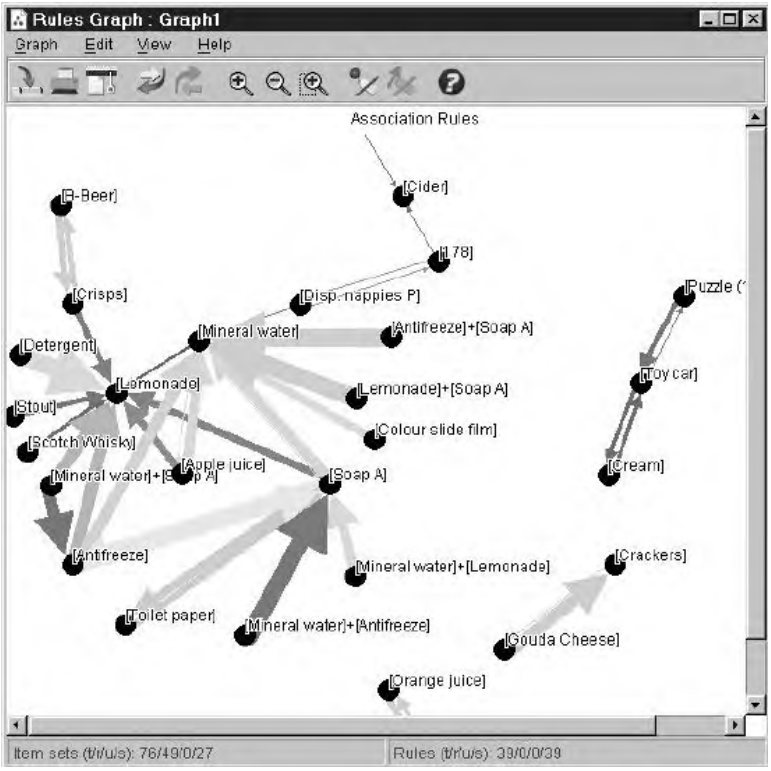


Рис. 1.14. Результат роботи системи *IBM Intelligent Miner*

Інтелектуальний інструмент *Miner* також містить рішення *IBM NetQuestion Solution* – потужне рішення для пошуку тексту в Інтернеті, для пошуку локального веб-сервера або домену мультисервера на основі текстової пошукової системи та веб-сканера.

1.2.4. *Oracle Text*

Oracle Text – використовує стандартну *SQL*-мову для індексації, пошуку та аналізу тексту та документів, що зберігаються в базі даних *Oracle*, у файлах та в Інтернеті. *Oracle Text* може виконувати лінгвістичний аналіз документів, а також пошук тексту, використовуючи різні стратегії, включаючи пошук за ключовими словами, контекстні запити, булеві операції, відповідність шаблонів, змішані тематичні запити, пошук розділів *HTML*, *XML* тощо. Він може відображати результати пошуку в різних форматах, включаючи неформатований текст, *HTML* з виділенням термінів та оригінальний формат документу.

Oracle Text підтримує декілька мов і використовує передові технології рейтингу релевантності для покращення якості пошуку. *Oracle Text* пропонує також розширені функції, такі як класифікація, кластеризація та підтримка метафор візуалізації інформації.

Якщо користувач вже використовує базу даних *Oracle*, налаштувати пошук тексту у його додатках досить легко.

Oracle Text використовує ту саму мову *SQL*, що і база даних, і безперебійно інтегрується з існуючим *SQL*. *Oracle Text* можна використовувати в будь-якій мові програмування, яка має інтерфейс *SQL*, тобто майже у всіх.

Вигляд додатку зображено на рисунку 1.15.

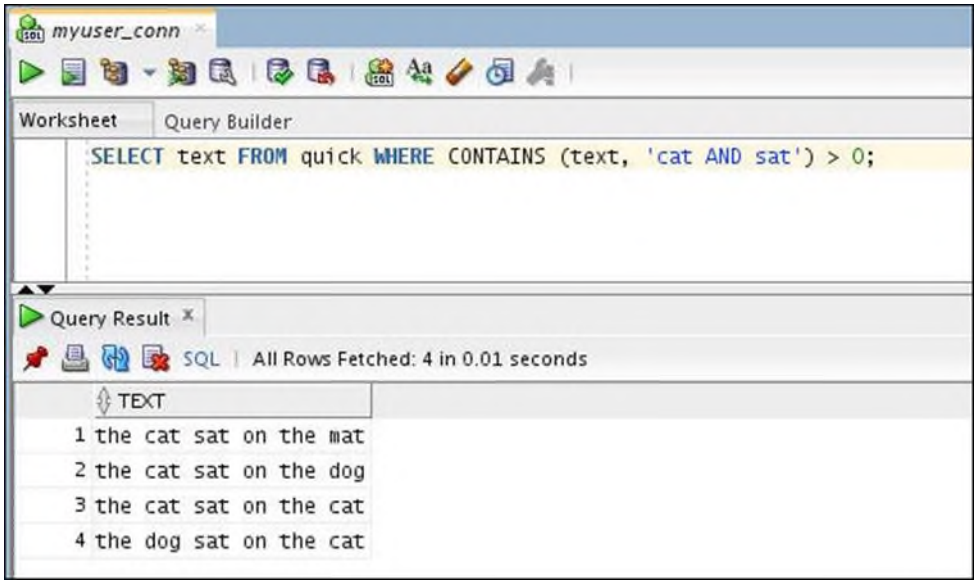


Рис. 1.15. Додаток *Oracle Text*

1.2.5. *Google PageRank*

Одним із найпопулярніших сервісів *Google* є аналітика та аналіз веб-сторінок *Google PageRank*. Надбудова для браузера *Google* – панель інструментів показує для кожної веб-сторінки ціле число від 0 до 10; Незважаючи на те, що сервіс виводить певні графіки та статистичні показники і що точно визначає значення оцінки, який саме алгоритм використовується – не розкривається. Кожен місяць *Google* оновлює

алгоритми, які істотно вдосконалюють формування видачі. На основі такої інформації користувач зможе проаналізувати стан свого сайту та виявити проблеми, через які його проект не може досягнути гарних результатів у просуванні.

1.2.6. *STATISTICA Text Miner*

STATISTICA Text Miner – це додатковий сервіс додатку *STATISTICA Data Miner*, що ідеально підходить для того, щоб перевести неструктурований текст у легко-зрозумілу, цільну інформацію, доступну для прийняття "золотих" рішень.

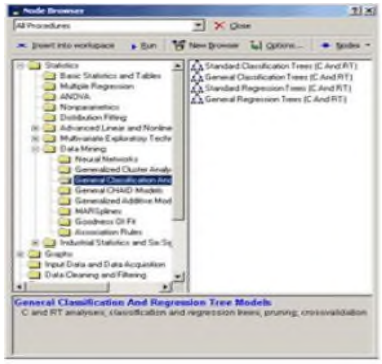
STATISTICA Text Miner дозволяє вибрати необхідні дані та створити їх. *STATISTICA Text Miner* інтегрується в додаток *STATISTICA Data Miner* і в інші продукти компанії *StatSoft.*

Наявне застосування використовує багатопотокові комп'ютерні технології для досягнення максимальної продуктивності передачі багатопроцесорних серверних систем.

Так само як і всі компоненти *STATISTICA Data Miner, STATISTICA Text Miner* спеціально розроблений, як загальнодоступний засіб з відкритою архітектурою, призначений для отримання даних з потоку неструктурованої інформації. Окремою особливістю те, що у своїх джерелах даних можна використовувати не лише текстові документи або веб-сторінки, а також посилання, списки або кластери.

Аналізуючи неструктуровану інформацію, можна навіть включити в себе бітові зображення, звукові файли та ін.

Вигляд інтерфейсу системи приведений на рисунку 1.16.



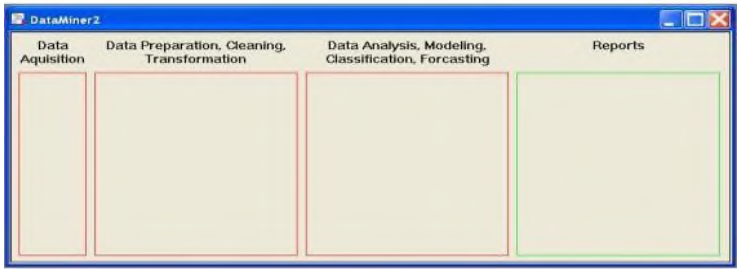


Рис. 1.16. Інтерфейс системи *STATISTICA Text Miner*

1.2.7. *ABBYY Compreno*

ABBYY Compreno – фактично не є окремим додатком, а набором алгоритмів компанії *ABBYY*, що використовують їх у своїх програмних продуктах.

На даний момент відомо про два програмних рішення, створених на базі алгоритмів *Compreno*:

– *Intelligent Search* – дозволяє з високою точністю шукати документи, грунтуючись на розумінні сенсу тексту. Завдяки алгоритмам *Compreno* воно враховує не тільки всі форми слів, але і їх семантичні значення, виявляє смислові зв'язки між словами, визначає зміст і контекст всього документа. Такий підхід дозволяє значно підвищити ефективність пошуку в порівнянні з традиційними системами повнотекстового пошуку, які шукають дані за ключовими словами;

– *Intelligent Tagger* – аналізує неструктуровану текстову інформацію і автоматично витягує з неї іменовані суті (персони, організації, дати та інші) і метадані документів. Отримані дані можна використовувати для вдосконалення і автоматизації різних бізнес-задач, таких як пошук і аналіз знань, класифікація та маршрутизація вхідної інформації, управління документацією та виявлення конфіденційних даних в ній.

Обидва рішення зараз підтримують англійську та російську мови.

Вигляд інтерфейсу додатку зображений на рисунку 1.17.

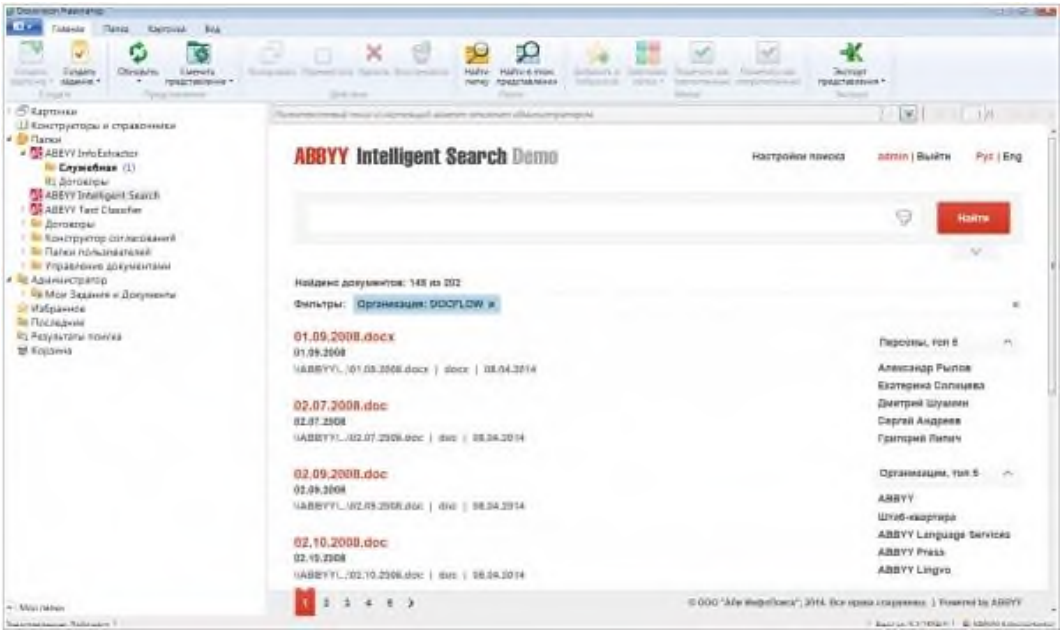


Рис. 1.17. Інтерфейс додатку *ABBYY Intelligent Search*

1.2.8. *AOTru*

Набір бібліотек, що постачається безкоштовно командою *AOT*. Немає візуального вигляду додатку, оскільки бібліотеки ставляться та імпортуються в різні програмні рішення, і можуть виглядати по-різному.

Компоненти, що становлять мовну модель, – лінгвістичні процесори, які один за одним обробляють вхідний текст. Вхід одного процесора є виходом іншого.

Виділяються наступні компоненти:

– графематичний аналіз. Виділення слів, цифрових комплексів, формул і т.д.;
– морфологічний аналіз. Побудова морфологічної інтерпретації слів вхідного тексту;
– синтаксичний аналіз. Побудова дерева залежностей за все пропозиції;
– семантичний аналіз. Побудова семантичного графа тексту.

Для кожного рівня розроблялася своя мова уявлення. Мова представлення складається з констант і правила їх комбінування. На графематичному рівні константами були графематичні дескриптори (ЛЕ – лексема, ЦК – цифровий комплекс і т.д.) На морфологічному рівні – граммеми (рд – родовий відмінок). На синтаксичному – назви відносин і груп (піпри – відношення між підметом і присудком, ПГ – прийменникова група). На семантичному – семантичні категорії і відносини.

Перелік популярних сервісів обробки текстової інформації, розділених за видами послуг представлено у таблиці 1.3.

Таблиця 1.3

Таблиця послуг та програмних сервісів, що їх надають

Послуга	Програмні сервіси, що надають дану послугу
Система порівняльного аналізу текстової інформації	*Text.ru, Advego plagiatus, AOT*
Синтаксичний аналіз	*Aot.tu, Cognitive Dwarf, ABBYY Compreno, AOT*
Класифікація	*TextAnalist, IBM Intelligent Miner for Text, Advego, Text Miner*
Автореферування	*Oracle Text*
Ранжування	Утиліти *Google*

Для того, щоб знайти серед безлічі програмних продуктів, а ще й придбати, а не просто прочитати про нього інформацію, необхідно здійснювати постійний пошук та аналіз додаткової інформації, інтернет-джерел і т.д., тобто відсутній маркетплейс, де зосереджені програмні продукти обробки текстової інформації, розбиті на категорії для зручності користувача, до того ж ніде немає опису та порівняння якості роботи цих систем, чому і буде присвячено дану дипломну роботу.

Саме тому всі подальші матеріали дипломної роботи спрямовані на розробку та реалізацію комп'ютерної системи надання послуг з обробки текстової інформації, реалізованої у вигляді маркетплейсу.

1.3. Висновки до розділу

У першому розділі дипломної роботи викладено опис поняття "маркетплейс", надано їх класифікацію, а також описано основні моделі маркетплейсів:

- *C2C* (*client-client*) – модель, що передбачає, що на майданчику кожна сторона відносин має рівні права. На таких маркетплейсах учасники зазвичай збираються по інтересам, обмінюються товарами, послугами, тощо;
- *B2C* (*business to client*) – модель, що передбачає, що одна сторона відносин, а саме постачальник товарів або послуг – є бізнесом. Зазвичай на таких майданчиках зібрані підприємці малого та середнього бізнесу, що надають послуги користувачам. Гарним прикладом маркетплейсу такої моделі в Україні є сайт *Rozetka.com.ua*;
- *B2B* (*business to business*) – модель, що передбачає, що обидві сторони відносин є бізнесом. На таких майданчиках великий та середній бізнес має можливість взаємодіяти один з одним та легко отримувати або постачати певний вид послуг або товарів.

Аналізуючи асортимент товарів, що виставлені сьогодні в сучасних популярних маркетплейсах, можна виокремити сегмент, який досі не охоплений маркетплейсами – це програмні продукти обробки текстової інформації, що у теперішній час суспільства знань та інформаційних технологій відіграють важливу роль у нашому житті.

Враховуючи викладені у даному розділі описи текстових аналізаторів, можна визначити, що вони відрізняються за типом програмного забезпечення, кількістю функцій, монетизацією системних особливостей, виконують функції пошуку тексту, синтаксичного аналізу, аналізу веб-сторінок, порівняльного аналізу текстової інформації, складність налаштування такого програмного забезпечення та напрями його діяльності.

Виходячи з цього, виникла необхідність створення легкого, дешевого та зрозумілого маркетплейсу отримання сервісів для обробки природної мови, що має на меті позбутися складності та великої вартості за користування своїх попередників, при цьому не втрачаючи функціональних можливостей.

Для розробки комп'ютерної системи надання послуг з обробки текстової інформації необхідно:

– ознайомитися з існуючими сервісами обробки інформації;
– дослідити методи створення маркетпейсів та автоматизованих систем обробки текстової інформації;
– вивчити алгоритми, що використовуються у функціях для обробки текстової інформації (визначення мови тексту, знаходження синонімів тощо);
– реалізувати алгоритми для обробки текстової інформації у вигляді автоматизованої системи мовами програмування *PHP*, *Python* та *JavaScript*;
– створити комп'ютерну систему надання послуг з обробки текстової інформації, реалізованої у вигляді маркетплейсу;
– проаналізувати результати роботи створеної комп'ютерної системи та сервісів з аналогічним функціоналом.

РОЗДІЛ 2
ФУНКЦІОНАЛЬНІ ОСОБЛИВОСТІ ТА АЛГОРИТМИ РОБОТИ КОМП’ЮТЕРНОЇ СИСТЕМИ НАДАННЯ ПОСЛУГ З ОБРОБКИ ТЕКСТОВОЇ ІНФОРМАЦІЇ

Для розробки комп'ютерної системи надання послуг з обробки текстової інформації у вигляді маркетплейсу необхідно реалізувати найпопулярніші функції з обробки тексту, описати алгоритми, за якими ці функції будуть працювати, і створити зручний та лаконічний інтерфейс користувача.

Можливість публікації власного контенту користувачами спричинила колосальне зростання даних з постійно зростаючою швидкістю. Найчастіше такі дані є неструктурованими, а їх обробка повинна відбуватися оперативно. Стало необхідним винайти нові підходи, методи та інструменти для здійснення більш ефективної обробки інформації.

З іншого боку дослідників давно хвилювали питання, пов'язані з обробкою природної мови, для організації більш зручної взаємодії між людиною і комп'ютером. Однак рішення більшості завдань даної області до певного моменту було складним через недостатню кількість вихідних даних.

Для реалізації в системі були обрані наступні функції:

– визначення мови введеного тексту;
– виділення компонент електронного текстового документу;
– визначення тональності тексту;
– вилучення знань з електронних текстових документів;
– автореферування текстів.

2.1. Інструменти для визначення мови введеного тексту

Тривіальний підхід для задачі визначення мови тексту реалізує перевірку слів, що знаходяться у повідомленні на наявність в словниках конкретної мови. Однак такий підхід недоцільний з практичної точки зору. Для роботи подібного методу

потрібна велика кількість обчислювальних потужностей і часу, а в реальних задачах необхідно здійснювати моментальну перевірку мови і витрачати на це якомога менше ресурсів.

Тому найкращим чином себе зарекомендував підхід, заснований на побудові ймовірнісної моделі за допомогою *N*-грам символів.

Такий спосіб дозволяє визначати мову з досить високою точністю і досягти оптимальних показників із співвідношенням витрат ресурсів і отриманої якості, проте має деякі недоліки.

Побудова ймовірнісної моделі на основі *N*-грам не дозволяє виділити достатньої кількості ознак з коротких текстів, що мають довжину в кілька слів. Повідомлення подібної довжини найчастіше зустрічаються в соціальних мережах і сервісах мікроблогів. Тому розпізнавання мови в текстах подібної довжини має деякі труднощі.

Процес автоматичного визначення мови неточний і принципово є ймовірнісним. Тобто завжди результат знаходиться з будь-якої ймовірністю, особливо це стосується мов, які мають дуже схожий або навіть ідентичний алфавіт, проте самі мови є різними. При цьому визначення мови залежить і від довжини рядка досліджуваного тексту – чим менше матеріалу для дослідження, тим складніше або навіть неможливим є таке визначення. Адже для статистики необхідно мати більший простір для підрахунку параметрів, а в короткому рядку майже неможливо отримати достатньо матеріалу для ідентифікації, особливо при аналізі мов, які мають у своїй основі однаковий алфавіт. У такому тексті банально може просто не зустрічатися унікальних символів і він буде визначений як текст іншої мови.

Тому першим обмеженням методу аналізу використовуваного алфавіту є довжина тексту – чим він більший, тим точніше аналіз. Для прикладу: слово "*rappel*". Дане слово є і у англійській, і у німецькій мовах. Англійською воно означає «спускатися на мотузці». Але таке ж саме слово є у німецькій мові. І там воно означає "(раптове) божевілля, напад сказу".

Цей метод має два різновиди. Перший варіант «відсотка використання алфавіту» базується на підрахунку кількості використаних унікальних символів алфавіту в тексті і розрахунку відсотку від загального обсягу. Другий різновид змінює кількість символів з тексту, яка збігається з алфавітом, при цьому деякі символи можуть потрапляти в різні алфавіти і зараховуватися до обох мов.

Другий метод заснований на використанні заздалегідь сформованих правил, які встановлюють ідентичність тексту за допомогою унікальних або типових для граматики мови послідовностей літер (наприклад, артиклі в англійській, літери "ь" і "е" в російській або "є" в українській).

Математичну модель *N*-грам можна виразити наступним чином: нехай заданий деякий скінченний алфавіт *V* – {*wi*}, де *wi* – це символ. Мовою *L*(*V*) називають множину ланцюгів скінченної довжини символів wi. Реченням називають довільний ланцюг з мови *L*(*V*). *N*-грамою у алфавіті *V* називають довільний ланцюг з мови *L*(*V*) довжиною *N*.

Нехай відомі вірогідності *p*(*wk*|*wk-N, wk-N+1...wk*-1), тобто вірогідності того, що при розборі данцюга, поточне слово – це wk, а *N* попередніми словами були *wk-N, wk-N*+1...*wk*-1. Тоді вірогідність деякого ланцюга *beg w*1...*wT end*, де *beg* та *end* – спеціальні символи початку та кінця ланцюга, можна вирахувати за формулою:

$$p(beg\ w1\ldots wTend) = p(w1|beg)p(w2|w1)..p(wT|w1..wT-1)p(end|w1..wT)$$

Правила для *N*-грам можуть розроблятися лінгвістами і дозволяють швидше і точніше визначити мову тексту, проте також не дають гарантованого результату. Їх спочатку потрібно створити, а значить володіти мовою на достатньому рівні, та й не так багато унікальних характерних послідовностей в різних мовах. Хоча, якщо заздалегідь відомо, які мови треба визначати, то між ними може бути більше унікальних поєднань, ніж якщо використовувати всі мови. Якщо вибір здійснюється тільки між російською та англійською, то таких буквосполучень явно більше, ніж в парі німецька-англійська.

Окремо слід зупинитися на випадку, коли в тексті змішуються слова різних алфавітів. Наприклад, імена або назви компаній і товарів можуть бути написані на оригінальній мові, найчастіше англійською, проте все речення сформульовано українською. Тут допоможе тільки варіант підрахунку загальної кількості символів, які належать алфавітам і на основі того, чиїх символів більше, приймати рішення.

N-грами – це *n*-літерні послідовності, отримані з тексту, що обробляється. Наприклад, слово «журналіст», розкладене в триграми (трьохбуквені послідовності) буде виглядати так: «жур», «урн», «рна», «нал», «алі», «іст». Приклад розкладання слова на *N*-грами приведено на рис.2.1.

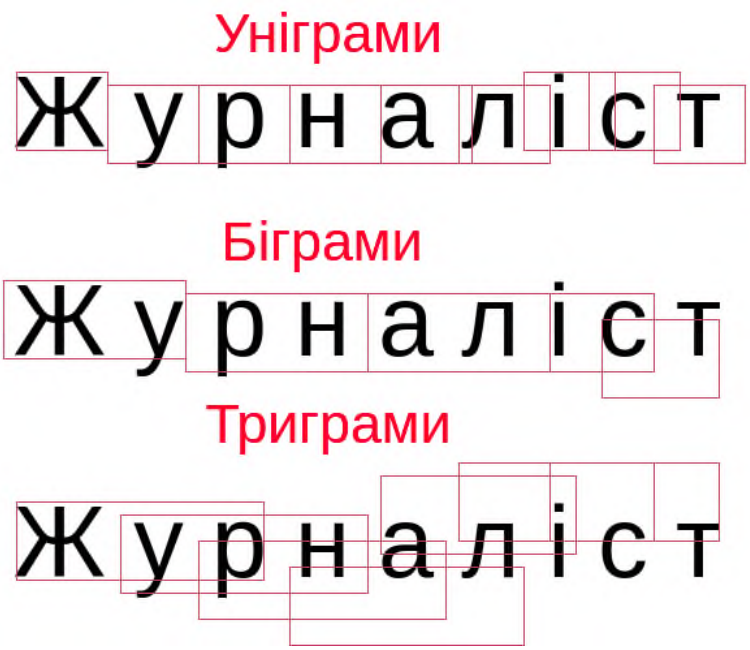


Рис. 2.1. Приклад розкладання слова на *N*-грами

Існує велика кількість методів вилучення таких послідовностей. Приклад коду, що реалізує виокремлення *N*-грам із слова, реалізований на мові програмування *PHP* наведений нижче.

```
<?php
function getNgrams($word, $n = 3) {
        $ngrams = array();
        for($i = 0; $i < strlen($match); $i++) {
                if($i > ($n - 2)) {
```

```
                $ng = '';
                for($j = $n-1; $j >= 0; $j--) {
                        $ng .= $match[$i-$j];
                }
                $ngrams[] = $ng;
        }
    }
    return $ngrams;
    }
```

Поглянувши на текст, розбитий на *N*-грами, можна помітити, що з їх допомогою досить легко визначати мову, на якому він написаний. Для цього існує безліч алгоритмів, що використовують дво- або триграми, які розраховують різні коефіцієнти «схожості», але всі вони сходяться в одному: спочатку слід побудувати статистичну модель розподілу триграм в кожній мові, а потім подивитися, якій з побудованих моделей найбільш повно відповідає заданий текст .

Один з найбільш очевидних питань – «що робити з пробілами». В системі було закладено логіку ігнорування пробілів, тобто генерація триграмів відбувається лише із слів. Також ігноруються слова довжиною менше 3-х букв. Враховується тільки частота, з якою зустрічається триграмма в даному тексті. В принципі, можна збільшити точність визначення, ввівши в алгоритм, наприклад, глобальну вагу триграми, яка показує, наскільки загальною дана триграмма є для всіх мов створеної моделі. Але і без цього метод триграм працює досить точно, особливо при невеликій кількості мов.

Для реалізації функції розпізнавання мови введеного тексту було реалізовано алгоритм «словника» з використанням *N*-грам. Алгоритм працює наступним чином:

– виокремлює структури з введеного тексту;
– перевіряє їх із словниками мов (у цих словниках зібрані спеціально «натреновані» моделі мов, що вміщують у собі найбільш популярні конструкції);

– розраховує вірогідність для кожної мови і віддає найбільш вірогідний результат;
– додатково перевіряє так звані “спеціальні символи”, що присутні тільки в одній мові, що дає стовідсотковий результат.

Функціональну схему роботи алгоритму зображено на рисунку 2.2.

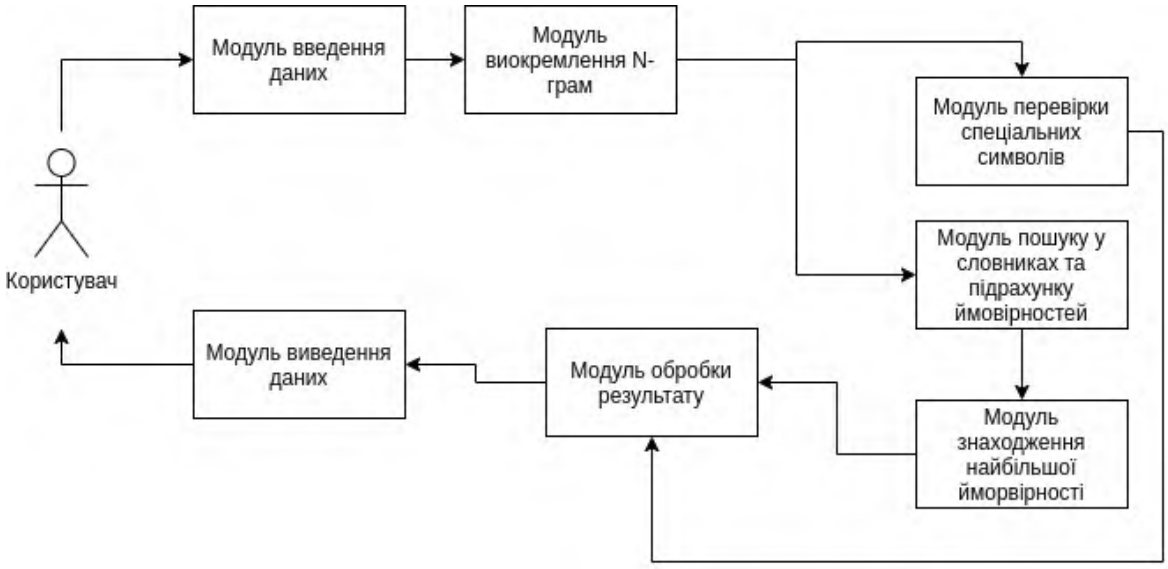


Рис. 2.2. Функціональна схема роботи алгоритму визначення мови введеного тексту

2.2. Алгоритм виділення компонент електронного текстового документу

У глобальній мережі кожен сайт, кожна сторінка будь-якого сервісу, оминаючи усі технології та фреймворки, все одно використовує мову розмітки *HTML* для відображення даних. Ця мова є єдиним стандартом і як будь-який інструмент програмування, має набір чітких правил. Стандартом для розмітки є «перетворення» будь-яких елементів сторінки (тексту, зображень) у відповідні елементи, або «теги». Тексту на сторінці зазвичай відведені найпопулярніші теги: *h*1, *h*2, ...*h*6, *p*, *span*.

При побудові систем дуже часто виникає потреба у отриманні даних з деякого сервісу, що не має функцій надсилання своїх даних. Тобто немає відкритої системи

для доступу. В такому випадку, а саме коли дані не є статичними, постає задача «вилучати» дані з публічної сторінки.

Важливим елементом веб-сторінки також є мета-дані. Це набір даних, що не відображаються в інтерфейсі, але доступні для пошукових машин у веб-просторі. Зазвичай там відображаються дані про автора сторінки, ключові слова для пошуку сторінки в пошуковому сервісі, короткий опис сторінки і т.д. Отримання таких даних є ще однією задачею виділення компонент електронного текстового документу, або «парсингу».

Парсинг сторінки реалізовано наступним чином. За допомогою інструментів мови програмування відправляється запит на сторінку, що цікавить користувача за введеним ним посиланням. У відповідь на такий запит, повертається рядок, що включає весь *HTML*-код сторінки. Існує безліч програмних пакетів, що перетворюють даний рядок у масив, де ключами є теги сторінки, а у значеннях знаходиться текст цього тегу. Таким чином, весь документ перетворюється у масив, з якого дуже легко отримати необхідну інформацію: мета-дані, ціни на продукт, прогноз погоди тощо.

Такі пакети в більшості працюють за алгоритмами, що побудовані на принципі регулярних виразів. Регулярні вирази – формальна мова пошуку і здійснення маніпуляцій з підрядками в тексті, заснована на використанні метасимволів. Для пошуку використовується рядок-зразок, що складається з символів і метасимволів і задає правило пошуку. Для маніпуляцій з текстом додатково задається рядок заміни, яка також може містити в собі спеціальні символи. Тобто, для пошуку в тексті, наприклад, лише усіх цифр, використовується правило виду: /[0-9]/*gm*.

Дані правила використовуються для пошуку в тексті тегів, та значень всередині них.

Процес виділення компонент електронного текстового документу у розробленій системі розбито на такі етапи:

– введення адреси сторінки користувачем;

– запит на дану сторінку;

– модуль парсингу сторінки;

– обробка результату та виділення компонент;

– модуль експорту сутностей;

– вивід результату користувачеві.

У розробленій системі немає чіткої цілі, які саме компоненти є найважливішими у результаті, тому зберігається вся сторінка, а користувачу повертаються лише мета-дані сторінки:

– автор;

– ключові слова;

– короткий опис сторінки.

Схему функціонування системи виділення компонент електронного текстового документу зображено на рисунку 2.3.

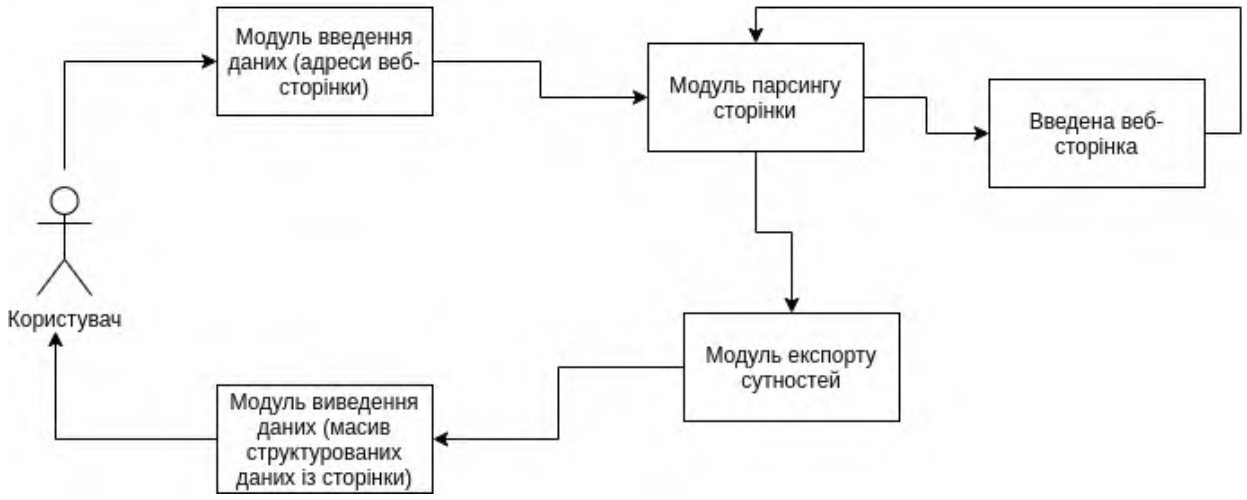


Рис. 2.3. Схема функціонування системи виділення компонент електронного текстового документу

2.3. Алгоритми визначення тональності тексту

Аналіз тональності тексту – клас методів контент-аналізу в комп'ютерній лінгвістиці, призначений для автоматизованого виявлення в текстах емоційно

забарвленої лексики і емоційної оцінки авторів по відношенню до об'єктів, мова про які йде в тексті.

Тональність – це емоційне ставлення автора висловлювання до деякого об'єкту, виражене в тексті. Емоційна складова, виражена на рівні лексеми або комунікативного фрагмента, називається лексичної тональністю. Тональність всього тексту в цілому можна визначити як функцію суми лексичних тональностей складових його одиниць.

Основною метою аналізу тональності є знаходження думок в тексті і виявлення їх властивостей. Які саме властивості будуть досліджуватися, залежить вже від поставленого завдання. Наприклад, метою аналізу може бути автор, тобто особа, якій належить думка.

Думки діляться на два типи:

– безпосереднє думку;
– порівняння;

Безпосереднє думку містить висловлювання автора про один об'єкт. Формальне визначення безпосередньо думки виглядає так: безпосередньо думкою називається кортеж з п'яти елементів (e, f, op, h, t) , де:

– (*Entity, feature*) – об'єкт тональності e (сутність, щодо якої висловлюється автор) або його властивості f (атрибути, частини об'єкта);
– *orientation* або *polarity* – тональна оцінка (емоційна позиція автора щодо згаданої теми);
– *holder* – суб'єкт тональності (автор, тобто кому належить ця думка);
– момент часу time, коли було залишено думку.

Приклади тональних оцінок:

– позитивна;
– негативна;
– нейтральна.

Під «нейтральною» мається на увазі, що текст не містить емоційного забарвлення. Також можуть існувати й інші тональні оцінки.

У сучасних системах автоматичного визначення емоційної оцінки тексту найчастіше використовується одновимірний емоційний простір: позитив чи негатив (добре чи погано).

Основним завданням в аналізі тональності є класифікація полярності даного документа, тобто визначення, чи є висловлена думка в документі або пропозиції позитивною, негативною або нейтральною.

Класифікація тональностей електронних документів наведена в табл. 2.1.

Таблиця 2.1

Види класифікації тональності

Класифікація	Опис
За бінарною шкалою	Полярність документа можна визначити по бінарній шкалі. У цьому випадку для визначення полярності документа використовується два класи оцінок: позитивна чи негативна. Одним із мінусів даного підходу є те, що емоційну складову документа не завжди можна однозначно визначити, тобто документ може містити ознаки як позитивної, так і негативної оцінки.
За багатополосною щкалою	Можна класифікувати полярність документа за багатополосною шкалою, що було розроблено Пангом і Снайдером. Ними було розширене основне завдання класифікації кіно-відгуків від оцінки «позитивний або негативний» в бік прогнозування рейтингу за трьох або чотирьох бальною шкалою. У той же час Снайдер

Закінчення табл. 2.1

	провів поглиблений аналіз оглядів ресторанів, прогнозуючи рейтинги їхніх різних властивостей, таких як їжа і атмосфера (за пяти-бальною шкалою).
Суб’єктивність / об’єктивність	Інший дослідницький напрямок – це ідентифікація суб’єктивності або об’єктивності. Це завдання зазвичай визначається як віднесення даного тексту в один з двох класів: суб’єктивний або об’єктивний. Ця проблема іноді може бути більш складною, ніж класифікація полярності: суб’єктивність слів та фраз може залежати від їх контексту, а об’єктивний документ може містити в собі суб’єктивні речення.

Для розробки функції визначення тональності введеного тексту було обрано ДСМ-метод.

Функціональна схема методу зображена на рис. 2.4.

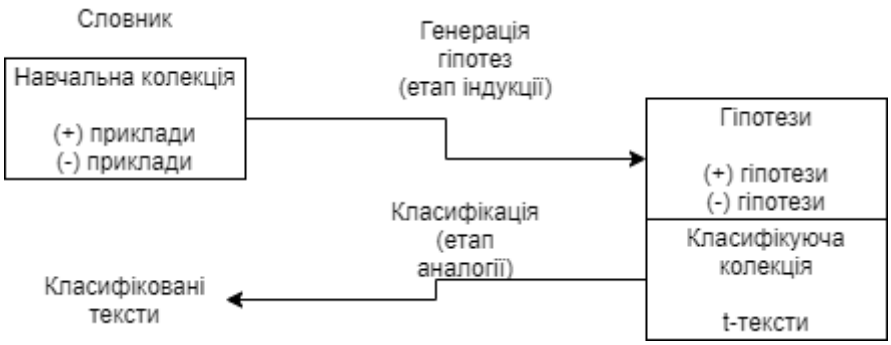


Рис. 2.4. Функціональна схема ДСМ-методу

Для реалізації ДСМ-методу використовуються три основних компоненти: словник, навчальна колекція текстів і тексти для класифікації. Словник і навчальна колекція використовуються для формування безлічі гіпотез, що характеризують приналежність тексту до певного класу. Гіпотези порівнюються з фрагментами текстів для класифікації на предмет збігу. За результатами порівняння робиться висновок про емоційне забарвлення цих текстів.

Словник може бути сформований як автоматично (містить без винятку всі слова з навчальної колекції), так і вручну (містить тільки слова, що мають ясно виражене емоційне забарвлення).

Навчальна колекція складається з текстів, тональність яких відома. Класифікуюча колекція містить тексти, тональність яких невідома і її потрібно визначити.

ДСМ-метод – це метод автоматичного породження гіпотез. Був запропонований В. К. Фіном в кінці 1970-х рр. Свою назву метод отримав від ініціалів відомого англійського філософа, логіка і економіста Джона Стюарта Мілля. ДСМ-метод являє собою формалізацію правдоподібних міркувань, яка дозволяє на основі аналізу наявних даних формувати гіпотези про те, якими властивостями можуть володіти розглянуті об'єкти. Даний метод – це синтез трьох розпізнавальних процедур – емпіричної індукції, структурної аналогії і абдукції.

Зазвичай використовуються такі умовні позначення: O – безліч об'єктів предметної області, P – безліч властивостей цих об'єктів, C – безліч характеристик об'єктів, які є можливими причинами властивостей, V – безліч істинних оцінок об'єктів.

На вхід методу подається безліч досліджуваних об'єктів і інформація про наявність або відсутність у них певних властивостей. Крім того, є ряд цільових ознак, кожна з яких розбиває вихідну безліч об'єктів на чотири непересічних підмножини:

– об'єкти, про які відомо, що вони мають дану ознаку;

– об'єкти, про які відомо, що вони не володіють даною ознакою;

– об'єкти, для яких існують аргументи як за, так і проти того, що вони мають дану ознаку;
– об'єкти, про які невідомо, мають вони цю ознаку чи ні.

У задачі визначення тональності тексту з двома емоційними категоріями множина *O* містить досліджувані тексти; множина *P* складається з одного елемента (властивості) *p*, що позначає позитивну тональність тексту (відсутність цієї властивості означає, що тональність тексту негативна); множина *C* включає характеристики, що відповідають за подання текстів, наприклад характеристика може бути окремим словом або словосполученням; безліч $V=|+1, -1,0,\tau|$, де +1 означає, що об'єкт має властивість *p*, -1 означає, що об'єкт не має властивість *p* , 0 – наявність протиріччя (тобто є аргументи як за, так і проти того, що об'єкт має властивість *p*), τ – відсутність інформації про властивості).

Множина текстів *O* складається з трьох підмножин: тексти позитивної тональності (+1), тексти негативної тональності (-1) і тексти, тональність яких потрібно визначити (τ-текст). Перші дві підмножини утворюють навчальну колекцію текстів, третя підмножина – тестову колекцію.

Ідея ДСМ-методу полягає в наступному. Спочатку складається колекція текстів, для яких точно відоме емоційне забарвлення. На основі наявної колекції проводиться навчання класифікатора. Воно полягає у формуванні гіпотез (етап індукції). Гіпотеза являє собою перетин текстів колекції. Для кожної емоційної категорії формується окрема множина гіпотез.

Наступним є етап аналогії. Сформовані гіпотези по черзі порівнюються з τ-текстом. Якщо гіпотеза міститься в оброблюваному тексті, то вона позначається будь-яким чином. Після того, як всі гіпотези перевірені на збіг з текстом, можна виділити множину помічених гіпотез. Така множина виділяється в кожній емоційної категорії. На останньому етапі залишається зробити висновок, до якого класу віднести τ-текст.

У задачі визначення тональності тексту використовується досить велика кількість характеристик об'єктів (близько 104) і породжених гіпотез (близько 104-106). Внаслідок цього відбуваються численні збіги характеристик як позитивних

гіпотез, так і негативних з τ -текстом, тобто мають місце множинні конфлікти. Для виходу з цієї ситуації використовується функція вирішення конфліктів. В якості критеріїв, що дозволяють присвоїти тональність τ-тексту, можна розглядати: сумарна кількість гіпотез (формула (2.1)), сумарна кількість характеристик у всіх гіпотезах (формула (2.2)), сумарна кількість батьків всіх гіпотез (формула (2.3)), добуток кількості характеристик на кількість батьків (формула (2.4)), зважене середнє арифметичне числа характеристик, тобто відношення добутку кількості характеристик на кількість батьків до загальної кількості батьків гіпотез одного класу (формула (2.5)), зважене середнє арифметичне числа батьків, тобто відношення добутку кількості характеристик на кількість батьків до загальної кількості характеристик гіпотез одного класу (формула (2.6)).

$$F = \sum_i h_j^+ - k \sum_j h_j^-, (h_i^+ \in H_\tau^+, h_i^- \in H_\tau^-), \tag{2.1}$$

де h_i^+ - позитивні гіпотези;

h_j^- - негативні гіпотези;

k – коефіцієнт, що враховує дисбаланс кількості позитивних і негативних текстів.

$$F = \sum_i c(h_j^+) - k \sum_j c(h_j^-), \tag{2.2}$$

де *c(h)* - кількість характеристик гіпотези *h*.

$$F = \sum_i r(h_i^+) - k \sum_j r(h_j^-), \tag{2.3}$$

де *r*(*h*) - кількість батьків гіпотези *h*.

$$F = \sum_i c(h_i^+) * r(h_i^+) - k \sum_j c(h_j^-) * r(h_j^-), \tag{2.4}$$

$$F = \frac{\sum_i c(h_i^+) * r(h_i^+)}{\sum_i r(h_i^+)} - k \frac{\sum_j c(h_j^-) * r(h_j^-)}{\sum_j r(h_j^-)}, \quad (2.5)$$

$$F = \frac{\sum_i c(h_i^+) * r(h_i^+)}{\sum_i c(h_i^+)} - k \frac{\sum_j c(h_j^-) * r(h_j^-)}{\sum_j c(h_j^-)}, \quad (2.6)$$

Значення, що повертається цією функцією, визначає категорію оброблюваного тексту.

На етапі індукції для того, щоб встановити подібності об'єктів, здійснюється пошук всіх загальних фрагментів об'єктів. Доведено, що для бінарного представлення характеристик таке завдання є *NP*-повним. Для пошуку всіх загальних фрагментів використано алгоритм Норріса, який має лінійну складність від числа загальних фрагментів, є інкрементним і одним з найефективніших серед аналогічних методів.

Було наведено опис алгоритму Норріса: нехай є набір множин (об'єктів). Вводиться та фіксується на цьому наборі множин який-небудь лінійний порядок. На *n*-ному кроці для n-ного об'єкта алгоритм доповнює набір перетинів, побудованих для попередніх *n*-1 множин, перетинами кожного безлічі цього набору з n-ним об'єктом.

Введено позначення через *n*(*m*) номер безлічі *m* (самі множини та їх перетину позначаються маленькими літерами, а підмножина номерів множин - великими літерами).

Нехай *L*(*k*) – безліч понять, отриманих при обробці перших *k* множин. Очевидно, що *L*(0) порожньо.

2.4. Методи розпізнавання іменованих сутностей в електронних текстових документах

Розпізнавання іменованої сутності (*NER*) - це підзадача вилучення інформації, яка намагається знайти та класифікувати іменовані сутності, що застосовані в неструктурованому тексті, за попередньо визначеними категоріями, такими як назви осіб, організації, місцезнаходження, медичні коди, час, грошові значення, відсотки тощо.

Більшість досліджень систем *NER* були структуровані як такі, що беруть анотований блок тексту.

Не дивлячись на те, що сутності часто бувають багатослівними, зазвичай завдання *NER* визначає задачі класифікації на рівні токенів, тобто кожен токен відноситься до одного з кількох можливих класів. Існує кілька стандартних способів зробити це, але найкращий з них називається *BIOES*-схема.

Сенс схеми полягає в тому, щоб до мети сутності (наприклад, *PER* для персоналу або *ORG* для організацій) додати певний префікс, який визначає позицію токена в строці сутності. Більш детально про схему:

- *B* – від слова початок (*begin*) – перший токен у строці сутності, який складається з більшої кількості слів;
- *I* – від слова всередині (*inside*) – це те, що знаходиться в середині;
- *E* – від слова закінчення (*ending*), це останній токен сутності, який складається більше ніж з одного елементу;
- *S* – одинарний (*single*). Ми додаємо цей префікс, якщо сутність є сутністю з одного слова.

Таким чином, до кожного типу сутності додаємо один із 4 можливих префіксів. Якщо токен не відноситься ні до якої сутності, він розміщується спеціальною міткою, як правило, ім’ям позначення *OUT* або *O*.

Для прикладу. Нехай існує текст: «Карл Фридріх Ієронім фон Мюнхгаузен народився у Боденвердері». Тут є одна багатослівна сутність – персона «Карл Фридріх Ієронім фон Мюнгхаузен» та одна однослівна – локація «Боденвердері».

Таким чином, *BIOES* – це можливість відображення проектів строк або анотацій на рівні токенів.

Зрозуміло, що за такою розміткою можна однозначно встановити межі всіх анотацій сутностей. Дійсно, про кожен токен, ми знаємо чи вірно, що сутність починається з цього токена або закінчується на нього, що означає, закінчити анотацію сутності на даному токені, або розширювати її на наступних токенах.

Переважна більшість дослідників, використовує цей спосіб (або його варіації з меншою кількістю міток – *BIOE* або *BIO*), але у нього є декількох істотних недоліків. Головний з них полягає в тому, що схема не дозволяє працювати з вкладеними або сутностями, що пересікаються. Наприклад, сутність «Міжнародний аеропорт «Київ» ім. Сікорського» – це одна організація. Але Сікорський сам по собі – це персона, і це теж необхідно навести у розмітці. За допомогою описаного вище способу розміщення неможливо передавати обидва цих фактів одночасно (тому що в одному токені можна зробити лише одну помітку). Відповідно, токен «Сікорського» може бути або частиною анотацій організації, або частиною анотацій персони, але ніколи не тим і іншим одночасно. Приклад розбиття тексту на токени зображений на рисунку 2.5.

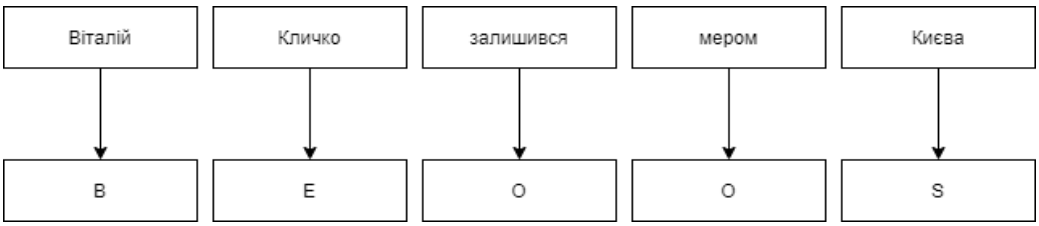


Рис. 2.5. Приклад розбиття тексту на токени

Другий приклад вкладених сутностей: «Кафедра комп’ютеризованих систем управління факультету кібербезпеки, комп’ютерної та програмної інженерії НАУ». Тут в ідеалі хотілося б виділити 3 вкладені організації, але створений вище варіант розмітки дозволяє виділити або 3 сутності, що не перетинаються, або одну сутність, що має анотацію всього наведеного фрагменту.

Крім стандартного способу зведення задачі до классифікації на рівні токенів, є і стандартний формат даних, у якому зручно зберігати розмітку для завдань *NER* (а також для багатьох інших завдань *NLP*). Цей формат називається *CoNLL-U*.

Основна ідея формату така: зберігання даних у вигляді таблиці, де одна строка відповідає одному токену, а колонки – конкретному типу ознак токена. У широкому розумінні формат *CoNLL-U* задає, які саме типи ознак включаються в таблицю – всього 10 типів ознак на кожен токен. Але дослідники зазвичай розглядають формат ширше та включають в себе ті типи ознак, які необхідні для конкретних завдань та методів їх вирішення.

Розроблювана система заснована на бібліотеці *Polyglot*. Дана бібліотека має натреновані моделі для майже 40 мов світу, для виявлення 3-ох типів сутностей:

– місця;
– організації;
– персони.

Для знаходження сутностей були виконані наступні дії:

– обробка введенного тексту користувачем;
– відправка тексту у мережу *Polyglot*, де текст поділяється на синтаксичні групи та кожній конструкції виділяється свій відповідний «токен». Вибір токену та поділ залежить від натренованої моделі та моделі даних. В даному випадку були використані відкриті моделі даних з *Wikipedia*, та модель представлення та обробки *Polyglot*;
– структуризація результату за группами токенів;
– повернення результату користувачеві.

Пошук найближчого сусіда використовується для пошуку об’єктів, схожих один на одного. Ідея полягає в тому, що з урахуванням вхідних даних пошук *NN* знаходить об’єкти в базі даних, подібні до вхідних даних. Як простий приклад, якщо у існує база даних статей новин, і необхідно отримати новини, подібні до введеного запиту, тоді виконується пошук найближчих сусідів для введеного запиту щодо всіх статей у базі даних і повертається 10 найкращих результатів.

У пошуку *NN* важлива функція відстані. Вона вирішує, наскільки подібні чи несхожі об’єкти. Більш низькі значення відстані вказують на те, що об'єкти подібні, тоді як більш високі вказують, що вони несхожі. Наприклад, якщо два об'єкти мають відстань 0, то вони однакові. Функціональна схема модуля розпізнавання іменованих сутностей зображена на рисунку 2.6.

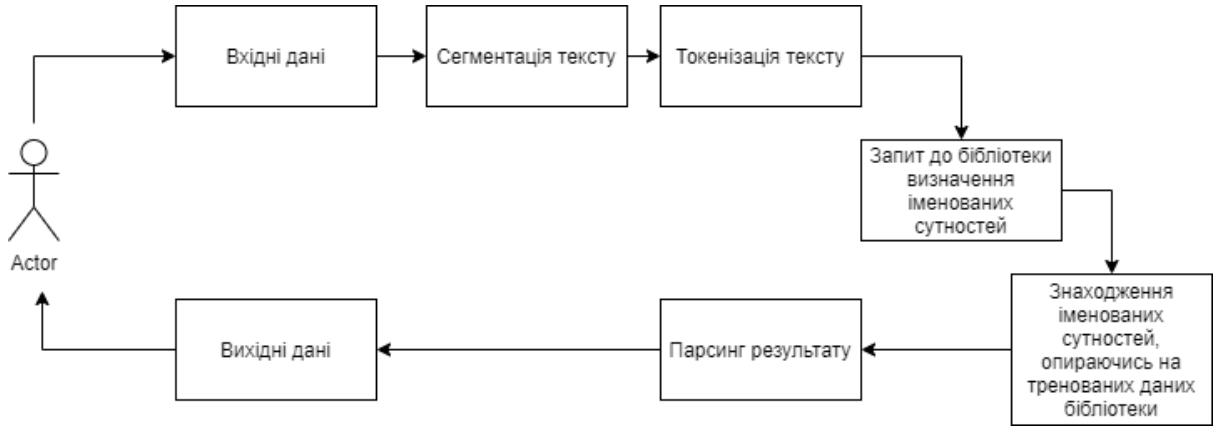


Рис. 2.6. Функціональна схема модуля розпізнавання іменованих сутностей

Для спрощення, було використано набір даних, наданий бібліотекою *scikit-learn*, яка називається “20 груп новин”. Набір даних “20 груп новин” містить близько 18000 публікацій груп новин на 20 тем. Код нижче завантажує дані.

```
from sklearn.datasets import fetch_20newsgroups
bunch = fetch_20newsgroups(remove='headers')
print(type(bunch), bunch.keys())
# (sklearn.utils.Bunch, dict_keys(['data', 'filenames', 'target_names', 'target', 'DESCR']))
```

Результат – це, в основному, об'єкт, схожий на *dict*, із ключами, наведеними вище.

Як відомо, потрібно представити сутності у векторному просторі, щоб використовувати найближчих сусідів. Оскільки наведені сутності є текстами,

необхідно використовувати певний прийом видобування ознак для вилучення векторів об’єктів. В даному випадку використовується *Tf-Idf*.

Для спрощення, було залишено налаштування за замовченням, виключаючи налаштування *max_features*. Код, що реалізує отримання ознак векторів об’єктів наведений нижче.

```
from sklearn.feature_extraction.text import TfidfVectorizer
vec = TfidfVectorizer(max_features=10_000)
features = vec.fit_transform(bunch.data)
print(features.shape) # (11314, 10000)
```

На цьому етапі уже є вектори функцій для цілого набору даних, де кожен вектор має 10000 елементів. Нарешті можна навчити модель найближчих сусідів.

Цей крок також дуже простий. Потрібно створити екземпляр об’єкта *NedestNeighbors* та навчити його, викликавши функцію *fit*. Існує декілька важливих параметрів, які, зазвичай доводиться змінювати залежно від потреб задачі:

– *n_neighbors*: кількість сусідів, які використовуються за замовчуванням;
– алгоритм: алгоритм, який використовується для обчислення найближчих сусідів: «*Ball_tree*», «*Kd_tree*», «*Force*», «*Auto*»;
– метрика: метрика для обчислення відстані. Можна використовувати будь-яку метрику від *scikit-learn* або *scipy.spatial.distance*. Якщо метрика є функцією, яку можна викликати, вона викликається у кожній парі екземплярів (рядків) і записується отримане значення.

Це все, що потрібно для навчання моделі *KNN*. Було використано косинус як метрику, оскільки він, як правило, використовується для подібності тексту.

Тепер, коли створена модель *KNN*, як знайти подібні елементи для даного вхідного тексту? Спочатку потрібно перетворити текст у вектор функції за допомогою функції *vec.transform*, а потім надати цей вектор як вхід для моделі *knn*.

Було розглянуто, який результат отримується від *KNN*. Щоб знайти найближчих сусідів, необхідно визвати функцією *kneighbours*. Перший параметр –

це список векторів функцій. Якщо *return_distance* має значення *False*, він повертає лише *2D*-масив, де кожен рядок містить k найближчих індексів сусідів для кожного вхідного вектора ознак.

Якщо *return_distance* має значення *True*, він повертає набір *2D*-масивів. У першому масиві кожен рядок містить відстані, а у другому масиві кожен рядок містить *k* найближчих індексів сусідів для кожного вхідного вектора ознак. Приклад коду даної функції наведено нижче.

```
knn.kneighbors(features[0:1], return_distance=False)
knn.kneighbors(features[0:1], return_distance=True)
```

Можна виділити, що повернені індекси є індексами векторів ознак, які використовуються під час навчання моделі *knn.fit* (ознак). Крім того, майже всі функції, такі як *fit, transform, kneighbors* тощо, у *sklearn*, очікують *2D*-масив як вхідні дані, тому було передано функції [0:1] як вхідні дані, а не просто функції [0].

Нарешті, наведений нижче код показує, як можна взяти необроблені вхідні тексти, витягти функції та знайти найближчих сусідів.

```
input_texts = ["any recommendations for good ftp sites?", "i need to clean my car"]
input_features = vec.transform(input_texts)
D,N= knn.kneighbors(input_features, n_neighbors=2, return_distance=True)
for input_text, distances, neighbors in zip(input_texts, D, N):
    print("Input text = ", input_text[:200], "\n")
    for dist, neighbor_idx in zip(distances, neighbors):
        print("Distance = ", dist, "Neighbor idx = ", neighbor_idx)
        print(bunch.data[neighbor_idx][:200])
        print("-"*200)
    print("="*200)
    print()
```

2.5. Автореферування текстів

Задача сумаризації текстів (автореферування) – одна з ключових, широко обговорюваних завдань *NLP*. Ідея заключається в тому, щоб скоротити великий об'єкт тексту до зв’язного короткого змісту, що відображає лише основні ідеї.

На даний момент людству доступна величезна кількість текстової інформації за будь-якою темою. Щоб скоротити час ознайомлення з інформацією, що представляє певний інтерес, використовують алгоритми сумаризації текстів. Їх завдання видати з потоку текстових даних головні ідеї та створити на їх основі скорочений текст для читання. Так, підведення підсумків може допомогти зрозуміти зміст тієї чи іншої наукової статті, отримати свіжі видання з новин або полегшити розуміння юридичного укладення чи фінансового звітування. Автореферування актуально практично в усіх областях, так як істотно скорочує час прочитання.

Економія часу на читанні актуальна та щорічно публікується багато статей, що описують нові методи та вдосконалення існуючих рішень. Найбільший успіх мають нейронні мережі, але є і більш прості і швидкі підходи, що використовуються в більшості статей у якості «відправної точки» для порівняння якості. Оптимального та універсального рішення завдання автоматичного реферування ще не знайдено. Нижче перелічено стандартні підходи та їх обмеження.

Всі алгоритми автоматичної сумаризації поділяються на два класи – екстрактивне узагальнення (*Extractive summarization*) та абстрактне узагальнення (*Abstractive summarization*).

Екстрактивне узагальнення – техніка, що дозволяє видавати з тексту найбільш якісно описуючі основні ідеї тексту речення, не думаючи про те, чи буде отриманий текст зв'язним та можливим для читання. Приведено опис підходів та класичних алгоритмів, що застосовуються для вирішення завдань.

Наприклад, при обробці першого параграфу статті Вікіпедії про компанію *Apple*, можна отримати такий результат: «*Apple* – американська корпорація, виробник персональних і планшетних комп'ютерів. У той же день компанія стала найдорожчою публічною компанією за всю історію.»

Абстрактне узагальнення – техніка, яка базується на виділенні основних ідей із тексту та подальшій генерації короткого змісту з нуля на основі виданих ідей.

На даний момент усі подібні алгоритми абстрактного узагальнення базуються на нейронних мережах, а це означає, що для їхнього навчання необхідна велика кількість пар текст-ручне підсумування. Не дивлячись на те, що підходи з цього класу демонструють найкращу якість та є найбільш наближеними до вирішення початкових завдань, часто нестача розміщених даних та суттєвого часу навчання роблять перепону використанню алгоритмів абстрактного узагальнення. В основному використовуються архітектури кодерів-декодерів (*encoder-decoder*), що використовують рекурентні нейронні мережі та архітектуру *Transformer*. Найбільш широко розповсюджена готова реалізація представлена в пакетах *OpenNMT-tf*.

Алгоритми екстрактивного узагальнення, як правило, досить прості у реалізації та засновані на простих евристиках. Більшість з них не вимагає жодної навчальної вибірки, що робить їх найкращими привабливими для використання на реальних даних. Частіше за все, кожному реченню із тексту, присвоюється вага «важливості», тоді вибираються кількість найбільш «важливих» речень.

При реалізації функції автореферування було використано два алгоритми: *SumBasic* і *DivRank*, які є достатньо «потужними» навіть для сумаризації декількох текстів на одну тему, але в даній роботі була зображена класична реалізація – підсумовування одного тексту.

SumBasic заснований на простому спостереженні: слова, що з'являються часто в документі, з більшою ймовірністю знайдуться в скорочених текстах, написаних людиною, ніж інші.

Алгоритм *SumBasic* складається з наступних кроків:

– для кожного унікального слова *wi* порахувати вірогідність його присутності у вхідних даних за формулою (2.7);
– зважити кожне речення *Sj*, порахувавши середню ймовірність *p*(*wi*) за формулою (2.8);
– вибрати речення з найбільшою вагою, що містить слово з максимальною ймовірністю;

– для кожного слова *wi* в обраному реченні оновити їх ймовірність за формулою (2.9);
– якщо вже обраних речень недостатньо перейти на крок 2.

$$p(wi)=\frac{n}{N}, \tag{2.7}$$

де *n* – кількість входжень слова *wi* в даних,

N – загальна кількість слів.

$$weight(Sj)=\sum(wi\in Sj)*\frac{p(wi)}{\{wi\mid wi\in Sj\mid\}}, \tag{2.8}$$

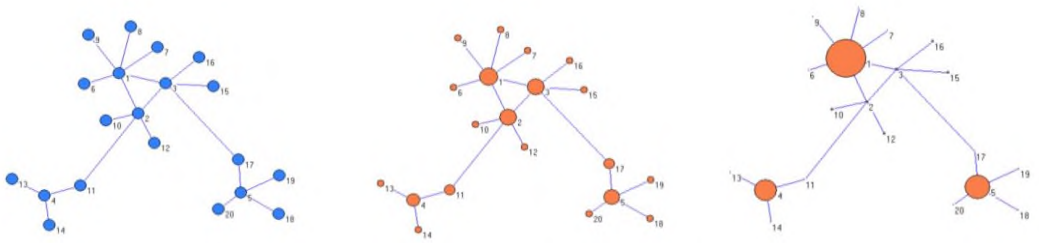
$$pnew(wi)=pold(wi)^2. \tag{2.9}$$

Крок 3 гарантує, що речення з найвірогіднішим словом буде обрано. Крок 4 додає алгоритму «чутливість» до контексту скорочення: оновлюючи ймовірності таким чином дозволяється спочатку неможливим словам чинити більший вплив на вибір речень, і, найголовніше, цей крок дозволяє ефективно застосовувати алгоритм для декількох документів, дозволяючи ігнорувати вже скорочену інформацію, реалізуючи «загасання» ймовірності слів.

SumBasic обраний через його простоту і елегантність для порівняння з більш складним алгоритмом *DivRank.*

DivRank (*Diverse Rank*) – алгоритм для зважування графів, подібний популярному *PageRank*, але застосовується для більш шикорого спектру функцій сумаризації. Для його використання необхідно побудувати граф, Вершини даного графу – це *TF-IDF* вектори речень. Ребра видаляються, якщо косинусна відстань між вершинами менше 0,1. Після зважування цього графа *DivRank* обирає *k* перших найвагоміших речень.

На рисунку 2.7, зображено «різноманітно» зважений граф за допомогою *DivRank* в порівняно з графом, зважений використовуючи *PageRank*. Якщо необхідно вибрати 3 вершини, що максимально і вмістко передають інформацію про граф, то алгоритм *DivRank* видасть вершини 1,4,5, що навіть візуально краще, ніж відповідь алгоритму 1,2,3 *PageRank*.



а) б) в)

Рис. 2.7. Зображення зваженого графу а) в початковому стані; б) за допомогою алгоритму *PageRank*; в) за допомогою алгоритму *DivRank*

Хоча в системі і присутні обидва цих алгоритми (*SumBasic* та *DivRank*), використовуватись надалі буде лише алгоритм *SumBasic*. Рішення було прийнято зважаючи на його простоту.

Функціональну схему функції автореферування тексту зображено на рисунку 2.8.

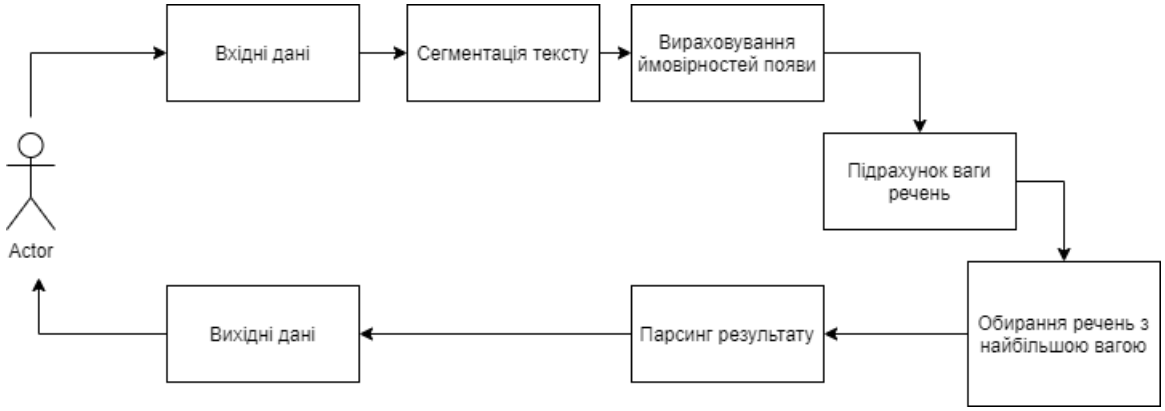


Рис. 2.8. Функціональна схема функції автореферування тексту

2.6. Висновки до розділу

В даному розділі було розглянуто функціональні особливості та алгоритми роботи комп’ютерної системи надання послуг з обробки текстової інформації.

Було досліджено інструменти для визначення мови введеного тексту, наведено схему роботи алгоритму даної функції в системі, що базується на побудові *N*-грам.

Алгоритм визначення мови введеного тексту включає наступні етапи: виокремлення структури введеного тексту; співставлення введених слів із словниками мов; розрахунок вірогідності для кожної мови, перевірка “спеціальних символів” і видача найбільш вірогідного результату.

Досліджено алгоритм виділення компонент електронного текстового документу, приведено схему алгоритму, що включає в себе введення адреси сторінки користувачем; запит на дану сторінку; модуль парсингу сторінки; обробка результату та виділення компонент; модуль експорту сутностей; виведення результату користувачеві.

Було досліджено методи аналізу тональності тексту, обрано ДСМ-метод для розробки у системі надання послуг з обробки текстової інформації, наведено функціональну схему етепів ДСМ-методу, а саме етапів індукції та аналогії.

Проаналізовано *BIOES*-схему, що лежить в основі методу розпізнавання іменовних сутностей в тексті, наведено її основні складові; наведено продемонстровано розбиття тексту на токени на конкретних прикладах; обрано три типи іменовних сутностей для розпізнавання в тексті: місця, персони, організації; розроблено функціональну схему модуля розпізнавання іменованих сутностей.

Було досліджено способи автореферування тексту алгоритмами *SumBasic* та та *DivRank*; обрано алгоритм *Sumbasic* для сумаризації тексту зважаючи на його простоту та легкість у розробці; наведено порівння результів роботи алгоритму *DivRank* та аналогічного йому – *PageRank*.

РОЗДІЛ 3
РЕЗУЛЬТАТИ ТА ЯКІСНІ ПОКАЗНИКИ ФУНКЦІОНУВАННЯ КОМП'ЮТЕРНОЇ СИСТЕМИ НАДАННЯ ПОСЛУГ З ОБРОБКИ ТЕКСТОВОЇ ІНФОРМАЦІЇ

На даний момент можна знайти безліч систем з обробки тексту в мережі. Як уже було наведено, вони відрізняються і за типом програмного забезпечення, і за кількістю реалізованих функцій, і, що найбільше – ціною. Саме тому для розробки системи надання послуг з обробки текстової інформації було взято за основу модель «маркетплейс». Всі необхідні функції з обробки тексту в одному місці, немає необхідності встановлювати додаток – він доступний в мережі інтернет. Безліч алгоритмів та можливість користувача обирати на що саме він витрачає кошти.

3.1. Архітектура комп'ютерної системи надання послуг з обробки текстової інформації

В основу системи лягла модель маркетплейсу, а саме модель *B2C* (*Business to Client*) – від бізнеса до клієнта. Це означає, що на майданчику є дві сутності: клієнт, та бізнес. Бізнес у співпраці із сервісом відповідає за функції, що виконує система: їх якість, швидкодію, необхідність користувачеві. Клієнт, в свою чергу, користується даними функціями, зазвичай небезкоштовно.

Маркетплейс має деяку власну кількість функцій (що описані у даній роботі). Ці функції надаються користувачеві безкоштовно у лімітованому обсязі, наприклад 20 безкоштовних викликів функції розпізнавання мови.

Бізнес зі своєї сторони працює із майданчиком таким чином:

– надає алгоритм або власну систему для інтеграції;

– влаштовує ціну за свої можливості у загальній системі.

Маркетплейс в свою чергу дороблює систему для того, щоб функції, що були надані постачальником були доступні для користувачів.

На даний момент система складається із таких описаних функцій:

– розпізнавання мови;
– парсинг веб-сторінки за її адресою (є можливість парсингу вставляючи код сторінки у систему);
– визначення тональності тексту;
– розпізнавання іменованих сутностей в тексті (місця, організації, персони);
– підбір синонімів до слова;
– автореферування тексту (сумаризація);
– створення графу сутностей слова за алгоритмами системи *Microsoft Concept Graph*.

Система була реалізована мовами програмування *PHP* (фреймоврк *Laravel*), *Python, Javascipt* (фреймворк *Vue.js*).

PHP – скриптова мова програмування, була створена для генерації *HTML*-сторінок на стороні веб-сервера. *PHP* є однією з найпоширеніших мов, що використовуються у сфері веб-розробок (разом із *Java*, *.NET*, *Perl*, *Python*, *Ruby*). *PHP* підтримується переважною більшістю хостинг-провайдерів. *PHP* – проект відкритого програмного забезпечення.

PHP інтерпретується веб-сервером у *HTML*-код, який передається на сторону клієнта. На відміну від скриптової мови *JavaScript*, користувач не бачить *PHP*-коду, тому що браузер отримує готовий *HTML*-код. Це є перевагою з точки зору безпеки, але погіршує інтерактивність сторінок. Розповсюдженою практикою є використання *PHP* для генерування *JavaScript*-кодів, які виконуються вже на стороні клієнта.

PHP – мова, у код якої можна вбудовувати безпосередньо *HTML*-код сторінок, які, у свою чергу, коректно оброблюватимуться *PHP*-інтерпретатором. Обробник *PHP* просто починає виконувати код після відкриваючого тегу (<?*php*) і продовжує виконання до того моменту, поки не зустріне закриваючий тег.

Велика різноманітність функцій *PHP* дає можливість уникати написання багаторядкових функцій, призначених для користувача, як це відбувається в *C* або *Pascal*.

Наявність інтерфейсів до багатьох баз даних: у *PHP* вбудовані бібліотеки для роботи з *MySQL*, *PostgreSQL*, *SQLite*, *mSQL*, *Oracle*, *dbm*, *Hyperware*, *Informix*, *InterBase*, *Sybase*. Завдяки стандарту відкритого інтерфейсу зв'язку з базами даних (англ. *Open Database Connectivity Standard*, *ODBC*) можна підключатися до всіх баз даних, до яких існує драйвер.

Нетрадиційність: мова *PHP* здаватиметься знайомою програмістам, що працюють в різних областях. Багато конструкцій мови запозичені з *C*, *Perl*. Код *PHP* дуже схожий на той, який зустрічається в типових програмах мовами *C* або *Pascal*. Це помітно знижує початкові зусилля при вивченні *PHP*. *PHP* – мова, що поєднує переваги *Perl* та *C* і спеціально спрямована на роботу в Інтернеті, мова з універсальним і зрозумілим синтаксисом. І хоча *PHP* є досить молодою мовою, вона здобула таку популярність серед *web*-програмістів, що в наш час є найпопулярнішою мовою для створення веб-застосунків (скриптів).

Наявність сирцевого коду та безкоштовність: стратегія *Open Source*, і розповсюдження початкових текстів програм в масах, безсумнівно справили сприятливий вплив на багато проектів, в першу чергу – *Linux*. Сказане відноситься і до історії створення *PHP*, оскільки підтримка користувачів зі всього світу виявилася дуже важливим чинником в розвитку проекту *PHP*. Ухвалення стратегії *Open Source* і безкоштовне розповсюдження початкових текстів *PHP* надало неоціненну послугу користувачам. Окрім цього, користувачі *PHP* в усьому світі є свого роду колективною службою підтримки, і в популярних електронних конференціях можна знайти відповіді, навіть на найскладніші питання.

Ефективність: ефективність є дуже важливим чинником у програмуванні для середовищ розрахованих на багато користувачів, до яких належить і *web*. Важливою перевагою *PHP* є те, що ця мова належить до інтерпретованих. Це дозволяє обробляти сценарії з достатньо високою швидкістю. За деякими оцінками, більшість *PHP*-сценаріїв (особливо не дуже великих розмірів) обробляються швидше за аналогічні їм програми, написані на *Perl*. Проте хоч би що робили розробники *PHP*, виконавчі файли, отримані за допомогою компіляції, працюватимуть значно

швидше – в десятки, а іноді і в сотні разів. Але продуктивність *PHP* достатня для створення цілком серйозних веб-застосунків.

Оскільки будь-яка мова програмування є лише інструментом, для розробки масштабних додатків необхідна певна архітектура. У кожної мови програмування є багато фреймворків, що сповідують ту чи іншу архітектуру. Для розробки системи було обрано фреймворк *Laravel*.

Laravel – безкоштовний, з відкритим кодом *PHP*-фреймворк, створений *Taylor Otwell* і призначений для розробки веб-додатків відповідно до шаблону архітектури *model–view–controller* (*MVC*). Особливостями *Laravel* є модульна система упакування з виділеним менеджером залежностей *Composer*, різні способи для доступу до реляційних баз даних, утиліти, які допомагають в розгортанні додатків і технічного обслуговування, а також його орієнтація на «синтаксичний цукор».

Станом на березень 2015 року, *Laravel* вважається одним з найпопулярніших *PHP* фреймворків, разом з *Symfony5*, *Nette*, *CodeIgniter*, *Yii*2 й іншими фреймворками.

MVC (модель-вигляд-контролер, англ. *model-view-controller*) – архітектурний шаблон, який використовується під час проектування та розробки програмного забезпечення.

Шаблон передбачає поділ системи на три взаємопов'язані частини: модель даних, вигляд (інтерфейс користувача) та модуль керування. Застосовується для відокремлення даних (моделі) від інтерфейсу користувача (вигляду) так, щоб зміни інтерфейсу користувача мінімально впливали на роботу з даними, а зміни в моделі даних могли здійснюватися без змін інтерфейсу користувача.

Мета шаблону – гнучкий дизайн програмного забезпечення, який повинен полегшувати подальші зміни чи розширення програм, а також надавати можливість повторного використання окремих компонентів програми. Крім того використання цього шаблону у великих системах сприяє впорядкованості їхньої структури і робить їх більш зрозумілими за рахунок зменшення складності.

У рамках архітектурного шаблону модель-вигляд-контролер (*MVC*) програма поділяється на три окремі, але взаємопов'язані частини з розподілом функцій між

компонентами. Модель (*Model*) відповідає за зберігання даних та їх структуру. Вигляд (*View*) відповідальний за представлення цих даних користувачеві, тобто інтерфейс програми. Контролер (*Controller*) керує компонентами, отримує сигнали у вигляді реакції на дії користувача (зміна положення курсора миші, натискання кнопки, ввід даних в текстове поле) і передає дані у модель.

Модель є центральним компонентом шаблону *MVC* і відображає поведінку застосунку, незалежну від інтерфейсу користувача. Модель стосується прямого керування даними, логікою та правилами застосунку.

Вигляд може являти собою будь-яке представлення інформації, одержуване на виході, наприклад графік чи діаграму. Одночасно можуть співіснувати кілька виглядів (представлень) однієї і тієї ж інформації, наприклад гістограма для керівництва компанії й таблиці для бухгалтерії.

Контролер одержує вхідні дані й перетворює їх на команди для моделі чи вигляду.

Модель інкапсулює ядро даних і основний функціонал їхньої обробки і не залежить від процесу вводу чи виводу даних.

Вигляд може мати декілька взаємопов'язаних областей, наприклад різні таблиці і поля форм, в яких відображаються дані.

У функції контролера входить відстеження визначених подій, що виникають в результаті дій користувача. Контролер дозволяє структурувати код шляхом групування пов'язаних дій в окремий клас. Наприклад у типовому *MVC*-проекті може бути користувацький контролер, що містить групу методів, пов'язаних з управлінням обліковим записом користувача, таких як реєстрація, авторизація, редагування профілю та зміна пароля.

Зареєстровані події транслюються в різні запити, що спрямовуються компонентам моделі або об'єктам, відповідальним за відображення даних. Відокремлення моделі від вигляду даних дозволяє незалежно використовувати різні компоненти для відображення інформації. Таким чином, якщо користувач через контролер внесе зміни до моделі даних, то інформація, подана одним або декількома

візуальними компонентами, буде автоматично відкоригована відповідно до змін, що відбулися.

Взаємодія моделі MVC зображена рисунку 3.1.

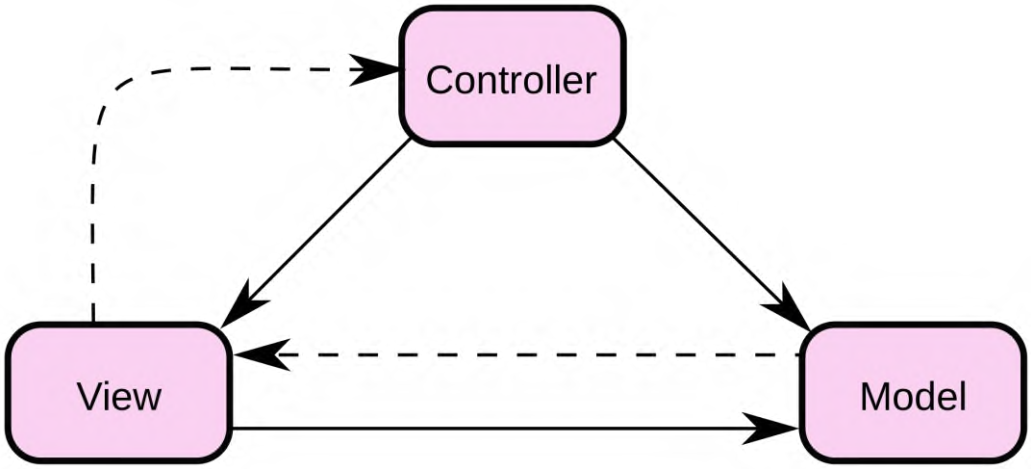


Рис. 3.1. Взаємодія моделі MVC

Оскільки мова програмування *PHP* ідеально підходить для розробки веб-додатків, але не підходить для складних математичних обчислень і т.д. виникла необхідність обрати мову програмування для обробки тексту згідно із цілями системи. Було обрано мову *Python* як ту, що найбільш зручна для складних обчислень, створення нейронних мереж та побудови імітаційних моделей.

Python (найчастіше вживане прочитання – «Пайтон») – інтерпретована об'єктно-орієнтована мова програмування високого рівня зі строгою динамічною типізацією. Структури даних високого рівня разом із динамічною семантикою та динамічним зв'язуванням роблять її привабливою для швидкої розробки програм, а також як засіб поєднування наявних компонентів. *Python* підтримує модулі та пакети модулів, що сприяє модульності та повторному використанню коду. Інтерпретатор *Python* та стандартні бібліотеки доступні як у скомпільованій, так і у вихідній формі на всіх основних платформах. В мові програмування *Python* підтримується кілька парадигм програмування, зокрема: об'єктно-орієнтована, процедурна, функціональна та аспектно-орієнтована.

Серед основних її переваг можна назвати такі:

– чистий синтаксис (для виділення блоків слід використовувати відступи);
– переносимість програм (що властиве більшості інтерпретованих мов);
– стандартний дистрибутив має велику кількість корисних модулів (включно з модулем для розробки графічного інтерфейсу);
– можливість використання *Python* в діалоговому режимі (дуже корисне для експериментування та розв'язання простих задач);
– стандартний дистрибутив має просте, але разом із тим досить потужне середовище розробки, яке зветься *IDLE* і яке написане мовою *Python*;
– зручний для розв'язання математичних проблем (має засоби роботи з комплексними числами, може оперувати з цілими числами довільної величини, у діалоговому режимі може використовуватися як потужний калькулятор);
– відкритий код (можливість редагувати його іншими користувачами).

Python має ефективні структури даних високого рівня та простий, але ефективний підхід до об'єктно-орієнтованого програмування. Елегантний синтаксис *Python*, динамічна обробка типів, а також те, що це інтерпретована мова, роблять її ідеальною для написання скриптів та швидкої розробки прикладних програм у багатьох галузях на більшості платформ.

Інтерпретатор мови *Python* може бути розширений функціями та типами даних, розробленими на *C* чи *C++* (або на іншій мові, яку можна викликати із *C*). *Python* також зручна як мова розширення для прикладних програм, що потребують подальшого налагодження.

Для відображення інтерфейсу користувача, було використано мову програмування *Javascript*, а саме фреймворк *VueJS*, що заснований на архітектурному принципі *SPA* (односторінковий додаток, англ.: *single-page-application*).

JavaScript (*JS*) – динамічна, об'єктно-орієнтована прототипна мова програмування. Реалізація стандарту *ECMAScript*. Найчастіше використовується для створення сценаріїв веб-сторінок, що надає можливість на боці клієнта (пристрої

кінцевого користувача) взаємодіяти з користувачем, керувати браузером, асинхронно обмінюватися даними з сервером, змінювати структуру та зовнішній вигляд веб-сторінки.

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.

Мова *JavaScript* використовується для:

– написання сценаріїв веб-сторінок для надання їм інтерактивності;
– створення односторінкових веб-застосунків (*React, AngularJS, Vue.js*);
– програмування на боці сервера (*Node.js*);
– стаціонарних застосунків (*Electron, NW.js*);
– мобільних застосунків (*React Native, Cordova*);
– сценаріїв в прикладних програмах (наприклад, в програмах зі складу *Adobe Creative Suite* чи *Apache JMeter*);
– всередині *PDF*-документів тощо.

Незважаючи на схожість назв, мови *Java* та *JavaScript* є двома різними мовами, що мають відмінну семантику, хоча й мають схожі риси в стандартних бібліотеках та правилах іменування. Синтаксис обох мов отриманий «у спадок» від мови *C*, але семантика та дизайн *JavaScript* є результатом впливу мов *Self* та *Scheme*.

JavaScript, наразі, є однією з найпопулярніших мов програмування в інтернеті.

Vue.js (читається як "в'ю", з англ. *view*) – *JavaScript*-фреймворк що використовує шаблон *SPA* для створення інтерфейсів користувача на основі моделей даних, через реактивне зв'язування даних.

Vue використовує синтаксис шаблонів на основі *HTML*, що дозволяє декларативно зв'язувати рендеринг дерева компонентів сторінки з основними екземплярами даних в *Vue*. Всі *Vue* шаблони є валідними *HTML*-кодами, і можуть бути розпарсені браузерами та *HTML*-парсерами. Всередині, *Vue* компілює шаблони

в рендерингові функції віртуального дерева компонентів сторінки. В поєднанні з реактивною системою, *Vue* здатний розумно обчислити кількість компонентів для ре-рендингу та застосувати мінімальну кількість маніпуляцій з деревом, коли стан застосунку зміниться.

Одна із найвиразніших особливостей *Vue* – це ненав'язлива реактивна система. Моделі це просто плоскі *JavaScript* об'єкти. Це робить керування станами дуже простим та інтуїтивним. *Vue* надає оптимізований ре-рендеринг без потреби робити що-небудь додатково. Кожен компонент слідкує за своїми реактивними залежностями під час рендерингу, тому система знає точно коли має відбуватись ре-рендеринг і які компоненти потрібно ре-рендерити.

Односторінковий застосунок (англ. *single-page application, SPA*) – це веб-застосунок чи веб-сайт, який вміщується на одній сторінці з метою забезпечити користувачу досвід близький до користування настільною програмою.

В односторінковому застосунку весь необхідний код – *HTML, JavaScript*, та *CSS* – завантажується разом зі сторінкою, або динамічно довантажується за потребою, зазвичай у відповідь на дії користувача. Сторінка не оновлюється і не перенаправляє користувача до іншої сторінки у процесі роботи з нею. Взаємодія з односторінковим застосунком часто включає в себе динамічний зв'язок з веб-сервером.

Для збереження даних, сесій користувачів, результатів виконання функцій та багато іншого було використано реляційну базу даних *Oracle MySQL*. Також, вважаючи складність деяких операцій в системі, та для того, аби вдосконалити швидкодію продукту, було використано сховище даних *Redis*.

MySQL – вільна система керування реляційними базами даних.

MySQL був розроблений компанією «*TcX*» для підвищення швидкодії обробки великих баз даних. Ця система керування базами даних (СУБД) з відкритим кодом була створена як альтернатива комерційним системам. *MySQL* з самого початку була дуже схожою на *msSQL*, проте з часом вона все розширювалася і зараз *MySQL* – одна з найпоширеніших систем керування базами даних. Вона використовується, в першу

чергу, для створення динамічних веб-сторінок, оскільки має чудову підтримку з боку різноманітних мов програмування.

MySQL – компактний багатопотоковий сервер баз даних. Характеризується високою швидкістю, стійкістю і простотою використання.

MySQL вважається гарним рішенням для малих і середніх застосувань. Сирцеві коди сервера компілюються на багатьох платформах. Найповніше можливості сервера виявляються в *UNIX*-системах, де є підтримка багатопоточності, що підвищує продуктивність системи в цілому.

Можливості сервера *MySQL*:

– простота у встановленні та використанні;
– підтримується необмежена кількість користувачів, що одночасно працюють із БД;
– кількість рядків у таблицях може досягати 50 млн;
– висока швидкість виконання команд;
– наявність простої і ефективної системи безпеки.

Redis – розподілене сховище пар ключ-значення, які зберігаються в оперативній пам'яті, з можливістю забезпечувати довговічність зберігання за бажанням користувача. Це програмне забезпечення з відкритим сирцевим кодом написане на *ANSI C*. Розробка *Redis* фінансується *VMware*.

Redis надає схожі на *Memcached* функції для зберігання даних в форматі ключ/значення, розширені підтримкою структурованих даних, таких як списки, хеші і множини. На відміну від *Memcached*, *Redis* забезпечує постійне зберігання даних на диску і гарантує збереження БД у разі аварійного завершення роботи. Клієнтські бібліотеки доступні для більшості популярних мов, включаючи *Perl*, *Python*, *PHP*, *Java*, *Ruby* і *Tcl*.

Є підтримка транзакцій, що дозволяють виконати за один крок групу команд, гарантуючи несуперечність і послідовність (команди від інших запитів не можуть вклинитися) виконання заданого набору команд, а в разі проблем дозволяючи відкотити зміни. Всі дані у повному обсязі кешуються в оперативній пам'яті. Зберігання всіх даних в оперативній пам'яті дозволяє досягнути значної

продуктивності: при тестуванні *Redis* на сервері з *CPU Xeon X*3320 2.5 *ГГц* вдалося забезпечити 110000 операцій запису і 81000 операцій читання за секунду.

Для управління даними підтримуються такі команди, як інкремент/декремент, стандартні операції над списками і множинами (об'єднання, перетин), перейменування ключів, множинні вибірки та функції сортування. Підтримується два режими зберігання: періодична синхронізація даних на диск і ведення на диску логу змін. У другому випадку гарантується повне збереження всіх змін. Можлива організація *master-slave* реплікації даних на кілька серверів, здійснювана в неблокуючому режимі. Доступний також режим обміну повідомленнями «публікація/підписка», при якому створюється канал, повідомлення з якого поширюються клієнтам за передплатою.

На зовнішньому рівні абстракції, модель даних в *Redis* це асоціативний масив в якому ключі відображаються в значення. Основною відмінністю між *Redis* та іншими базами такого типу в тому, що значення словника не обмежені рядковими типами.

Оскільки поняття маркетплейсу передбачає оплату користувачами деяких послуг в системі, виникла необхідність отримувати платежі онлайн. Саме для цього в систему був інтегрований сервіс *LiqPay*.

LiqPay – українська платіжна система, відкритий веб-застосунок, який дозволяє приймати платежі і переказувати гроші за допомогою мобільного телефону, Інтернету і платіжних карток у всьому світі. Материнська компанія – ПриватБанк.

Система *LiqPay* була створена в 2008 році. Виступає альтернативою *WebMoney* і *PayPal*.

Система *LiqPay* дозволяє:

– створювати мікроплатежі (платежі від 0,02 у.о.);
– створювати масові платежі;
– створювати миттєві перекази між рахунками *LiqPay*;
– приймати платежі на сайті;

– виводити кошти на картки системи *VISA* або будь-яку картку Приватбанку (*VISA/MASTERCARD*);
– проводити оплату послуг;
– поповнювати рахунок мобільного телефону;
– поповнювати рахунок *Skype*;
– здійснювати обмін валют між рахунками *LiqPay*;
– створювати платформи *API*;
– переводити в готівку чеки *Google*.

Архітектура комп'ютерної системи надання послуг з обробки текстової інформації зображений на рисунку 3.2.

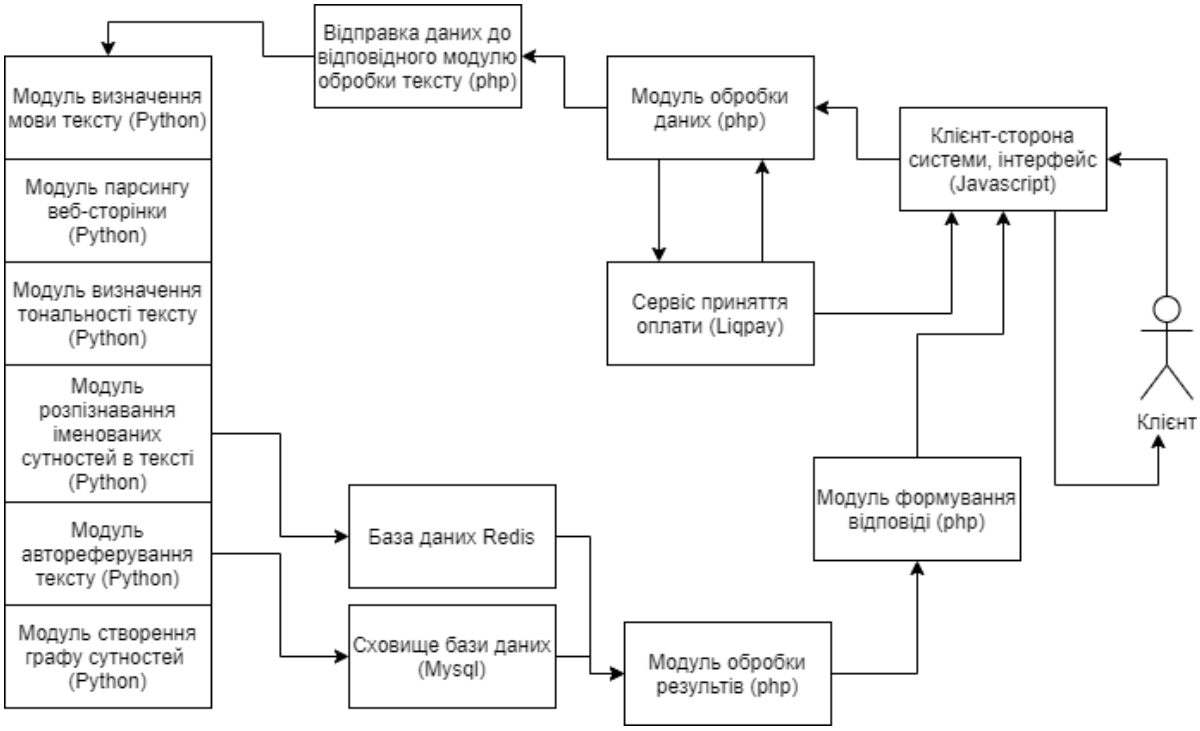


Рис. 3.2. Архітектура комп'ютерної системи надання послуг з обробки текстової інформації

Як видно з рисунку, архітектура системи складається з таких компонент:
– інтерфейс – частина системи, призначена для виводу та вводу даних, відображення результату та іншої взаємодії. Розроблена мовою програмування *Javascript* (фреймоворк *Vue.js*) за технологією *SPA*;

– модуль обробки даних – обробляє вхідні дані користувача на сервері на предмет правильності введених даних, чи зареєстрований користувач або чи є в нього оплата даної функції (якщо необхідно);
– сервіс прийняття оплати – проводить операції з оплатою від користувача, якщо це необхідно. Вся логіка банківських операцій виконується на сервері сторонньої компанії (*Liqpay*), з якою сервер системи спілкується по інтерфейсу *API*;
– відправка даних до відповідного модулю обробки тексту – програмний модуль, що обирає, яку функцію необхідно викликати для обробки тексту, тобто здійснює роутинг;
– модуль визначення тексту – модуль, шо створений за допомогою мови програмування *Python*, визначає мову введеного тексту користувачем за методом словника, з використанням *N*-грам;
– модуль парсингу веб-сторінки – проводить виділення компонент з веб-сторінки, використовується *Python*-бібліотека *Polyglot*;
– модуль визначення тональності тексту – визначає емоційне забарвлення тексту, за шкалою від -1 до +1, з використанням реалізації ДСМ-методу в стандартній бібліотеці *Python*;
– модуль розпізнавання іменованих сутностей – вибір з тексту сутностей за класифікаціями (місце, персона, організація). Реалізована схема *BIOES*, що доступна до використання в *Python*-бібліотеці *Polyglot*;
– модуль автореферування тексту – сумаризація тексту за екстрактивним алгоритмом *SumBasic*;
– сховище бази даних *Mysql* – реляційна база даних для збереження інформації про результати роботи, дані користувача, оплату та ін.;
– база даних *Redis* – база даних для збереження «великих даних» з метою пришвидшення роботи системи. Відмінність *Redis* від *Mysql* є в тому, що дані зберігається в оперативній пам’яті, що значно пришвидшує генерацію результату;

– модуль обробки результатів – перетворює дані, що були отримані під час операцій з текстом від бібліотек до стандартизованого вигляду в системі;
– модуль формування відповіді – в залежності від вибраного алгоритму, формує дані для результату, що буде відправлений до інтерфейсу, що відображається користувачу.

3.2. Аналіз роботи комп'ютерної системи надання послуг з обробки текстової інформації

Одним із правил побудови успішного маркетплейсу є розташування доступу даних у системі таким чином, щоб користувачу було легко зрозуміти сенс системи та де які компоненти знаходяться. Тому робота клієнта із системою починається із головної сторінки.

На головній сторінці розміщена інформація про систему, її можливості, та умови використання. Також повинні бути чітко видні посилання на основні функції додатку, або меню. Заохочувальна інформація, що створена для того, щоб клієнт придбав можливості системи тощо також повинна бути зображена на головній сторінці.

Вигляд головної сторінки системи зображений на рисунку 3.3.

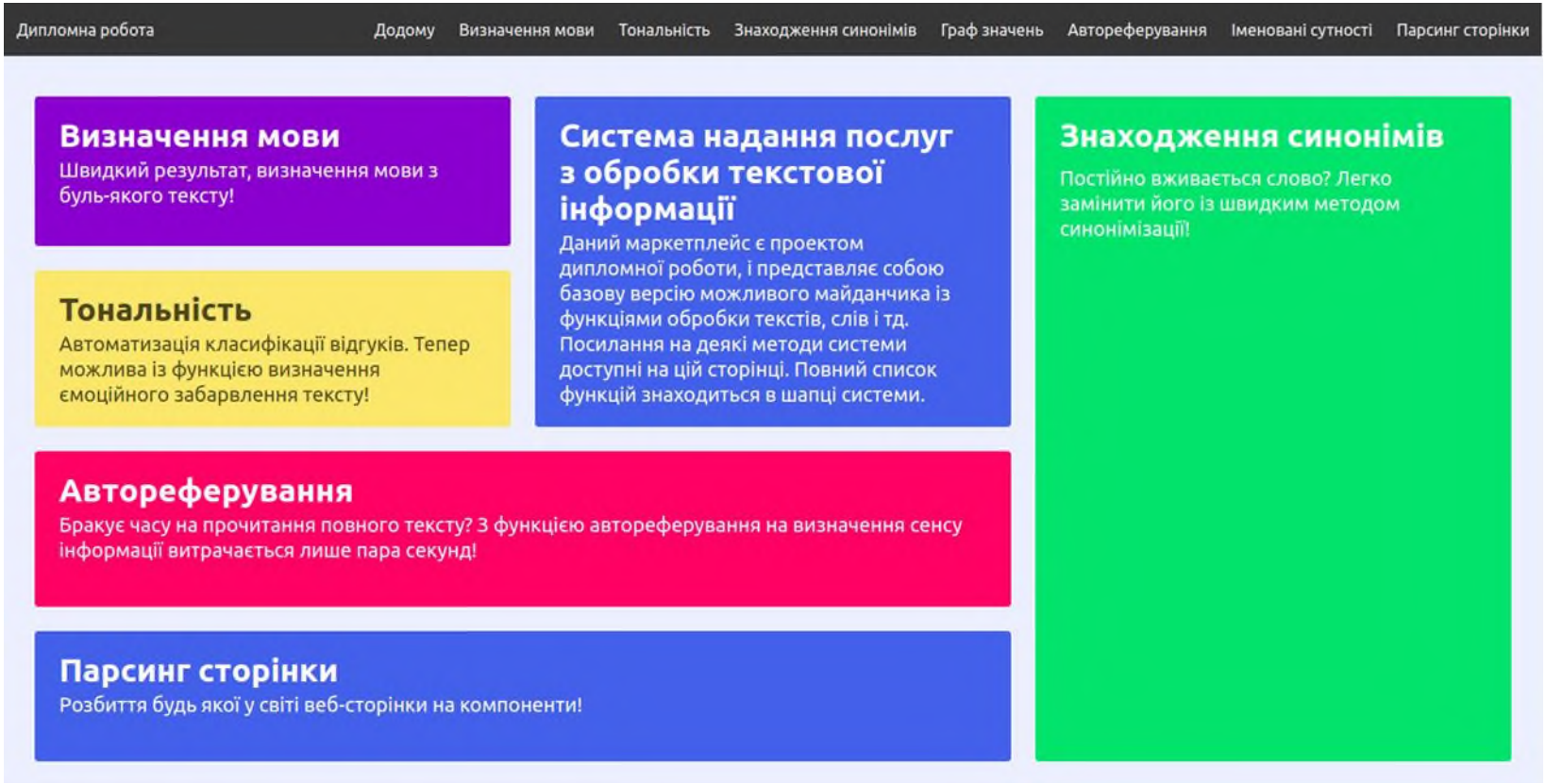


Рис. 3.3. Зображення головної сторінки системи

В системі присутнє меню, в якому знаходяться посилання на усі можливості системи. Клікнувши на певну функцію в меню, інтерфейс змінюється, і користувачу стає доступна можливість запуску необхідного компоненту додатку. Вигляд меню системи зображений на рисунку 3.4.

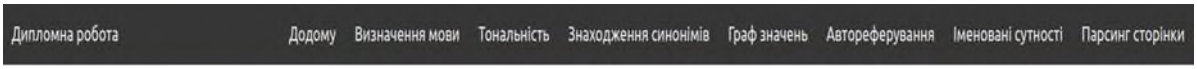


Рис. 3.4. Вигляд меню додатку

Після переходу на компонент визначення мови, користувачу доступна форма для введення тексту, кнопка «визначити» та інформаційний блок із поясненням елементу системи. Інтерфейс сторінки компонента визначення мови зображений на рисунку 3.5.

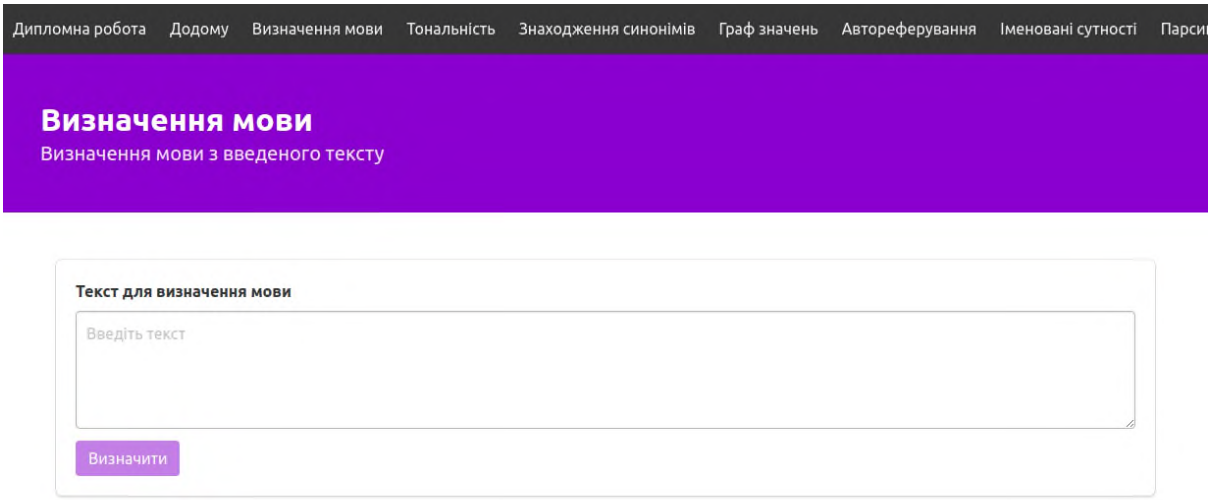


Рис. 3.5. Вигляд сторінки компонента визначення мови

Кнопка відправлення тексту на обробку є недоступною доти, доки клієнт не введе в поле хоча б три символи. Після введення, клієнт натискає кнопку і запит відправляється на сервер. Оскільки система працює швидко, немає необхідності відображати інформацію користувачу про необхідність «почекати». Після обробки запиту на сторінці відображається результат роботи системи. Над введеним текстом з’являється блок із мовою тексту, відображеною на англійській та мові оригіналу.

Також у правому верхньому куті «вилітає» повідомлення про успішне завершення роботи функції. Результат роботи компонента зображений на рисунку 3.6.

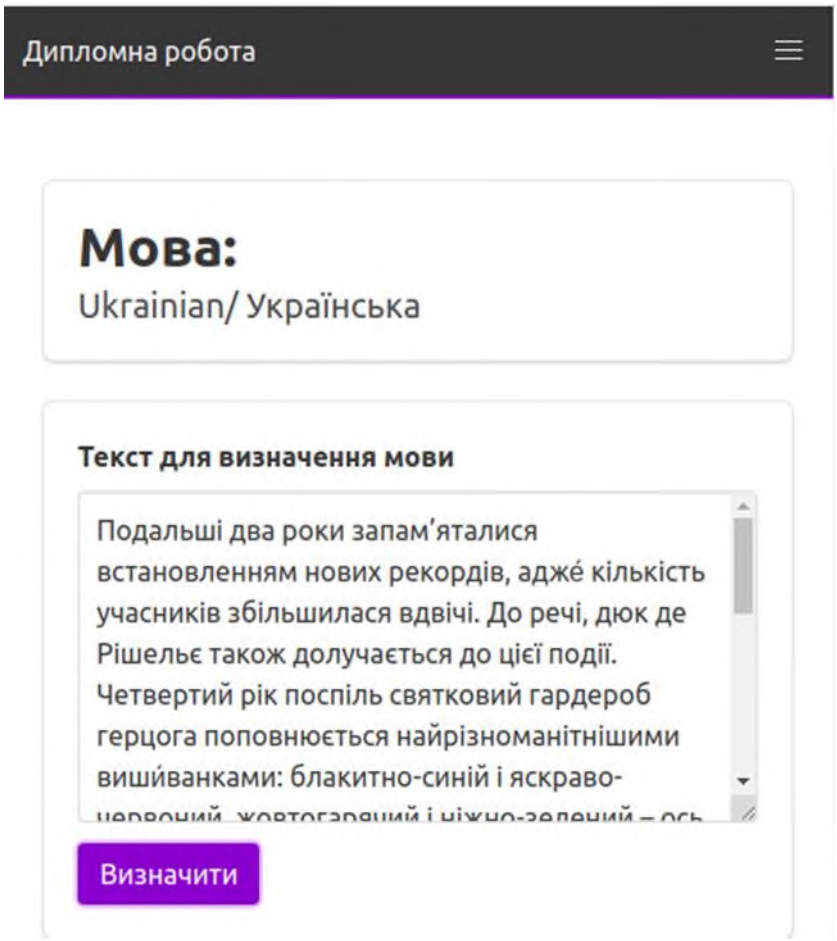


Рис. 3.6. Результат роботи компонента визначення мови тексту

Для підрахунку точності функції визначення мови було проведено дослідження: було взято 25 текстів різної довжини різними мовами в якості вхідних даних. Результати дослідження наведені у таблиці 3.1.

Таблиця 3.1

Результати дослідження функції визначення мови введеного тексту

№ тексту	довжина тексту	коректна мова	Мова, визначена системою
1	три речення	українська	українська

Продовження табл. 3.1

2	три речення	російська	російська
3	три речення	англійська	англійська
4	три речення	німецька	німецька
5	три речення	французька	французька
6	три речення	іспанська	іспанська
7	три речення	італійська	італійська
8	три речення	португальська	португальська
9	три речення	білоруська	білоруська
10	три речення	болгарська	болгарська
11	одне речення	українська	українська
12	одне речення	російська	російська
13	одне речення	англійська	англійська
14	одне речення	німецька	німецька
15	одне речення	французька	англійська
16	одне речення	іспанська	англійська
17	одне речення	італійська	італійська
18	одне речення	португальська	іспанська
19	одне речення	білоруська	російська
20	одне речення	болгарська	болгарська
21	три слова	українська	російська

Закінчення табл. 3.1

22	три слова	російська	російська
23	три слова	англійська	англійська
24	три слова	німецька	німецька
25	три слова	французька	англійська

Як можна побачити з таблиці, на 25 введених текстів приходиться 20 правильно визначених текстів. Причому для текстів, що складають 3 речення і більше – показник точності сягає 100%. Загальний показник точності системи – 80%.

На сторінці функції системи, що відповідає за тональність тексту представлений інформаційний блок, поле для введення тексту для оцінки та вибір мови введеного тексту. На початковому етапі системи користувачу необхідно вибрати із списку трьох мов, доступних в системі. Для розробки пропонується автоматично визначити мову введеного тексту в подальшому.

Результат визначення тональності тексту представлений у вигляді трьох можливих станів: погане емоційне забарвлення (-1), нейтральне емоційне забарвлення (0) та добре емоційне забарвлення (1).

Інтерфейс сторінки системи із елементом визначення тональності тексту зображений на рисунку 3.7.

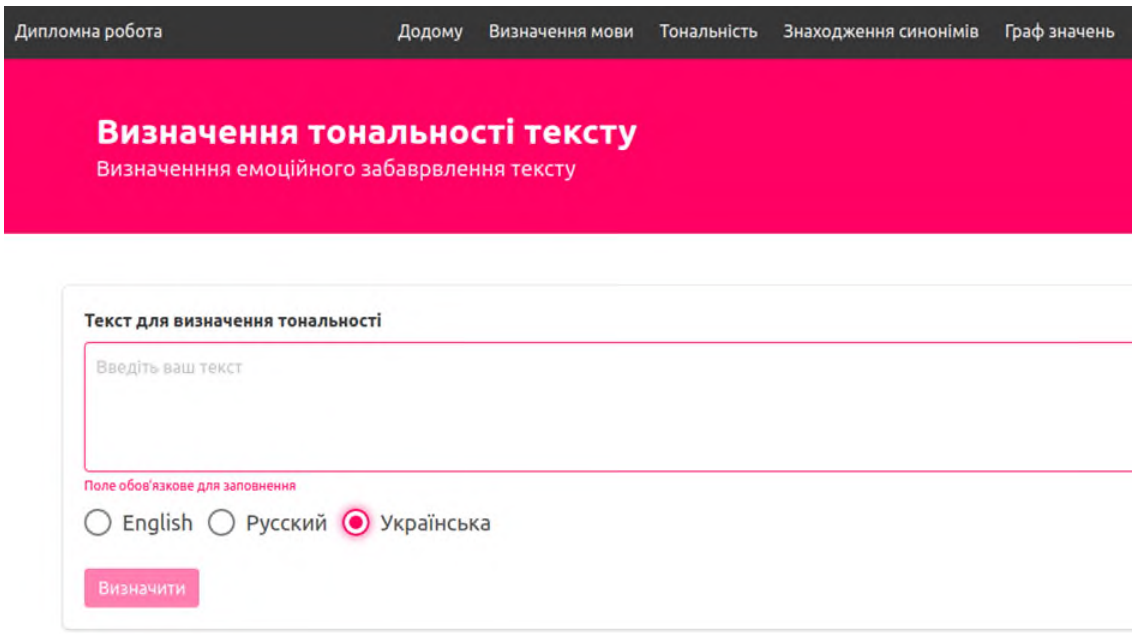


Рис. 3.7. Інтерфейс сторінки системи із елементом визначення тональності тексту

Після введення тексту та вибору мови, результат виводиться на сторінку у вигляді текстового повідомлення та картинки емоційного зображення.

Результат роботи функції визначення тональності тексту зображено на рисунку 3.8.

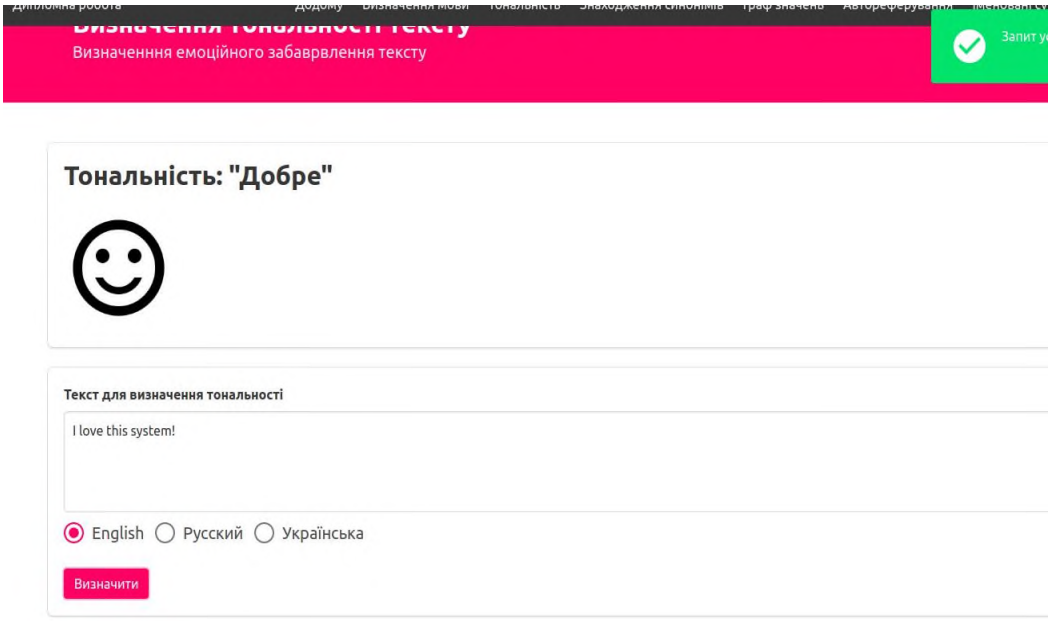


Рис. 3.8. Результат роботи функції визначення тональності системи

Для підрахунку точності функції визначення тональності було проведено дослідження: було взято 24 тексти різними мовами в якості вхідних даних. Результати дослідження наведені у таблиці 3.2.

Таблиця 3.2

Таблиця результатів дослідження функції визначення тональності

№ тексту	мова тексту	правильне емоційне забарвлення	емоційне забарвлення, визначене системою
1	англійська	погано	погано
2	українська	нейтрально	нейтрально
3	російська	добре	добре
4	англійська	нейтрально	нейтрально
5	українська	добре	нейтрально
6	російська	погано	погано
7	англійська	добре	добре
8	українська	погано	нейтрально
9	російська	нейтрально	нейтрально
10	англійська	погано	погано
11	українська	нейтрально	нейтрально
12	російська	добре	нейтрально
13	англійська	нейтрально	нейтрально
14	українська	добре	добре

Закінчення табл. 3.2

15	російська	погано	нейтрально
16	англійська	добре	нейтрально
17	українська	погано	нейтрально
18	російська	нейтрально	нейтрально
19	англійська	нейтрально	нейтрально
20	українська	добре	нейтрально
21	російська	погано	погано
22	англійська	добре	добре
23	українська	погано	нейтрально
24	російська	нейтрально	погано

Як видно з таблиці, на 24 тексти трьому мовами прийшлося 9 помилок системи, із них: англійською – одна помилка, українською – п’ять помилок і три помилки російською мовою. Точність системи в цілому 63%. При цьому, великий вплив на зниження цього відсотку зіграла не популярність української мови при навчанні моделей у мережі.

Сторінка пошуку синонімів має схожий інтерфейс із попередніми сторінками, з відмінністю у тому, що у поле введення можна ввести лише одне слово.

Вихідними даними даного компонента є список слів, що алгоритм розцінює як найбільш близькі за значенням до введеного слова. Результат роботи методу знаходження синонімів введеного слова зображений на рисунку 3.9.

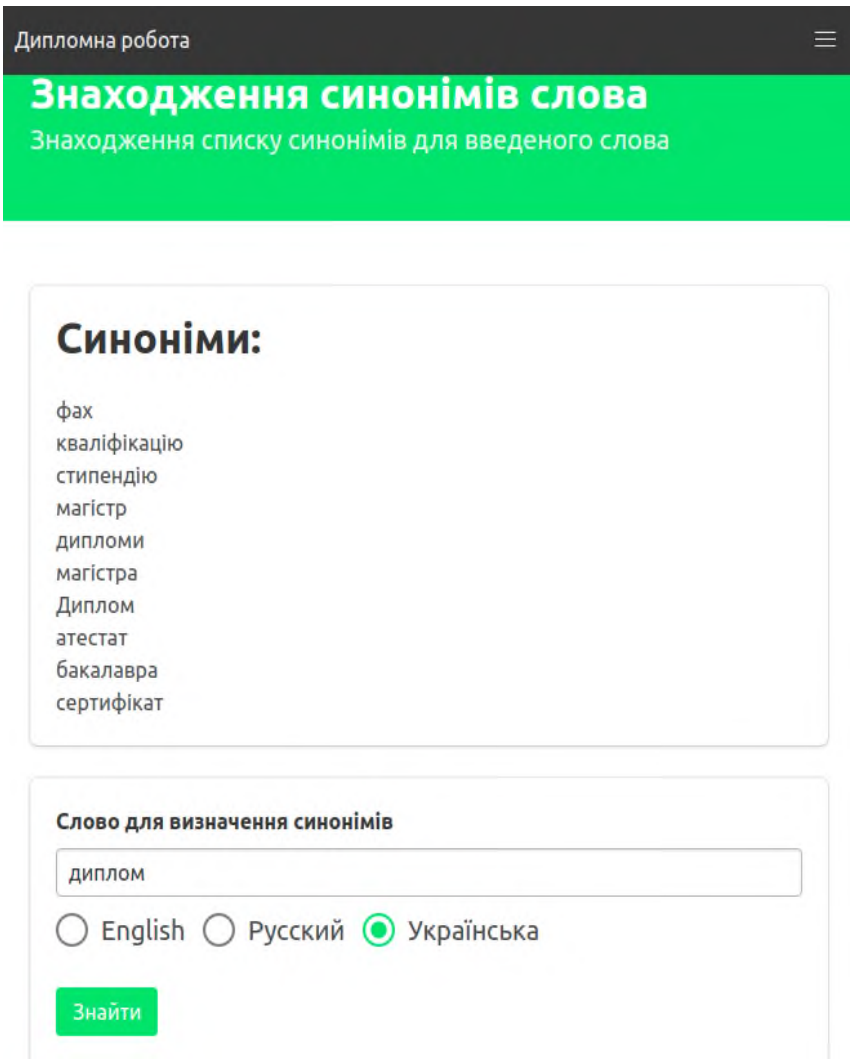


Рис. 3.9. Результат роботи роботи методу знаходження синонімів введеного слова

Для оцінки якості були взяті однакові слова трьома мовами, присутніми в системі в якості вхідних даних як в розроблену систему, так і в системи-аналоги. Результати такого дослідження наведені в таблиці 3.3.

Таблиця 3.3

Результат дослідження роботи функції синонімізації

слово	мова	кількість підібраних синонімів системи	кількість правильно підібраних синонімів	назва сторонньої системи	кількість підібраних синонімів	кількість правильно підібраних синонімів
диплом	українська	10	7	синоніми.укр	4	3
диплом	російська	10	7	*sinonim.org*	5	3
diploma	англійська	10	9	*thesaurus.com*	13	6
музика	українська	10	6	синоніми.укр	11	3
музыка	російська	10	7	*sinonim.org*	91	2
music	англійська	10	8	*thesaurus.com*	20	4
знання	українська	10	7	синоніми.укр	6	4
знания	російська	10	6	*sinonim.org*	12	6
knowledge	англійська	10	8	*thesaurus.com*	40	20
наука	українська	10	8	синоніми.укр	10	8

Закінчення табл. 3.3

наука	російська	10	7	*sinonim.org*	14	1
science	англійська	10	10	*thesaurus.com*	14	5
мистецтво	українська	10	9	синоніми.укр	3	2
искусство	російська	10	8	*sinonim.org*	53	10
art	англійська	10	9	*thesaurus.com*	19	8
спеціалізація	українська	10	5	синоніми.укр	0	0
специализация	російська	10	6	*sinonim.org*	8	1
specialization	англійська	10	8	*thesaurus.com*	16	10

Із таблиці 3.3. видно, що якість роботи функції в розробленій системі дорівнює: для англійської мови – 86%, для української – 70% та російської – 68%. Якщо порівняти отримані дані із дослідження із аналогічними даними сторонніх систем, взятих для порівняння – можна побачити діаграму порівняння якості роботи функції синонімізації розробленої системи, та сторонніх популярних систем, що зображений на рис.3.10.

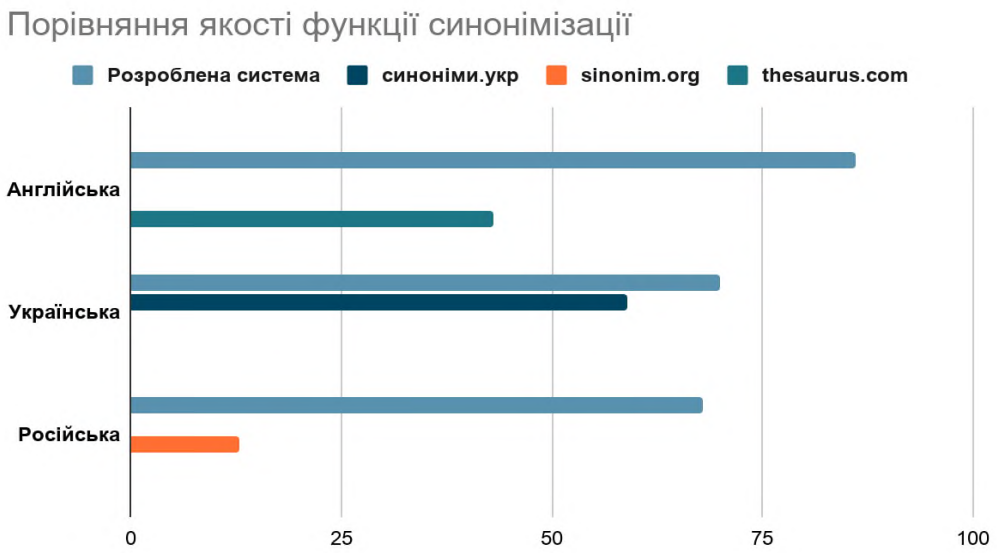


Рис. 3.10. Діаграма порівняння якості роботи функції синонімізації розробленої системи та сторонніх популярних систем

Граф сутностей *Microsoft* – елемент системи, що демонструє можливості та зручність використання моделі «маркетплейс». Це є яскравим прикладом того випадку, коли реалізація функції знаходиться у бізнеса, що її пропонує, та від маркетплейсу потрібна лише швидка інтеграція. В даному випадку слово, що вводить користувач відправляється на сервер *Microsoft* за допомогою методів *API*, та у відповідь система отримує масив елементів із відповідними вагами. Ваги сортуються від більшого до меншого і виводяться користувачу. Ваги є довжиною вектора у уявному графі між словом, та його представленням. Ваги варуються від 0 до 1 (чим більше вага – тим ближче значення до слова). Граф будується за алгоритмами та власною базою знань *Microsoft*, яких не має у відкритому доступі. Мінусом даного компоненту є його робота лише з англійською мовою. Результат роботи компоненту *Microsoft Graph* зображений на рисунку 3.11.

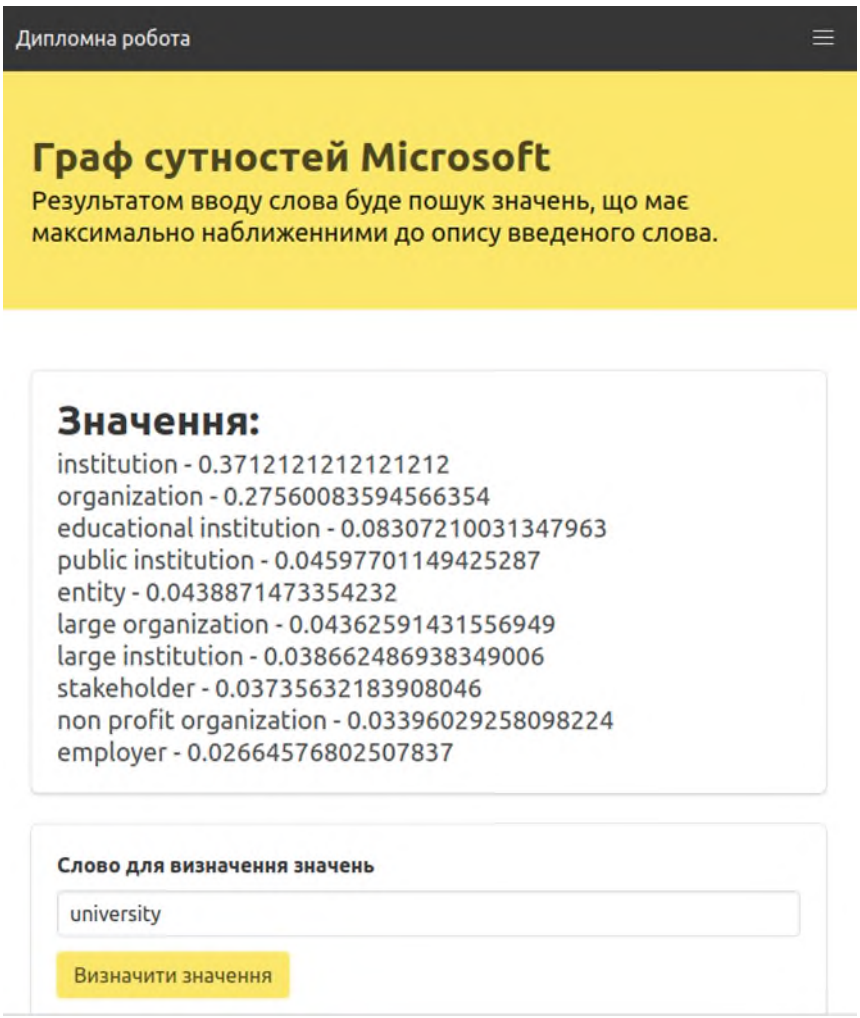


Рис. 3.11. Результат роботи компоненту *Microsoft Graph*

Сторінка функції автореферування представлена в системі в простому дизайні і представляє собою лише інформаційний блок та блок введення тексту. Після обробки в інтерфейсі користувача з’являється блок із реченням, або парою речень, що із найбільшою вірогідність передають сенс введеного тексту. Оскільки методи здійснення сумаризації є екстрактивним, вихідний текст, що вирваний із підтексту не завжди якісно описує текст.

Результат роботи функції автореферування системи зображений на рисунку 3.12.

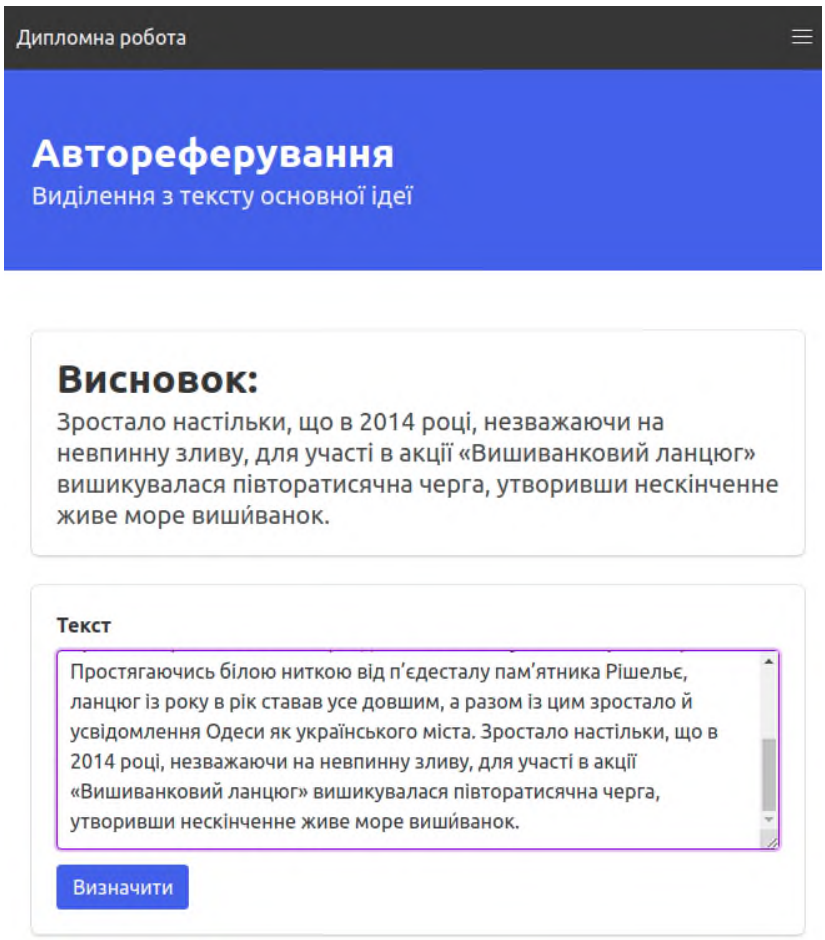


Рис. 3.12. Результат роботи функції автореферування

Однією із основних функцій системи є можливість знаходження іменованих сутностей в тексті. Інтерфейс даного елементу додатку включає вибір мови та поле введення тексту. Результат відображається на сторінці у вигляді списку сутностей, що згруповані за категоріями.

Результат знаходження іменованих сутностей в тексті зображений на рисунку 3.13.

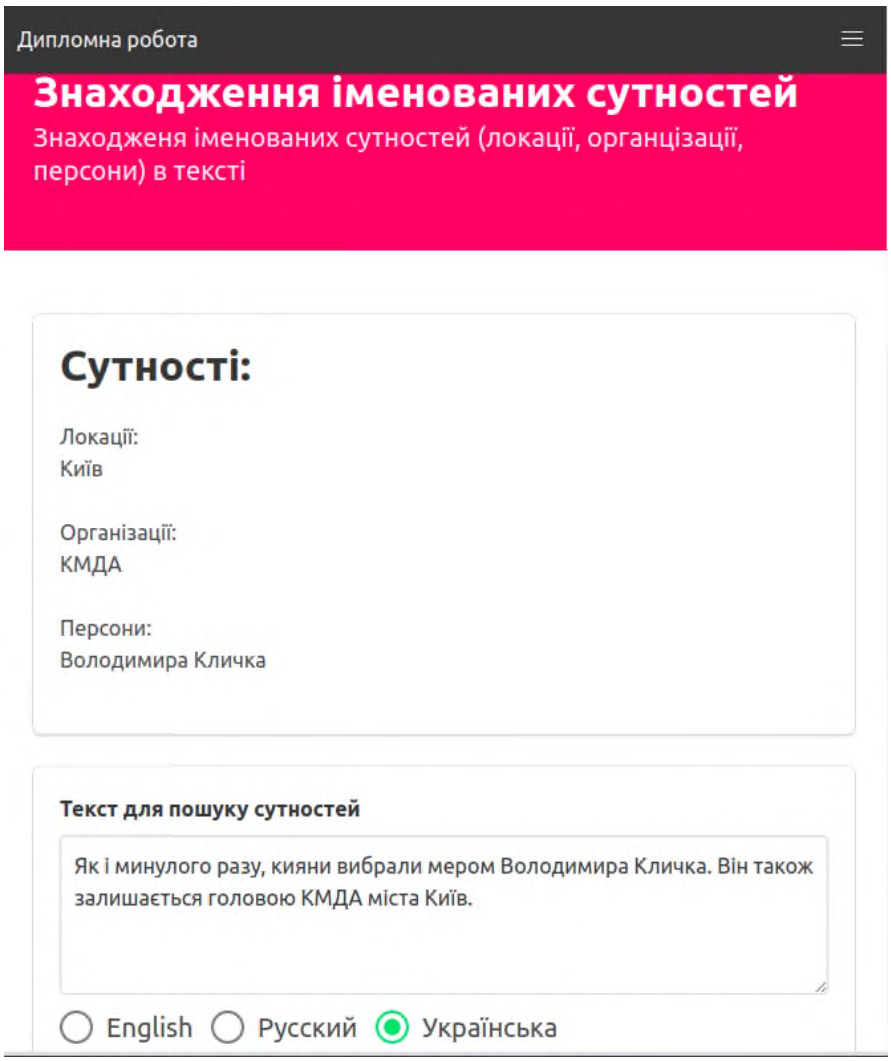


Рис. 3.13. Результат знаходження іменованих сутностей в тексті

Для оцінки якості результату знаходження іменованих сутностей в тексті було взято 21 текст трьома мовами, та порівняно кількість заданих іменованих сутностей з тими, що знайшла система. Результати дослідження наведені у таблиці 3.4.

Таблиця 3.4

Результати дослідження якості роботи модулю знаходження іменованих сутностей в тексті

№ тексту	мова	кількість іменованих сутностей в тексті	кількість коректно знайдених сутностей
1	українська	5	5
2	англійська	5	5
3	російська	5	5
4	українська	15	14
5	англійська	15	15
6	російська	15	15
7	українська	20	20
8	англійська	20	19
9	російська	20	18
10	українська	30	25
11	англійська	30	30
12	російська	30	24
13	українська	35	32
14	англійська	35	35
15	російська	35	18
16	українська	40	28

Закінчення табл. 3.4

17	англійська	40	36
18	російська	40	30
19	українська	50	42
20	англійська	50	47
21	російська	50	40

За результатами дослідження було виведено такі статистичні показники якості функції за трьома мовами:

– українська – 85% правильно визначених іменованих сутностей;

– англійська – 96%;

– російська – 77%.

Останнім наявним методом системи надання послуг обробки текстової інформації є парсинг веб-сторінки. Його інтерфейс представляє собою поле, в яке користувач вводить посилання на бажану веб-сторінку. В майбутньому пропонується дати можливість клієнту завантажувати файл сторінки вручну. Приклад роботи функції парсингу веб-сторінки зображений на рисунку 3.14.

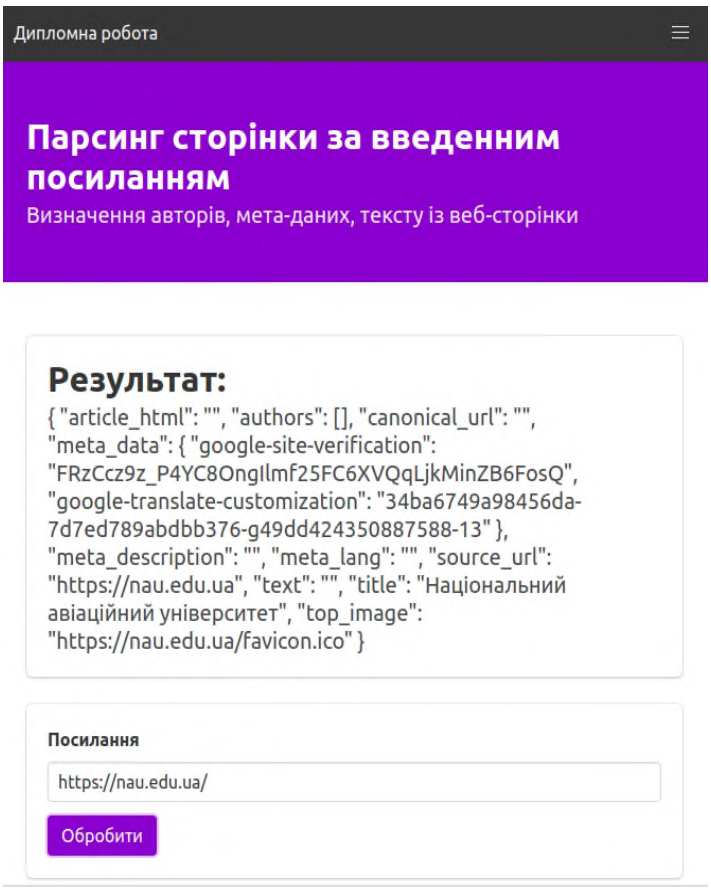


Рис. 3.14. Приклад роботи функції парсингу веб-сторінки

3.3. Висновки до розділу

В даному розділі було досліджено результати та якісні показники системи надання послуг з обробки текстової інформації, а саме:

– було розроблено архітектуру додатку за моделлю «маркетплейс»;
– було наведено стек технологій та архітектурних рішень при програмуванні системи (мови програмування: *PHP*, *Python*, *Javascipt*; патерни програмування: *MVC*, *SPA*; бази даних: *Mysql*, *Redis*);
– було продемонстровано інтерфейс додатку: його меню, особливості побудови, дизайн кожної сторінки, вивід результатів тощо;
– було проведено тестування працездатності додатку: повний шлях користувача по всім функціям, що доступні із наведенням та поясненням результатів;

– було досліджено результати роботи функцій системи надання послуг з обробки текстової інформації: було замірено якісні показники функцій додатку на основі власних тестувань, при цьому заміри проводилися для трьох доступних мов в системі: української, англійської та російської. Було проведено порівняння функції пошуку синонімів для кожної мови у розробленій системі, та у популярних сторонніх додатках. Результатами даного порівняння вишла діаграма, що наочно показує перевагу якісних характеристик розробленого додатку у всіх доступних мовах.

ВИСНОВКИ

Під час виконання дипломної роботи було здійснено аналіз існуючих систем надання послуг з обробки текстової інформації. Було викладено опис поняття «маркетплейс», надано їх класифікацію, а також описано основні моделі маркетплейсів: *C2C, B2C, B2B*. Для розробки системи в дипломній роботі було обрано модель маркетплейсу *B2C* – модель, що передбачає, що одна сторона відносин, а саме постачальник товарів або послуг – є бізнесом. Зазвичай на таких майданчиках зібрані підприємці малого та середнього бізнесу, що надають послуги користувачам.

Були викладені описи текстових аналізаторів, та було визначено, що вони відрізняються за типом програмного забезпечення, кількістю функцій, монетизацією системних особливостей, виконують функції пошуку тексту, синтаксичного аналізу, аналізу веб-сторінок, порівняльного аналізу текстової інформації, складність налаштування такого програмного забезпечення та напрями його діяльності.

Було розглянуто функціональні особливості та алгоритми роботи комп'ютерної системи надання послуг з обробки текстової інформації.

Було досліджено інструменти для визначення мови введеного тексту, наведено схему роботи алгоритму даної функції в системі, що базується на побудові *N*-грам.

Було наведено алгоритм визначення мови введеного тексту, а саме він включає наступні етапи: виокремлення структури введеного тексту; співставлення введених слів із словниками мов; розрахунок вірогідності для кожної мови, перевірка «спеціальних символів» і видача найбільш вірогідного результату.

Було досліджено алгоритм виділення компонент електронного текстового документу, приведено схему алгоритму, що включає в себе введення адреси сторінки користувачем; запит на дану сторінку; модуль парсингу сторінки; обробка результату та виділення компонент; модуль експорту сутностей; виведення результату користувачеві.

Було досліджено результати та якісні показники системи надання послуг з обробки текстової інформації. Було розроблено архітектуру додатку за моделлю «маркетплейс». Було наведено стек технологій та архітектурних рішень при програмуванні системи (мови програмування: *PHP*, *Python*, *Javascipt*; патерни програмування: *MVC*, *SPA*; бази даних: *Mysql*, *Redis*).

Було продемонстровано інтерфейс додатку: його меню, особливості побудови, дизайн кожної сторінки, вивід результатів. Було проведено тестування працездатності додатку: повний шлях користувача по всім функціям, що доступні із наведенням та поясненням результатів.

Було досліджено результати роботи функцій системи надання послуг з обробки текстової інформації: було замірено якісні показники функцій додатку на основі власних тестувань, при цьому заміри проводилися для трьох доступних мов в системі: української, англійської та російської.

Було проведено порівняння функції пошуку синонімів для кожної мови у розробленій системі, та у популярних сторонніх додатках. Результатами даного порівняння стала діаграма, що наочно показує перевагу якісних характеристик розробленого додатку у всіх доступних мовах

Матеріали дипломної роботи рекомендується використовувати при проведенні наукових досліджень, у навчальному процесі фахівців з системного програмування, веб-програмування та розробки систем обробки текстової інформації.

Прогнозні припущення про розвиток об’єкту та предмету дослідження – зручна система надання послуг з обробки текстової інформації, що має просту та зрозумілу архітектуру, яку можна швидко масштабувати. Також перевагою розробки або впровадження системи є вже створена модель монетизації.

СПИСОК БІБЛІОГРАФІЧНИХ ПОСИЛАНЬ ВИКОРИСТАНИХ ДЖЕРЕЛ

1. ГОСТ 19.701–90 ЕСПД. Схемы алгоритмов, программ, данных и систем. Условные обозначения и правила выполнения.
2. ДСТУ ГОСТ 7.1:2006. Бібліографічний запис. Бібліографічний опис. Загальні вимоги та правила складання.
3. ГОСТ 2.106–96 ЕСКД “Текстовые документы”.
4. ДСТУ 3008–95 “Документація. Звіти у сфері науки і техніки. Структура і правила оформлення”.
5. Бойченко С.В., Іванченко О.В. Положення про дипломні роботи (проекти) випускників Національного авіаційного університету. Затверджено наказом ректора від 14.12.2017 № 594/од.
6. *Boris Weltz, Angela Tran Kingyens. A guide to marketplaces. Versionone,* 2015 – 52 с.
7. *What is Marketplace and how it works* [*Electronic Resource*]. – 14.10.2019. – *Access mode: https://postium.ru/marketplejs-chto-eto-i-kak-rabotaet/*
8. А. А. Антонова. Об использовании синтаксического анализатора *Cognitive Dwarf* 2.0. Труды ИСА РАН, 2008 – 25 с.
9. *Alexey Sokirko. A technical overview of DWDS/Dialing Concordance [Electronic Resource]*. - 2003. – *Access mode*: *http://aot.ru/docs/OverviewOfConcordance.htm*
10. *Analyst – about an approach [Electronic Resource]*. – 2001. – *Access mode: https://www.analyst.ru/index.php?lang=rus&dir=content/tech/&id=approach&left=content/tech/menu.txt*
11. *IBM Intelligent Miner for Text Version [Electronic Resource]*. – 1999. – *Access mode: https://www-01.ibm.com/common/ssi/cgi-bin/ssialias?subtype=ca&infotype=an&appname=iSource&supplier=897&lettern um=ENUS298-447#:~:text=IBM%20Intelligent%20Miner%20for%20Text%20offers%20system%*

20integrators%2C%20solution%20providers,content%20management%2C%20and%20knowledge%20management

12. *Putting your text to work with Oracle Text [Electronic Resource]*. – 2020 – *Access mode: https://www.oracle.com/database/technologies/enterprise-edition.html*
13. *PageRank, Wikipedia [Electronic Resource]*. – 2018 – *Access mode: https://ru.wikipedia.org/wiki/PageRank#PageRank_%D0%B2_%D0%BF%D1%80%D0%BE%D0%B4%D1%83%D0%BA%D1%82%D0%B0%D1%85_Google*
14. Анализ текстов *Text Mining [Electronic Resource]*. – 2010 – *Access mode*: *http://statsoft.ru/products/STATISTICA_Data_Miner/STATISTICA_Text_Miner/*
15. Ананьев Денис Владиславович. Разработка алгоритма определения языка для коротких текстов. Санкт-Петербургский государственный университет, 2016 – 25 с.
16. *Marco Lui. Langid.py: An Off-the-shelf Language Identification Tool. Department of Computing and Information Systems University of Melbourne*, 2012 – 10 с.
17. Автоматическое определение языка произвольного текста на PHP — библиотека *PHPLangautodetect [Electronic Resource]*. – 15.06.2008 – *Access mode: https://abrdev.com/?p=346*
18. *Kenneth Heafield. Language Identification and Modeling in Specialized Hardware. Bloomberg L.P*, 2015 – 10 с.
19. *Victoria Bobicev. Emotions in words: developing a multilingual WordNet-Affect. Technical University of Moldova*, 2009 – 25 с.
20. *Sentiment analysis: conception, methods [Electronic Resource]*. – 2020 – *Access mode: http://datareview.info/article/analiz-tonalnosti-teksta-kontseptsiya-metodyi-oblasti-primeneniya/*
21. *P. Bo. A Sentimental Education: Sentiment Analysis Using Subjectivity Summarization Based on Minimum Cuts. ACL*, 2004 – 100 с.
22. *R. Bhayani. Twitter Sentiment Classification Using Distant Supervision. Stanford*, 2009 – 50 с.
23. *NLP. Basics. Techics [Electronic Resource]*. – 14.05.2019 – *Access mode: https://habr.com/ru/company/abbyy/blog/449514/*

24. *NLP with Python: Nearest Neighbors Search [Electronic Resource]*. – 27.12.2019 – *Access mode: https://sanjayasubedi.com.np/nlp/nlp-with-python-nearest-neighbor-search/*
25. *Text summarization [Electronic Resource]*. – 2019 – *Access mode: https://ddecisions.ai/testsummary*
26. *Auto-summarization for three languages [Electronic Resource]*. – 26.11.2105 – *Access mode: https://habr.com/ru/post/271771/*
27. *Andi Gutmans. PHP5 Power Programming. Prentice Hall*, 2005 – 704 с.