**Київський національний торговельно-економічний університет**
**Кафедра інженерії програмного забезпечення та кібербезпеки**

# ВИПУСКНИЙ КВАЛІФІКАЦІЙНИЙ ПРОЄКТ

**на тему:**

# «Проектування аналітично-лінгвістичної системи на основі SUM-методів опорних векторів»

Студентки 2м курсу, 2 групи, спеціальності 121 «Інженерія програмного забезпечення» спеціалізації «Інженерія програмного забезпечення»

_____ підпис студента

Роскіної Єлизавети Олегівни

Науковий керівник кандидат економічних наук, доцент кафедри інженерії програмного забезпечення та кібербезпеки

_____ підпис керівника

Палагута Катерина Олексіївна

Гарант освітньої програми доктор економічних наук, професор кафедри інженерії програмного забезпечення та кібербезпеки

_____ підпис гаранта

Токар Володимир Володимирович

КИЇВ - 2021

**Київський національний торговельно-економічний університет**

Факультет інформаційних технологій
Кафедра інженерії програмного забезпечення та кібербезпеки
Освітній ступінь магістр
Спеціальність 121 «Інженерія програмного забезпечення»

**Затверджую**
Зав. кафедри інженерії програмного забезпечення та кібербезпеки
Криворучко О. В.
«10» листопада 2020 р.

# Завдання
## на випускний кваліфікаційний проєкт студентові

Роскіній Єлизаветі Олегівні
(прізвище, ім'я, по батькові)

1. Тема випускного кваліфікаційного проєкту «Пошук та виявлення неявних взаємозв'язків між термінами в колекції текстів»

Затверджена наказом ректора від «30» листопада 2020 р. № 3224

2. Строк здачі студентом закінченого проєкту 25 листопада 2021

3. Цільова установка та вихідні дані до проєкту

*Мета проєкту* є створення застосунку з використання тезаурусу для пошуку важливих україномовних термінів і термінологічних зв'язків між ними

*Об'єкт дослідження* - Функція тезаурусу під час пошуку інформації

*Предмет дослідження* розробка структурної схеми алгоритму

4. Консультанти проєкту із зазначенням розділів, які консультують:

| Розділ | Консультант (прізвище, ініціали) | Підпис, дата | |
|---|---|---|---|
| | | Завдання видав | Завдання прийняв |
| | | | |
| | | | |

5. Зміст випускного кваліфікаційного проєкту (перелік питань за кожним розділом)

ВСТУП

РОЗДІЛ 1 ОГЛЯД ІСНУЮЧИХ ПІДХОДІВ

1.1 Роль тезаурусу в інформаційному пошуку

1.2 Тезаурус як форма подання зв’язаної термінології

1.3 Rdf як формат публікації тезаурусів

1.4 Огляд стандарту json-ld як конкретної специфікації rdf

1.5 Дослідження існуючих автоматизованих методів побудови тезаурусів

1.5.1 Статистичні методи

1.5.1.1 Ранжування термінів за метрикою TF·IDF

1.5.1.2 Метод спільного вжитку

1.5.1.3 Метод концептуального простору

1.5.2 Лексикографічні методи

1.5.2.1 Огляд лексикографічних відомостей щодо термінології

1.5.2.2 Метод генерації тезаурусу з використанням техніки браку знань

1.5.2.3 Гіпонімія як основа для пошуку ієрархічних зв’язків. Метод збирання гіпонімів з великих текстових колекцій

1.5.2.4 Метод збирання гіпонімів з великих текстових колекцій

РОЗДІЛ 2. МЕТОДИ ФОРМУВАННЯ ТЕЗАУРУСА

2.1 Структурна схема алгоритму

2.1.1 Індексування і фільтрація важливих термінів

6. Календарний план виконання проєкту

| № пор. | Назва етапів випускного кваліфікаційного проєкту | Строк виконання етапів проєкту | |
|---|---|---|---|
| | | за планом | фактично |
| 1 | 2 | 3 | 4 |
| 1. | *Вибір теми випускного кваліфікаційного проєкту* | *21.09.2020* | *21.09.2020* |
| 2. | *Розробка та затвердження завдання на проєкт магістра (стац/заоч)* | *06.11.2020* | *22.12.2020* |
| 3. | *Вступ та перелік літературних джерел* | *27.02.2021* | *27.02.2021* |
| 4. | *Розробка технічного завдання* | *20.03.2021* | *20.03.2021* |
| 5. | *Розділ 1. Огляд існуючих підходів* | *16.04.2021* | *16.04.2021* |
| 6. | *Розділ 2. Методи формування тезауруса* | *24.05.2021* | *24.05.2021* |
| 7. | *Розділ 3. Практична частина* | *21.06.2021* | *21.06.2021* |
| 8. | *Розробка програми та методики тестування* | *18.10.2021* | *18.10.2021* |
| 9. | *Написання наукової статті* | *22.05.2021* | *22.05.2021* |
| 10. | *Керівництво користувача* | *21.10.2021* | *21.10.2021* |
| 11. | *Висновки та пропозиції* | *01.11.2021* | *01.11.2021* |
| 12. | *Здача випускного кваліфікаційного проєкту на кафедру (перша перевірка)* | *03.11.2021* | *03.11.2021* |
| 13. | *Підготовка автореферату та презентації доповіді* | *03.11.2021* | *03.11.2021* |
| 14. | *Попередній захист випускного кваліфікаційного проєкту* | *22.11.2021 – 25.11.2021* | *22.11.2021* |
| 15. | *Здача зброшурованої випускного кваліфікаційного проєкту* | *25.11.2021* | *25.11.2021* |
| 16. | *Зовнішнє рецензування випускного кваліфікаційного проєкту* | *26.11.2021* | *26.11.2021* |
| 17. | *Підготовка до публічного захисту випускного кваліфікаційного проєкту* | | |

7. Дата видачі завдання «10» листопада 2020 р.

8. Науковий керівник випускного кваліфікаційного проєкту Криворучко О.В

(прізвище, ініціали, підпис)

9. Гарант освітньої програми Токар В.В.

(прізвище, ініціали, підпис)

10. Завдання прийняв до виконання студент Роскіна Є.О.

(прізвище, ініціали, підпис)

**11. Відгук керівника випускного кваліфікаційного проєкту**

Науковий керівник випускного кваліфікаційного проєкту

_____

*(підпис, дата)*

Відмітка про попередній захист Палагута К.О.

*(ПІБ, підпис, дата)*

12. Висновок про випускний кваліфікаційний проєкт

Випускний кваліфікаційний проєкт студента Роскіна Є.О.

*(прізвище, ініціали)*

може бути допущена до захисту екзаменаційній комісії.

Гарант освітньої програми Токар В.В.

*(прізвище, ініціали, підпис)*

Завідувач кафедри Криворучко О. В.

*(підпис, прізвище, ініціали)*

«_____»_____20 ___р.

## АНОТАЦІЯ

**Роскіна Є.О. Пошук термінів та виявлення неявних взаємозв'язків між ними в колекції текстів**

У роботі розглянуто аналіз наявних підходів побудови тезаурусів, розробці прикладного методу, що надав би можливість досить точно визначати важливі україномовні терміни і термінологічні зв'язки між ними, і реалізації запропонованого методу у вигляді веб застосунку для аналізу його ефективності на справжніх даних, з подальшою можливістю застосування розробленого компоненту як складової пошукової системи наукових документів.

**Ключові слова:** пошук термінів, тезаурус, веб-застосунок, реляційна база даних.

## ABSTRACT

**Roskina Y. Search and detection not evident relationships between terms in collection of texts**

In this work the analysis of existing approaches thesauri construction, development of application method that would provide the opportunity to accurately define important terms and terminology Ukrainian-relationships between them and the proposed method as a web application for analysis of its effectiveness on real data, with the further possibility application component developed as part of the search engine of scientific documents.

**Keywords:** term search, thesaurus, web application, relational database

**ЗМІСТ**

# ВСТУП

Сфера наукових досліджень є найбільш продуктивною формою як збагачення людського понятійного простору новими концептами і зв'язками між ними, так і безпосередньо пов'язаним з цим лексичним процесом словотвору для опису нових концепцій. На думку дослідників [9, 5], найбільш цінним джерелом розширення лексики кожного з нас вважається наукова україномовна література. В зв'язку з цим створення відповідної термінології, її характеристики і уніфікації розроблених терміносистем у відповідних спеціалізованих словниках, пошуку нових термінів потребує турботливого ставлення в даній сфері. Мінливість понять з часом та об'єктивність складності причин є основними підставами, через які дуже часто темпи створення і актуалізації словників не встигають за поступом прогресу в найновіших дослідженнях. Проте залишається гострою необхідність порозуміння серед дослідників на понятійному рівні, що вимагає як уніфікованої і доступної термінологічної бази, так і якісної пошукової системи наукових документів. Уніфікована та доступна термінологічна база на рівні з якісною пошуковою системою являється вимогою, так як потреба порозуміння серед дослідників на понятійному рівні залишається нагальною.

Використання тезаурусу [35], що являється довідником характеру і сили зв'язків між термінами є одним із дієвих способів покращення доцільності пошукової видачі таких систем. Існує декілька методів побудови тезаурусів серед яких автоматизований метод якнайкраще підходить для сфери наукових досліджень в зв'язку з тим, що оновлення інформації відбувається з високими темпами, а також пов'язана с високою собівартості долучення експертів до такої роботи.

Магістерська робота присвячена аналізу наявних підходів побудови тезаурусів, створенні прикладного методу для точного визначення вагомих україномовних термінів та термінологічних зв’язків між ними, а також втілення запропонованого методу у вигляді веб-застосунку для аналізу його ефективності на реальних даних, з подальшою можливістю використання розробленого компоненту як складової пошукової системи наукових документів.

Поява у відкритому доступі прикладних рішень для аналізу текстів українською мовою, з можливостями використання таких методів як лематизація і тегування за частинами мови тільки підтверджують актуальність цієї проблеми. Під час розробки методу було прийнято до уваги обмеженість документарних колекцій, що були випущені українською мовою. Це потребувало врахування можливості ітеративного додавання наукових документів до термінологічної бази з послідуючим оновленням змісту тезауруса.

# РОЗДІЛ 1

# ОГЛЯД ІСНУЮЧИХ ПІДХОДІВ

## 1.1 ФУНКЦІЯ ТЕЗАУРУСУ ПІД ЧАС ПОШУКУ ІНФОРМАЦІЇ

В основному, інформаційна вимога користувача пошукової системи не відповідає термінам, що зустрічаються в документах. Також, часто буває, що користувач помилково розуміє в конкретний час в певному місці термінологію області знань, в якій він здійснює пошук. Пошукова видача потребує покращення, це можна виконати за допомогою використання тезаурусів термінів предметних областей. Тезауруси є таблицями термінів, що поєднують пов'язані між собою терміни, зазвичай вказуючи тип зв'язку (NT, BT, USE, RT) [11]. Для того щоб правильніше класифікувати документи по категоріях можна використовувати тезауруси на етапі індексації документів інформаційними системами. Або їх можна також використовувати під час пошуку, розширюючи пошуковий запит користувача пов'язаними термінами.

Тут доцільно згадати експерименти Фурне і Ландауера [22, с.17], в яких показано, що однакові терміни для опису тих самих речей були обрані більшістю користувачів тільки у 20% випадків, що свідчить про потребу для пошукових систем враховувати синонімію термінів, що в свою чергу задається саме в тезаурусах. До того ж, були виконані порівняльні заміри релевантності звичайної пошукової видачі з розширеною пошуковою видачою з додаванням інформацію з тезаурусу, на думку Чена [22, с.178], відображають можливість її удосконалення. Таким чином, можна вважати обґрунтованим використання тезаурусів Опираючись на дану інформацію, можна вважати аргументованим використання тезаурусів в пошукових системах з метою поліпшення їх якості.

Існує основна проблема створення тезаурусів – основна кількість комерційних баз даних, які поширюють наукову інформацію, вони складаються фахівцями зі складання тезаурусів та експертами з областей знань, тобто своїми руками. Термінологічні словники потребують частішого оновлення, адже вони досить швидко старіють в таких новітніх областях знань, як біоінформатика або комп'ютерна інженерія, де належна термінологія тільки формується і створюється велика кількість сучасних публікацій, що потребує залучення експертів. На противагу такому підходу, існують методи автоматизованої побудови тезаурусів, що в якості корпусу беруть усі найновіші публікації з теми, і будують на їх основі зв'язки між термінами. Всупереч даному підходу кращу використовувати методи автоматизованої побудови тезаурусів, де за основу беруться усі новітні публікації з теми, і будують на їх фундаменті зв'язки між термінами. За рахунок даної системи зменшується час та витрати на оновлення термінологічних зв'язків. Моніка Лассі [35] оприлюднила публікацію, де було опубліковано основні методи автоматизованої побудови тезаурусів, такі як метод баєсівських мереж, метод спільного вжитку, а також метод концептуального простору. Дані методи мають як різну ефективність і часову оцінку складності, так і принципи - статистичний і лексикографічний. Згадані вище методи відрізняються своєю специфікацію, маючи як різну ефективність та часову оцінку складності, так і принципи - статистичний і лексикографічний. Саме тому дана магістерська теза присвячена розробці та застосуванню нового методу, що може скомбінувати ідеї, які згадуються в цих підходах.

## 1.2 ТЕЗАУРУС ЯК СТРУКТУРА ПОДАННЯ ЗВ'ЯЗАНОЇ ТЕРМІНОЛОГІЇ

**Тезаурус –** це керований словник, що використовується з метою покращення процесу пошуку пов'язаних термінів і містить семантичні зв'язки між термінами. В роботі Шутца і Педерсона [39, с.307] тезаурус має таке визначення: "Ми визначаємо тезаурус як просте відношення між словами і іншими близько пов'язаними словами", висвітлюючи далі, що тезаурус має бути досить винятковим, щоб надавати користувачу синоніми шуканих слів. Таке означення є доволі загальним та коротким, окрім цього існує ще велика кількість опису даного терміну. Всупереч даному означенню, в роботі Міллера [37, с.489] опубліковано більш точне визначення тезауруса, що являється "лексико-семантичною моделлю концептуальної реальності або її представника, що виражена у формі системи термінів і їх зв'язків, пропонує доступ за допомогою багатьох аспектів і використовується в якості системи обробки і пошуку всередині модулю інформаційної пошукової системи". Цікавим є те, що автор підкреслює практичність застосування програмних модулів із такою функціональністю, а також принципову нерозривність теоретичної моделі тезауруса.

Зв'язки між термінами в тезаурусах мають загальні позначення, що наведені нижче:

- ширший термін (BT),
- вужчий термін (NT),
- переважний термін (USE),
- пов'язаний термін (RT) – використовується для позначення зв'язків, де неможливо точно встановити напрямок зв'язку.

Одним з найкращих способів складання тезаурусів, де залучаються експерти з конкретних галузей знань – це спосіб конвекційного складання тезаурусів, але він є достатньо дорогоцінним. Задля зменшення вартості складання тезаурусів можна використовувати вже створені лексикографічні бази даних, таких як WordNet, де є істотні труднощі при застосуванні такого корпусу загальної лексики до доменно-специфічних областей знань [39, с.308]. На противагу є альтернативний підхід, що використовує автоматизовані методи створення тезаурусів, які можуть підготувати достатньо точний фундамент для майбутнього уточнення знавцями.

## 1.3 ВИКОРИСТАННЯ RDF ДЛЯ ПУБЛІКАЦІЇ ТЕЗАУРУСІВ

Одним із найпопулярніших способів подання даних і метаданих для технологій семантичного вебу є Формат RDF [38]. Фундаментом даного формату знаходиться ідея подання інформації у вигляді триплетів “суб’єкт”-“предикат”-“об’єкт”, якщо ми не заглиблюємося у деталі абстрактного синтаксису і певних специфікацій, які точно розроблені згідно з міжнародними стандартами. Ця загальна, здавалося б, нескладна модель може вдало задовольнити вимоги тезауруса для відтворення його змісту. Однак, крім відповідності вимогам рівня узгодженості моделі, міжнародний формат має ще одну надзвичайно важливу функцію - отримати широку глобальну підтримку на рівні впровадження прикладних систем. Як зазначалося раніше, роль тезаурусу залежить не тільки від коректності та обсягу відображуваного посилання на термін, але й від фактичного використання програмного модуля, можливості машинної обробки та застосовності доступу. Оскільки дані, що обробляються програмною частиною тезаурусу, можуть бути безпосередньо опубліковані в Інтернеті у міжнародно прийнятному форматі, ми пропонуємо

формат RDF та підлеглу систему веб-служб, а інтерфейс програмного забезпечення використовуємо як остаточний доступ до тезаурусу.

## 1.4 АНАЛІЗ СТАНДАРТУ JSON-LD ЯК ВИЗНАЧЕНОЇ СПЕЦИФІКАЦІЇ RDF

Серед конкретних специфікацій RDF, формат JSON-LD, поданий у стандарті ISO-25964 [31], на нашу думку найкращим чином відповідає поставленій задачі публікації ресурсів тезауруса у вигляді веб-сервісу. Ми вважаємо, що серед конкретних специфікацій RDF, формат JSON-LD, запропонований у стандарті ISO-25964 [31], може найкращим чином відповідати завданням публікації тезаурусових ресурсів у формі веб-служб Переваги формату, який ми визначили, включають:

- найбільш задовільним є формат JSON для сприйняття людиною ,порівнюючи з іншими альтернативами;
- серіалізацію даних у формат JSON-LD, використовуючи веб-службу, достатньо просто втілити в життя завдяки широкій підтримці базового формату JSON в прикладних бібліотеках з відкритим кодом;
- втілення моделі зв'язаних даних, які являються необхідною властивістю визначеного раніше тезауруса як системи доступу до даних пов'язаних один до одного термінів, і певним чином беремо за взірець ідеї концепції RESTful HATEOAS [29] веб-сервісів, ми вважаємо, що це один з найдосконаліших промислових методів підтримки веб-сервісів;

- можливість достатньо укоротити запис без втрати досконалості визначення границь за допомогою підтримки елемента контексту для опису доменних обмежень і типів полів тіла документа.

До основних понять формату відносяться [32]:

1. IRI – інтернаціональні ідентифікатори ресурсів,
2. В основному використовується в контексті присвоєння скорочень до IRI,
3. Ідентифікатори вузлів і типізовані значення.

Запропонованих базових елементів формату повинно бути достатньо для подання даних тезаурусу у мінімально прийнятному до стандарту вигляді.

## 1.5 АНАЛІЗ НАЯВНИХ АВТОМАТИЗОВАНИХ МЕТОДІВ ПОБУДОВИ ТЕЗАУРУСІВ

Способи автоматизованого утворення тезаурусів ділять на дві основні категорії:

- Статистичні – посилене використання частотних та позиційних характеристик термінів у документах як основу різних моделей для виявлення взаємозв'язку між термінами;
- Лексикографічні - використання даних у сфері обробки людської мови за рахунок реалізації граматичного аналізу, морфологічного аналізу та інших видів аналізу тексту для пошуку семантичних зв’язків на основі інформації, отриманої лише з тексту. Водночас у словниковому методі зазвичай застосовується широко використовуваний мовний корпус, зібраний фахівцями, який містить загальні правила використання слів, форм слів та ряд синонімів, а також доступні пакети програм для первинного аналізу вільного тексту , зокрема утиліти тегування за частинами мови,

лематизатори, стемери і поготів. Також, для методів, що мають статистичний характер, такою основою є утиліти ранжування термінів та індексування. Запропонована класифікація методів на дві категорії є переважно умоглядною, а не взаємовиключною, тому в більшості досліджень технології різних методів використовуються в поєднанні, причому один метод має перевагу.

### 1.5.1 Статистичні методи

Фундаментом для багатьох статистичних методів пошуку взаємозалежних факторів між термінами є утворення індексу термінів, які описують зміст документів якнайкращім способом. У свою чергу, побудова індексу зазвичай вимагає ранжування термінів у порядку важливості, а найбільш часто використовуваними методами зважування, що походять від алгоритмів пошукової системи, є використання частоти термінів (TF), зворотної частоти документів (IDF) та їх комбінацій. Розглянемо цей метод більш докладно.

#### 1.5.1.1 Ранжування термінів використовуючи метрику TF·IDF

Метрика зважування TF·IDF, у багатьох своїх модифікаціях грала значну роль основної техніки ранжування в пошукових системах на певному етапі їх розвитку, і велика кількість досліджень з покращення формул [17] і проведених експериментів зробили дану методику досить ефективною. У більшості своїх модифікаціях метрика зважування TF·IDF відігравала вагому роль ключової техніки в пошукових системах у декотрому періоду їх розвитку,

та чимала кількість проведених експериментів й досліджень з покращення формул [17] зробили таку техніку достатньо ефективною.

Розглянемо конкретний зміст цих показників. Тому в якості міри ваги використовується лише частота термінів, а найкращими термінами в індексі є ті терміни, які мають середню частоту в документі. Найбільш повторюваними словами в документі вважається неможливість відрізнити документ від інших документів у наборі документів, а найменш вживані слова мало впливають на зміст якісного опису документа.

Метрика інвертованої документарної частоти, замість того, враховує виникнення слів у всьому наборі документів, а не в окремому документі. Тому слова, знайдені у всіх або принаймні багатьох документах колекції, вважаються невідмінними від документів, навпаки, слова, що містяться в декількох документах, вважаються для опису цих документів.

Поєднання цих методів призвело до загального використання переваг двох технік: терміни, що з'являються в документі із середньою частотою, та низькочастотні у збірнику стають дуже важливими.

Базові формули для обчислення даних метрик наведені нижче.

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

де $n_i$ відповідає кількості входжень слова, що шукається в документ, а у знаменнику – кількість усіх слів у документі.

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

де |D| - кількість документів в колекції, а в знаменнику кількість документів, де знаходиться шуканий термін.

### 1.5.1.2 Техніка спільного вжитку

Такий метод являється одним із варіантів в інформаційному пошуку задля формування багатослівних термінів, і він є статистичною альтернативою лексикографічного методу тегування за частинами мови. Основним елементом, що використовується для обчислення за цим методом, є те, як часто цей термін з'являється у певних контекстних рамках різного розміру, таких як цілі документи, розділи документів, абзаци та інші менші елементи. У цьому випадку, чим ближче слово з'являється в контексті вибраного кадру, тим частіше воно використовується як міра спільного вжитку. В дослідженні Шутца і Педерсона [39] у фундамент покладено створення матриці $C$, елементи якої $C_{ij}$ показують кількість разів сумісного вжитку між словами $i$ та $j$ в контекстній рамці розміру $k$ для досягненні такого підходу. Через великі розміри та тривалий час побудови цієї матриці, проблеми ефективності, пов'язані з цією реалізацією, очевидні. Звідси, логічно, може виникнути невпевненість щодо якості знайдених термінологічних зв'язків за згаданою технікою, як у статті Чена [23, с.178], що запевняє щодо неефективності складеного за даним методом тезаурусу при використанні до задач пошуку, і радить особистий метод, який ми розглянемо далі.

### 1.5.1.3 Метод концептуального простору

Цей метод вводить поняття концептуального простору як мережі термінів та їх зважених асоціацій, вони можуть відтворювати поняття та їх взаємозв'язки у відповідному інформаційному просторі у вигляді колекцій документів у базі даних.

Даний підхід має чотири етапи:

1. збір документів і списку об’єктів інтересу,
2. фільтрація об’єктів і автоматизована індексація,
3. аналіз методом спільного вжитку,
4. асоціативний пошук.

Етап збору документів у методі вважається досить важливим, адже базова колекція документів має бути достатньо повною для репрезентативного опису усіх домен-специфічних концептів окремо взятої області. На наступному кроці фільтрування об’єктів інтересу визначалася множина термінів, що відповідала б ключовим поняттям предметної колекції. Через те що не всі ключові поняття зазвичай включені в опис предметної області, застосовується техніка автоматизованого індексування колекції для знаходження таких термінів з тексту. В рамках індексування, в методі були заподіяні такі кроки:

- звернення до словників для ідентифікації окремих слів у тексті,
- наступне фільтрування списку слів за допомогою вилучення стопслів, що не підходять в якості кандидатів на включення до індексу,
- стемінг слів, що залишились після фільтрації,
- формування термінологічних фраз шляхом поєднання суміжних слів у документах і включення їх до індексу.

На третьому етапі, аналіз методом спільного вжитку в описаному методі починався з обчислення частот термінів і інвертованих документарних частот, з наданням переваг у зважуванні термінам в заголовках документів, ключовим словам відомим з предметної області заздалегідь і побудованим багатослівним термінам. Аналіз здійснювався на базі розробленої Ченом і Лінчем [25]

асиметричної функції кластеризації, що за доведенням авторів надає кращі результати ніж загальновживана косинусна міра схожості.

Модель асоціативного пошуку, включена в даний метод, наближена до ментальних способів представлення інформаційних потреб користувачів пошукової системи у вигляді мережі термінів і зв'язків між ними, що зазвичай є нечіткими. Авторами методу було розроблено інтерфейс навігації по побудованому тезаурусу, в основу котрого було покладено нейронну мережу Хопфілда, що в результаті своєї роботи по досягненню збіжності уточнювала ваги між термінами, таким чином наближаючи стан зв'язків між термінами до подібного за силою зв'язків стану їх представлення в людській пам'яті [24].

### 1.5.2 Лексикографічні методи

Лексикографічні методи пошуку взаємозв'язків між термінами ґрунтуються на принципах безпосереднього вказівки на зв'язок між словами за допомогою мовних засобів, а характер взаємозв'язку можна визначити на основі синтаксичної та лексичної структури висловлювань. Перш ніж розглянути основні положення цих методів, а також їх безпосередній зв'язок із формуванням термінології, необхідно подати кілька визначень.

#### 1.5.2.1 Огляд лексикографічних відомостей щодо термінології

**Терміном** (лат. terminus – «рубіж, межа»), називають слово або словосполучення, що позначає поняття певної галузі науки, техніки тощо. Основними ознаками терміну є : системність, наявність дефініції, тенденція до однозначності в межах свого термінологічного поля, тобто термінології певної галузі [12, с.629]. Зазначені ознаки наявні тільки в конкретних

терміносистемах, але за їх межами термін втрачає вказані характеристики і стає загальновживаним словом. Слід також зазначити, що основна функція терміна – номінативна та що терміни у більшості випадках позбавлені емоційного забарвлення.

**Термін** – одиниця лексичного складу мови, завданням якої є називати реалії даної спеціальності та бути підставою для наукового висловлювання.

Без цих найменувань неможливо здійснювати наукову роботу. У науковому тексті термін виступає як стабільний елемент, який приєднують до одиниць загальновживаного лексичного складу мови. Відзначаються дві основні сфери функціонування термінів: сфера внутрішньо-наукова та сфера зовнішніх зв'язків мови.

Слід мати на увазі те, що в кожній із галузей конкретний термін має інше семантичне значення, яке може збігатися із значенням у загальній мові або може відрізнятися від нього. Варто також зазначити, що на зміну значення терміна впливають мовні та і позамовні фактори.

Розглядаючи типологію термінів, для наших задач є ключовою типологія за формою, а саме:

- однослівні: кластеризація, означення, алгоритм
- багатослівні, або термінологічні словосполучення (найчастіше двослівні терміни): часова складність, соціологічне дослідження

Для задач пошуку термінів також важливо надати способи утворення термінів. Відомі такі способи утворення термінів [33]:

1) утворюванням похідних слів: додавання префіксів та суфіксів до існуючих слів;
2) складанням слів у термінологічні словосполучення

3) об’єднуванням слів у композити – складені слова

4) використанням скорочених слів або скорочень

5) перенесенням значення

6) запозиченням слів з інших мов

В розрізі лексикографічних методів пошуку зв’язків в термінології, найцікавішим для нас є спосіб складання термінологічних словосполучень.

Цей спосіб належить до найпродуктивніших способів словотвору. Два або більше слів зливаються в одне словосполучення. Окремі слова втрачають у словосполученні свою самостійність та їх питоме значення зникає. У новому сполученні його члени набувають нового змісту, який є в даній формі однозначний. Термінологічні словосполучення, як правило, поділяють на номінативні та дієслівні. Кожне термінологічне словосполучення виступає у мові як певна форма, тобто неможливо його члени пересувати або заміняти синонімами.

З поняття терміну логічно випливає означення термінології: Термінологія (лат. terminus – «рубіж, межа» і грец. λόγος – «слово, вчення»):

1) Сукупність термінів, що обслуговують певну сферу знань, пов’язаних із системою понять : мистецтво, техніка, виробництво тощо. На думку вчених, слово термін уперше з’явилося у Німеччині в 1876 р.

2) Розділ лексикології, що займається загально-теоретичними питаннями терміна [12, с.231].

В українській мові, крім поняття термінологія, досі часто користуються терміном термінознавство. Дефініція його така – «наука, яка займається

загальновживаними питаннями терміна, термінології, номенклатури» [7, с.143].

На даному етапі ми можемо побачити прямий зв'язок між тезаурусом, термінологією і лексикографією.

**Лексикографія** – словникарство, розділ мовознавства, що займається створенням словників та їх теоретичних засад [6]. У рамках лексикографії слід розрізняти лексикографічну теорію, що включає в себе основні проблеми лексикографії, з якими зустрічається лексикограф у своїй роботі та власне лексикографічну практику, тобто практичні питання щодо збору лексикографічного матеріалу, його розробку та з рештою укладання словникового тексту. Лексикографія пов'язана з іншою мовознавчою дисципліною – лексикологією. Головним завданням лексикографії є пояснення незрозумілих слів, яке здійснювалося початково у вигляді глос, тобто тлумачення написів на полях і в тексті рукописних книг. Лексикографія займається словникарським кодифікуванням лексики будь-якої мови.

Основним об'єктом лексикографії є слово. Будь-який словник містить певну кількість слів конкретної досліджуваної мови або мов, тобто необхідно з точки зору лексикографії розглядати слово і мову.

Одним із завдань лексикографії є укладання словників, адже «словник є корисним помічником для спеціальної термінологічної роботи, а також для стилізації наукового тексту, тому що в них наочно зібрані спеціальні назви та звороти» [26, с.57]. Словники відображають культуру й мову народу та є джерелом правил її написання, вимови, наголошування тощо.

Важливо, що серед типів словників, складанню котрих присвячена наука і ремесло лексикографії, почесне місце займають саме термінологічні словники.

Отже, зв'язок проблематики складання тезаурусів і використання лексикографічних методів є вкрай необхідною умовою пошуку вдалих рішень.

Адже в працях [23, 42] зазначалася одна з найпоширеніших проблем всіх статистичних методів – проблематика індексування фразових термінів, або, в нашому випадку, термінологічних словосполучень. Зокрема, автор зазначав необхідність побудови якісних рішень на основі лінгвістичних особливостей текстів, зокрема з використанням техніки тегування за частинами мови, як одну із головних задач покращення статистичних методів в інформаційному пошуку.

Для того щоб таку методику спробувати застосувати для пошуку термінів, необхідно розглянути можливі способи утворення термінів.

Розглянемо способи утворення термінів шляхом складання словосполучень. Згідно з роботою Кріслової [33, с.49], що стосується лінгвістичної україномовної термінології, можна виділити наступні схеми утворення термінологічних словосполучень за частинами мови окремих слів. Тут необхідно зробити припущення, що нижчеподана схема буде певним чином відповідати і способам творення в термінологічних системах інших галузей знань, що можливо подалі необхідно буде врахувати під час модифікації схеми.

Двослівні словосполучення:

Прикметник + іменник

*(когнітивна лінгвістика, односкладне речення, мотивована основа)*

Іменник + іменник

*(контекстуальність комунікації, методи лінгвістики, оператор порівняння)*

Тричленні термінологічні словосполучення можуть бути побудовані таким способом:

Іменник + прикметник + іменник

*(розповідь від першої особи, дієслово доконаного виду, актуалізація мовних засобів)*

Прикметник + прикметник + іменник

*(додатковий орієнтаційний момент, дієслівне двоскладне речення, актуальний теперішній час)*

Прикметник + іменник + іменник

Крім двочленних та тричленних словосполучень, досить часто в термінології виникають чотиричленні терміни, структура котрих може виглядати:

Прикметник+ іменник + прикметник + іменник

*(словниковий склад літературної мови, стилістичне транспортування мовних засобів, позачасове вживання теперішнього часу, односкладне речення з допоміжним дієсловом)*

Прикметник + іменник + іменник + іменник

*(аналітичне творення ступенів порівняння)*

Іменник + прийменник + іменник + іменник

*(графіка із застосуванням диграфів, прикметник із значенням можливості )*

Важливість саме іменникових словосполучень, що відповідали б термінам, підтверджується і в праці Лендау, де зазначається, що серед науково-технічної літератури і в термінологічних словниках більшість термінів виражені іменниками [9, с.173].

Подальші схожі роздуми побачимо в методі Г. Грефенстета.

### 1.5.2.2 Метод генерації тезаурусу з використанням техніки браку знань

Даний метод був вперше описаний і застосований Грегорі Грефенстетом [28]. Покладені в основу методу техніки видобування термінів і зв'язків між ними для побудови початкової версії тезаурусів з будь-якої колекції специфічних за областю текстів названі техніками з браком знань через те, що для побудови такого тезаурусу наявність будь-яких знань про область знань є зовсім не необхідною, і така побудова виводиться першочергово з наявних текстів. Дослідники застосували даний метод до більш ніж 20 різних корпусів текстів, розміром від 1 до 6 мегабайт, і отримали позитивні результати. Розглянемо ж лексикографічні закономірності, покладені в основу даних технік більш детально.

Відкритий текст колекції документів в першу чергу розбивався на токени за регулярною граматикою. Потім кожен токен аналізувався, і до нього з лексикону мови добиралися теги частин мови і їх основних параметрів, що мав у собі токен. Далі тегований текст позбавляли неоднозначності стохастичним методом, так щоб кожен токен мав тільки одну форму частини мови і один тег. Після цього готовий текст розбирався, і збиралися зазначені у форматі тезаурусу залежності між словами.

Для іменників, збирались як атрибути всі модифікуючі прикметники і дієслова, для котрих дані іменники були суб'єктами або об'єктами, а також всі інші іменники, що відносились до них як додатки або входили в умовні звороти. Схожість між іменниками обчислювалась за допомогою зваженої метрики Жаккарда [16, с.260]. В результаті порівнянь для кожного іменника зберігався сортований список найбільш схожих на даний термін слів з тексту. Для скорочення такого списку застосовувався фільтр, що залишав тільки ті

слова, що є взаємно найбільш схожими, тобто щоб кожен з термінів містився в рамках 10 найбільш схожих слів в обох списках. Цікаво, що з результуючому списку схожих слів автори використали порівняльну частоту входження термінів у якості основи для побудови ієрархічних зв'язків в тезаурусі, зазначаючи що даний обраний метод є надзвичайно слабким і неточним.

Для виокремлення загальновживаних фраз, або термінологічних словосполучень, автори використали тільки модель з двослівних словосполучень, утворених іменниками, спираючись на проблематику пошуку більш довгих підпослідовностей у тексті. Між іншим, серед усіх таких фраз в результуючий тезаурус потрапили тільки 10 найбільш вживаних, кожна з котрих зустрічалася в тексті немодифікованою принаймні 3 рази. Щоб підібрати список синонімічних фраз, для зважування був обраний інший метод, що брав до уваги ширший за синтаксичний контекст, що виходив за рамки окремих словосполучень. Як показали експерименти, для фразових словосполучень, частота вживання котрих значно менша, адекватну оцінку надавав тільки аналіз ширшого контексту, тобто цілих речень.

Окрім окреслених груп слів, в даному методі також відбувався пошук дієслів і сімейств слів. У підсумку, дослідникам вдалося побудувати 20 різних тезаурусів на різних тематичних колекціях текстів, для чого було застосовано 3 різних техніки: використання локальної лексико-синтаксичної інформації для побудови зв'язків близькості між термінами, використання більш ширшого контексту речення для пошуку двослівних фраз, і насамкінець присутність у всьому документі для виявлення сімейств термінів. В реалізації лінгвістичних підходів не було використано складних лінгвістичних закономірностей, таких як семантичні маркери на іменниках, або детальний морфологічний аналіз, і все ж таки отримані результати говорять на користь ефективності застосованих технік.

### 1.5.2.3 Гіпонімія як основа для пошуку ієрархічних зв'язків

В наступному лексикографічному методі ключову роль грає поняття гіпонімії, котре ми розглянемо тут детальніше.

Гіперо-гіпонімія – це родо-видові відношення в лексико-семантичній системі [4, с.206]. Наприклад: береза, дуб, клен, явір, сосна - дерево; троянда, рожа, настурція, тюльпан, нарцис - квітка. Родові слова називають гіперонімами, а видові — гіпонімами.

Гіперо-гіпонімія близька до синонімії. ЇЇ навіть називають, але на відміну від синонімії, яка допускає двосторонню заміну в тексті (першого синоніма на другий і навпаки), в гіперо-гіпонімії можлива тільки одностороння заміна — заміна гіпоніма на гіперонім.

Зрозуміло, що явище гіпонімії є безпосереднім вказівником в тексті на зв'язок типу "загальне-конкретне" між термінами, котрий є невід'ємною складовою тезаурусів. Розвиваючи дану ідею, розглянемо статтю М. Хеарста [30], котрий описав автоматизований лексикографічний метод видобування гіпонімів з тексту.

### 1.5.2.4 Метод збирання гіпонімів з великих текстових колекцій

Дві головні цілі, вказані даним підходом – елімінація потреби у попередньо складених знаннях про предметну область і можливість застосування методу на широкому колі текстових колекцій. В роботі було виокремлено множину лексико-синтаксичних шаблонів, що є широко розповсюдженими серед колекцій текстів різних жанрів, що безпосередньо

вказують на шукані лексичні залежності, і котрі легко розпізнати в тексті як програмними засобами, так і власноруч.

В центрі методу полягає ідея про вміст великої кількості корисної інформації про предметну область в самому тексті, що може бути виявлена як людиною, так і алгоритмом, не вдаючись до занадто конкретних деталей окреслених явищ і речей, тобто не потребуючи від системи здійснення глибокого лексикографічного чи семантичного аналізу. Наприклад, з речення: “Мови програмування, такі як Java, C++ і Smalltalk, відносяться до класу імперативних об’єктно-орієнтованих мов”, - людина чи алгоритм, що не має ніякого уявлення про окремі мови програмування, з тексту знає про відношення між поняттями, узагальнювальне поняття, а також окремих представників. Подане в формальній лекико-синтаксичній нотації авторів статті, попереднє висловлювання набуде вигляду:

$$NP_0, такі\ як\ \{NP_1, NP_2\ ...\ ,\ (або\ |\ і)\}\ NP_n$$

де $NP_i$ – відповідає певному термінологічному іменниковому словосполученню, що потребує, щоб для всіх $NP_i$, $1 < i < n$, виконувалось твердження

$$hyponym(NPi, NP0)$$

З речення-прикладу можна зробити висновок, що виконується

*hyponym(“Java”,”мова програмування”), hyponym(“C++”, “мова програмування”), hyponym(“Smalltalk”, “мова програмування”)*

Дану техніку пошуку таксономічних зв’язків автори пов’язують з попередніми дослідженнями, зокрема з роботою Алшаві [18], що використав ієрархію шаблонів для інтерпретації означень, що складалися переважно з індикаторів частин мови і символів-масок. Основним недоліком такого підходу автори вважають проблему підбору такої множини шаблонів, що б з

однаковою точністю вказували б на спрямованість зв'язку в текстах різних стилів. З основних прикладних напрямків, де застосування пошуку гіпонімії є виграшним, виділяють сферу приросту і перевірки корпусів мов, складених людиною (наприклад, WordNet), сфери пошуку приблизних значень незнайомих іменників в термінологічних словосполученнях і семантичної схожості між словосполученнями, що стоять в одному ряду підпорядкування з узагальнювальним гіперонімом.

Серед знайдених авторами методу експериментальним чином надійних шаблонів, в статті було подано шість окремих формальних нотацій таких шаблонів з прикладами. Не обмежуючись в широті застосування даних шаблонів, використавши ті самі граматичні правила розробленої авторами нотації для скорочення кількості шаблонів, здійснивши переклад ключових слів на українську мову, ми подамо зазначені шість шаблонів у вигляді трьох формальних правил:

1. **такі NP як {NP,}* {(або | і)} NP**

роботи таких дослідників як Хеарст, Грефенстет і Ашаві.

*⇒ hyponym("Хеарст", "дослідник"), hyponym("Грефенстет", " дослідник"), hyponym("Ашаві", " дослідник")*

2. **NP {, NP} * {,} {(або | і) інші NP**

Набряки, гематоми, рани і інші травми

*⇒ hyponym("набряк", "травма"), hyponym("гематома", "травма"), hyponym("рана", "травма")*

3. **NP {,} (зокрема | особливо) {NP,}* {або | і} NP**

Всі демократичні країни, зокрема Канада і Великобританія

*⇒ hyponym("Канада", "демократична країна"),*

*hyponym("Великобританія", "демократична країна")*

Серед помічених проблем реалізації даного методу, дослідники зазначають проблему вилучення інфінітивної форми термінологічного словосполучення з тексту. Адже навіть при використанні лематизації слів, що входять в склад словосполучення, необхідно знову розв'язати задачу узгодження слів в словосполученні, що може потребувати введення великої кількості додаткових правил і винятків. Якщо ж дані поліпшення вилучених з тексту словосполучень не проводити, якість складених у словник предикатів відношень значно втрачається. Ті ж самі, і навіть більш ширші проблеми ми зустрінемо під час застосування таких шаблонів для україномовних текстів.

Серед описаних проблем узгодження слів і приведення до нормальної форми словосполучень, зустрічалися наступні:

- іменники всередині словосполучень, що подані в множині;
- числові модифікатори прикметників, такі як "деякі", "інші", можуть бути усунуті без втрати смислу;
- порівняльні слова, так як "важливий", "менший" – найкращі кандидати на усунення з фрази.

Щодо доречності усування модифікаторів іменників у словосполученні, автори доходять висновку про залежність даного фактору від широти вживаності тезаурусу, що будується. Якщо в статті основні експерименти проводились на тезаурусі загального вжитку, і було зроблено висновок про найчастіше використання однослівних іменників в якості елементів таксономічних відношень, то для більш спеціалізованих колекцій, зокрема наукових публікацій, більш характерними будуть багатослівні термінологічні словосполучення.

Експериментальні запуски реалізованого методу на енциклопедичних колекціях різної довжини показували загалом прийнятні результати щодо

кількості знайдених збігів з шаблонами, і отриманих зв'язків. Тестування точності отриманих зв'язків проводилось за рахунок порівняння з уже розміченими людиною парами значень в корпусі англійської мови WordNet.

Підсумовуючи основні здобутки даного методу, можна вказати на порівняльну дешевизну застосування даного методу для автоматизованого збору семантичних лексичних зв'язків в вільно текстових документах. Даний метод позиціонується як альтернатива статистичним методам, і має перед ним перевагу в точності роботи на рідких зв'язках між термінами, що зустрічаються в тексті поодиноко, і не можуть буті оброблені вдало статистичними методами. Представлені в дослідженні шаблони і стратегії відсікання модифікаторів іменників не претендують на повноту, і залишають певну свободу для майбутніх доповнень.

# РОЗДІЛ 2

# МЕТОДИ ФОРМУВАННЯ ТЕЗАУРУСА

## 2.1 СТРУКТУРНА СХЕМА АЛГОРИТМУ

### 2.1.1 Індексування і фільтрація важливих термінів

Процес побудови термінології на основі колекції текстів можна розкласти на два принципових кроки:

1. по-перше, видобування зі всіх слів, що зустрічаються в текстах документів, таких, що відповідають термінам в області знань відповідних документів;
2. по-друге – встановлення на множині даних термінів відношень, що використовуються в тезаурусах, зокрема симетричне відношення пов’язаних термінів (RT – Related Term), і асиметричні відношення типу “частина-ціле” (BT – Broader Term, і NT – Narrower Term відповідно).

Завдання виокремлення термінів з множини усіх слів документа смисловим чином є подібним до звичайної операції індексування текстів пошуковими системами. Під час такого індексування здійснюється побудова таблиці відповідності ідентифікатора індексованого документа з певним списком ключових слів, що найкращим чином розкривають зміст даного документа, є його ключовими словами. Індексація може проводитись як експертами власноруч, так і автоматизовано за допомогою різних алгоритмів. Найбільш вживаний алгоритм індексування базується на зважуванні слів на основі їх частоти присутності в документах.

Частота терміну (TF), інвертована документарна частота (IDF), а також їх комбінація TF·IDF є найбільш широко вживаними техніками зважування.

Отримавши зважену послідовність слів за такою методикою, відсортовану по спаданню ваги, на початку послідовності будемо мати слова, що найкращим чином характеризують зміст документів, а отже є кандидатами в терміни.

Слід зауважити, що ефективність такого зважування, особливо під час обрахунку метрики IDF, цілком залежить від розміру і різноманіття документів колекції. Якщо ж колекція документів, на основі якої відбувається пошук термінів, є досить невеликою, даний метод буде надзвичайно чутливим до стилю тексту і окремих вживаних слів певної предметної області, і, таким чином, не зможе виявити кандидатів в терміни з достатньою точністю.

На даному етапі, слід повернутися до поставленої задачі, де ми на першому кроці намагаємось обмежити кількість слів, по потраплять у список термінології, і застосувати загальновживаний метод зважування на основі TF·IDF для наших потреб.

Отже, для обмеження даного відсортованого списку слів можна ввести обмежувальний оператор, що надав би можливість визначити граничний елемент списку, після котрого починається перелік загальновживаних слів як наукового стилю текстів, так і текстів загальної тематики.

Дана функція може мати наступні запропоновані варіації для нашого методу:

1. “Проходять усі” – хвіст послідовності не відкидається, всі слова документів інтерпретуються як терміни, що беруть участь у пошуку зв’язків. Очікувані недоліки застосування даного оператора – значне збільшення простору пошуку і, відповідно, часової складності обрахунків, а також наявність в кінцевому тезаурусі великої кількості зайвих, таких, що не мають практичної цінності зв’язків між словами

текстів. Однак, результат застосування може бути використаний на етапі досліджень інших методів обмеження, наведених нижче, для порівняння результатів їх застосування із базовим варіантом.

2. “Стоп-список” – параметризований оператор, що відкидає задану наперед параметром кількість слів у хвості послідовності. Даний метод використовує один з популярних методів вилучення стоп-слів в пошукових системах, проте залишається чутливим до розміру колекції текстів. За допомогою такого відсіювання, наприклад, 100 слів в кінці послідовності, можна позбутися певних загальновживаних слів, проте навряд вдасться здійснити відокремлення загальновживаних слів наукового стилю мовлення.
3. Пропорційний метод – подібний до оператору “стоп-список”, з обмеженням в якості параметру певного відсотку слів у хвості послідовності. Ґрунтується на методологічній засаді про відомість статистичного розподілу термінів в колекціях наукових текстів.
4. Метод найшвидшого спуску. Базується на запропонованій нами гіпотезі значного стрибку функції розподілу частот термінів при переході з загальновживаних слів до специфічних термінів. Підхід полягає в послідовному аналізі підпослідовностей слів розміру k, і обранні в якості бар’єру останнього елементу такої підпослідовності, що має найбільший стрибок частоти серед усіх інших.

Під час дослідних експериментів було вирішено зупинитись на пропорційному підході до обмеження вхідного списку термінів, що б надавав принаймні першочергове відкидання загальновживаних слів, і мав би зручний для варіації під час експериментів параметр відсотку відсічу. Однак для розроблюваного методу даний вибір не є остаточно зафіксованим, і програмна

реалізація методу може мати усі наведені вище способі обмеження списку вхідних слів, разом із їх комбінаціями у разі потреби.

### 2.1.2 Зважування термінів за метрикою документарної частоти еталонної колекції

Зрозуміло, що наведені способи обмеження списку слів будуть працювати лише за умови застосування надійної схеми зважування, що в свою чергу, в нашому випадку, буде залежати від способу підрахунку складової документарної частоти термінів, чутливої до складу і розміру колекцій.

В нашій роботі проблему замалих колекцій текстів для надійного зважування запропоновано вирішувати шляхом побудови компоненту довідкової системи документарних частот термінів.

Тут можна виділити два підходи:

- Побудова і індексація великої і різноманітної навчальної колекції текстів наукової тематики, з наступним зберіганням отриманих документарних частот як еталонних. В свою чергу, в якості документарної основи для такої колекції, можна запропонувати випадкову вибірку 1000 україномовних публікацій з системи Google Scolar.
- Звертання до вже існуючих частот термінів в великих пошукових системах. Зокрема, можна за основу документарної частоти брати параметр кількості знайдених результатів при пошуку ключового слова. До переваг даної методики слід віднести як власне вже готовність до використання (частоти пораховані), так і те, що покладена в основу даної пошукової системи колекція документів надзвичайно велика, таким чином можна зробити припущення про надійне порівняння між собою

документарних частот слів, отриманої у цей спосіб. До недоліків належить складність і ненадійність доступу (для роботи алгоритму необхідний постійний широкосмуговий доступ в Інтернет, реалізація пошукового скрипту буде використовувати або забагато зайвих даних під час довантаження і розбору веб-сторінок, або потраплятиме під квоти використання відповідного програмного інтерфейсу), а також непостійність результату бо система динамічна.

### 2.1.3 Метод пошуку зв'язків між термінами

Після отримання першочергового списку термінів, для складання тезаурусу необхідно віднайти характер і направленість зв'язків між термінами.

В даній роботі вводиться поняття характеристичного фрагменту тексту, що є безпосереднім входженням терміну в документ у певному контексті, зокрема як члену відповідного речення. З-поміж багатьох методів розгляду контексту вживання слів, як-от частин оточувальних словосполучень і зворотів, речень, чи взагалі вікон з фіксованим розміром кількості слів, ми обрали саме речення в якості основи для наших досліджень, виходячи з наявних інструментів, що дозволяли б застосувати методику тегування за частинами мови в якості основи для лексикографічних методів.

Отже, наступним кроком є знаходження характеристичних фрагментів тексту для усіх термінів зі списку. Даний пошук може бути здійснений лінійно, проте маючи на увазі можливість масштабування розробленого методу, запропоновано використати одну з пошукових систем з відкритим кодом, що повертала б всі документи з нашої колекції що містять певний термін, таким чином обмежуючи простір лінійного пошуку. Далі, серед знайдених документів здійснюється пошук характеристичних фрагментів – речень, в які входить даний термін.

Наступним кроком здійснюється аналіз всіх знайдених характеристичних фрагментів, із застосуванням різних методик для визначення типу зв'язку:

1. Застосування найпростішого методу спільного вживання термінів всередині одного характеристичного фрагменту. Даний метод дозволяє встановити зв'язок пов'язаних термінів (RT), якщо дані терміни входять в характеристичні фрагменти тексту разом з початковим терміном.
2. Застосування множини визначених лексикографічних шаблонів, що дозволяють віднайти зв'язки типів BT, NT і RT.

### 2.1.4 Розширення тезауруса термінологічними словосполученнями

Застосування лексикографічних шаблонів базується на методі пошуку іменникових термінологічних словосполучень, що в свою чергу, у разі збіжності з текстом, дозволяє виокремити не тільки однослівні терміні, але і такі що подаються декількома словами, котрих є набагато більше. Таким чином, побічним продуктом застосування лексикографічних шаблонів є розширення першочергового списку термінів термінологічними словосполученнями, чого не можна було досягти на першому етапі тільки за рахунок індексування в рамках використаних інструментів.

### 2.1.5 Методи пошуку збіжності за лексикографічним шаблоном

Для застосування визначених нами лексикографічних шаблонів, введемо таку формальну нотацію:

Лексикографічний шаблон (LP) – впорядкований список операторів збіжності.

Оператор збіжності – команда, що вимагає застосування операції пошуку збіжності типу іменникового словосполучення (NP – Noun Phrase), або конкретного слова чи символу з синонімічного ряду (EW – Exact Word).

NP – оператор збіжності, що виконує пошук іменникового словосполучення за рахунок застосування вказаних для кожного такого оператору списку правил збіжності по частинах мови. Повертає в якості результату всі знайдені у фразі іменникові словосполучення в порядку даних правил збіжності, а також позиції знайдених іменникових словосполучень у фразі. До операторів збіжності даного типу в якості параметру можна задати їх роль (індекси 1 і 0).

Роль оператора NP - індекс 1 або 0, що вказує на головну або другорядну роль даного оператору в шаблоні (записується як NP1 або NP0).

EW – оператор збіжності, що здійснює пошук входження конкретного символу або слова в фразу зі списку можливих альтернатив, повертає позиції входжень таких слів.

W – оператор вікна, що вказує мінімальні і максимальні рамки розміру вікна, що має роль маски збіжності з будь-якими підпослідовностями слів в реченні.

IT – оператор ітерації, що позначає повторювану послідовність операторів в шаблоні.

Правило збіжності (MR) – задана послідовність тегів частин мови, котрій має відповідати підпослідовність слів у реченні.

Теги частин мови (N, A, P) – параметри конфігурації правил збіжності для виокремлення термінологічних словосполучень, що позначають іменник (N), прикметник (A), і прийменник (P) відповідно.

Задовільнення шаблону – знаходження множини задовільняючих операторів збіжності підпослідовностей слів, де кожна позиція такої підпослідовності відповідає як порядку входження у фразу, так і порядку

оператора, визначеного у шаблоні. Тобто, всі можливі збіги по окремим операторам мають бути об’єднані в результуючу множину шляхом обмеження по слідуванню правил.

Наприклад, щоб зафіксувати в нашій формальній нотації лексикографічний шаблон 1, що відповідає за прямі означення з використанням тире, маємо записати наступне.

$$LP = (NP_0(MR<A,N>), EW(“–”,”-”), NP_1(MR<N,N>))$$

Даний шаблон має задовільнити наступна фраза:

*“Соціологічне дослідження — система процедур для отримання наукових знань про соціальні явища і процеси”.*

При цьому першому оператору збіжності буде відповідати термінологічне словосполучення “соціологічне дослідження”, оператору збіжності по слову було надано дві альтернативи – власне символ “тире”, а також дефіс, для обробки випадків заміни даного символу у вхідному тексті, останньому оператору відповідає словосполучення “система процедур”.

Таким чином, оператори збіжності типу EW у шаблоні грають роль фіксованих точок шаблону, в той час як оператори NP – роль наповнюваних змінних, що видобувають словосполучення з фрази під час задовільнення шаблону.

### 2.1.6 Інтерпретація зв’язків за збіжністю тексту з шаблоном

Під час складання шаблону, до параметрів NP додатково вказується параметр головної чи другорядної ролі в шаблоні, що інтерпретують зв’язки між отриманими збігами по NP наступним чином:

- між представниками $NP_0$ і $NP_1$ встановлюється зв’язок BT;
- між представниками $NP_1$ і $NP_0$ встановлюється зв’язок NT;
- між представниками однакових ролей встановлюється зв’язок RT.

Підґрунтям для такої інтерпретації є те, що у більшості шаблонів на відповідних місцях термінологічних словосполучень за частинами речення бувають або однорідні означення чи додатки, або узагальнювальні слова, або, наприклад, у разі збіжності з шаблоном прямих означень у тексті – відповідно термін і його родова приналежність. Таким чином, у тексті в разі збіжності з шаблоном направленість зв'язку є чітку визначеною.

### 2.1.7 Підходи до отримання фразових іменникових словосполучень з тексту

Для реалізації алгоритму пошуку гіпонімів, спершу необхідно навчити систему розпізнавати фразові словосполучення. Пропонований підхід - фіксація іменників у реченні, з наступним добиранням навколишніх слів за правилами.

За Херстом, відношення між термінами "загальне-часткове" відповідають гіпонімічним зв'язкам у тексті. І цьому досліднику вдалося виокремити підмножину шаблонів, що була б достатньо точною і виконуваною для більшості текстів, враховуючи відмінності в формуванні термінології в окремих сферах інтересів.

Зважаючи на схожість наукового стилю на міжнародному рівні, здається вдалою думка про локалізацію знайдених шаблонів для української мови, з доданням нових.

Для того щоб звузити рамки дослідження і досягти певного результату для специфічних, проте найбільш уживаних способах творення термінології, до розгляду було залучено тільки терміни-іменники і іменникові словосполучення. Таке рішення було прийнято, виходячи з тих припущень, що більшість термінів утворено саме поодинокими іменниками або

термінологічними словосполученнями, тому і зв'язки в тексті слід шукати між іменниками і словосполученнями.

При цьому пропонується використати каскадний підхід, за котрим спочатку буде проводитись пошук найбільш широких і неточних зв'язків між усіма словами в реченні, потім серед іменників в реченні, потім між оточувальними ці іменники словосполученнями.

Під позначенням NP в записі шаблону слід розуміти іменникове словосполучення, що може складатись як з одного іменника, так і набувати набагато складніших форм.

З шаблонів, що відповідають за зв'язки між термінами у реченні, було обрано наступні категорії:

1. Прямі означення і дефініції, з використанням характерних для української мови знаків пунктуації і слів-зв'язок ( тире, слова "є",

   "вважається", "слід розуміти" і т. д.)

   Наприклад: NP (– | -) (це | є | вважається | означає) NP
2. Шаблони за Херстом, а саме:

   a. такий NP як {NP ,}* {(і | або)} NP

   b. NP {, NP} * {,} (або | і) інший NP

   c. NP {,} (включаючи | особливо) {NP,}* {і | або} NP
3. Шаблони на позначення зв'язків частина-ціле

   Наприклад: NP (є частиною | входить в | складається з) NP

Всі подані шаблони розширюються синонімічними і схожими за вживанням словами в формулах шаблону. Під час співставлення речень з шаблоном відбувається приведення всіх слів до нормальної форми, що дозволяє зменшити необхідну кількість варіацій шаблону.

### 2.1.8 Етапи експериментального застосування лексикографічних шаблонів

1. Відповідно до оглянутої літератури і відомостей з лексикографії, побудова і тестування на простих прикладах-реченнях шаблонів, що дозволять виокремити термінологічні словосполучення
2. Тестування на простих реченнях шаблонів, що виокремлюють зв'язки між термінами (термінологічними словосполученнями)
3. Тестування побудованих шаблонів на визначеннях з термінологічних словників. Оцінка точності запропонованих методів.
4. Тестування на тематичних колекціях

Під час застосування правил, враховується їх черговість, таким чином в першу чергу віднаходяться і потрапляють в якості елементів збігу ті іменникові словосполучення, що є ширшими за кількістю слів, а отже рідшими за вживанням.

Таким чином для налаштування підсистеми пошуку термінологічних словосполучень, необхідно правильно підібрати не тільки правила їх формування по частинах мови, а ще й послідовність застосування.

## 2.2 МАТЕМАТИЧНА МОДЕЛЬ І ФОРМАЛІЗАЦІЯ МЕТОДУ

### 2.2.1 Формальні позначення

$\mathbb{D}$ – множина текстових документів.

$\mathbb{LP}$ - множина лексикографічних шаблонів.

$\mathbb{T}$ - множина термінів тезауруса.

$T_F$ – відсортований за метрикою $TF \cdot IDF$ і обмежений функцією $limit(T)$ список важливих однослівних термінів колекції

$T_E$ – множина багатослівних термінологічних словосполучень.

$\mathbb{R}$ - множина зв'язків тезауруса,

$R_i \in \{(T_1, T_2, Rel)$, де $Rel \in \{RT, BT, NT\}$, і $T_1, T_2 \in \mathbb{T}$

$\mathbb{C}_t$ - множина характеристичних фрагментів тексту для терміна t.

$\mathbb{S}_C$ - множина речень характеристичного фрагменту C.

$Lem_S$ – множина лематизованих слів речення S

$M_{lp}$ – послідовність збіжних з лексикографічним шаблоном термінологічних словосполучень.

$limit(T)$:$\{t|t \in T_s\} \rightarrow \{t'|t' \in T_F\}$ – функція обмеження відсортованого списку термінів.

$extract(d)$:$D \rightarrow \{t|t \in T\}$ - функція видобування термінів з документу.

$sort(T,d)$:$\{T\} \times D \rightarrow (t')_1^{|T|}, \forall t_i, t_j, i < j \Leftrightarrow tf(t_i, d) \cdot idf(t_i) \geq tf(t_j, d) \cdot idf(t_j)$ - функція, що будує послідовність відсортованих термінів документа за спаданням метрики TF·IDF.

$tf(t, d)$:$T \times D \rightarrow \boldsymbol{R}$ - функція обчислення частоту терміну в документі.

$idf(t)$:$T \rightarrow \boldsymbol{R}$ – функція, що ставить кожному терміну у відповідність його інвертовану документарну частоту з еталонної колекції.

$findCF(t)$:$T \rightarrow \{c \in \mathbb{C}_t\}$ - функція пошуку характеристичних фрагментів терміна.

$split(c)$:$\mathbb{C}_t \rightarrow \{s|s \in \mathbb{S}_C\}$ - функція розбиття характеристичного фрагменту тексту на речення.

$lem(s)$:$\mathbb{S}_C \rightarrow (lem|lem \in Lem_S)$ - функція отримання послідовності лем з речення.

$match(lp,s)$:$\mathbb{LP} \times \mathbb{S}_C \rightarrow \{m|m \in M_{lp}\}$ - функція задовільнення шаблону, що повертає множину послідовностей співпавших термінологічних словосполучень в порядку слідування позицій шаблону.

$inferRelations(M_{lp})$:$\{m|m \in M_{lp}\} \rightarrow \mathbb{R}$ - функція встановлення зв'язків на множині послідовностей збігів до шаблону.

### 2.2.2 Псевдокод розробленого методу

Рисунок 1. Формальний запис методу побудови тезауруса

1. $\forall d \in \mathbb{D}$:
   $T_i \coloneqq extract(d)$
   $(t)_{i=1}^{|T_i|} \coloneqq sort(T_i, d)$
   $T_F \coloneqq T_F \cup limit((t)_{i=1}^{|T_i|})$
2. $T_F \coloneqq limit(sort(T_F))$
3. $\forall t \in T_F$:
   $\mathbb{C}_t \coloneqq findCF(t)$
   $\forall c \in \mathbb{C}_t$:
   $\mathbb{S}_C \coloneqq split(c)$
   $\forall s \in \mathbb{S}_C$:
   a. $\forall lem \in lem(s)$: якщо $lem \in T_F$: $\mathbb{R} \coloneqq \mathbb{R} \cup (t, lem, RT) \cup (lem, t, RT)$
   b. $\forall lp \in \mathbb{LP}$:
   1. $M_{lp} \coloneqq match(lp, s)$
   2. $R_{lp} \coloneqq inferRelations(M_{lp})$; $\mathbb{R} \coloneqq \mathbb{R} \cup R_{lp}$
   3. $T_E \coloneqq T_E \cup terms(M_{lp})$
4. $\mathbb{T} \coloneqq T_F \cup T_E$
5. $Thesauri \coloneqq (\mathbb{T}, \mathbb{R})$

### 2.2.3 Формалізація правил збіжності з лексикографічним шаблоном

$\mathbb{LP}=\{(pe)_1^l | pe \in PE\}$ - множина лексикографічних шаблонів, задана як множина елементів шаблону.

$PE= \{NP_0, NP_1, EW, W, IT\}$ – елементи шаблону.

$NP_0=\{((mr)_1^m, 0) | mr \in MR\}$; $NP_1=\{((mr)_1^m, 1) | mr \in MR\}$ – множини команд пошуку термінологічних словосполучень, з вказанням головної (1) чи вторинної (0) ролі словосполучення у зв'язку в шаблоні.

$MR = \{(tag)_1^k | tag \in \{'N', 'A', 'P'\}\}$ – множина правил збіжності задана послідовностями тегів за частинами мови.

$EW = \{(ew)_1^n | ew \in Lem\}$ – множина команд пошуку прямої збіжності зі словом, задана на послідовностях альтернатив лем.

$W = \{(min, max) | min, max \in \mathbb{N}\}$ множина команд пошуку вікон, що задана парами мінімальної і максимальної довжини вікна в кількості позицій лем у реченні.

$IT = \{(it)_1^t | it \in PE\}$ - множина команд пошуку ітерацій, що задана на підпослідовностях елементів шаблону.

$P_m = \{((l)_1^v, p) | l \in Lem, p \in \mathbb{N}\}$ – множина фразових збігів, задана парами послідовностей лем і позиції першої леми.

$M_{lp} = (p)_1^v | \exists lp = (pe)_1^l, \forall s \in S, \forall p_i \in apply(pe_j, s), pe_j \in \{NP_0, NP_1\}$

послідовність фразових збігів по операторам $NP_0$, $NP_1$ шаблону.

$apply(pe, s)$:PE×$S$→ $\{p | p \in P_M\}$ - функція співставлення елементу шаблону з фразою, що ставить у відповідність множину фразових збігів.

$$match(lp, s): \mathbb{LP} \times \mathbb{S}_C \to \{m' | m' \in M_{lp}\}, \text{коли } \forall lp = (pe)_1^l,$$
$$\exists (m)_1^l | m \in apply(pe_i, s), \text{таке що } \forall m_i, m_j, i < j \Leftrightarrow m_i = ((l)_1^{vi}, p_i), \quad m_j = ((l)_1^{vj}, p_j), p_i \le p_j, i \{m'_{\overline{1,n}}\} \subseteq \{m_{\overline{1,l}}\}$$

$$infer Relations(M_{lp}): \{m | m \in M_{lp}\}$$
$$\to \left\{ r \;\middle|\; \begin{array}{c} r = (T_1, T_2, BT), коли\ \exists lp = (pe)_1^l, \exists s_1, \exists pm_1 \in apply(pe_i, s_1), pm_1 = (T_1, p_1), \\ \text{і } \exists s_2, \exists pm_2 \in apply(pe_j, s_2), pm_2 = (T_2, p_2), \text{і } pe_i \in NP_1, pe_j \in NP_0, \\ або\ r = (T_1, T_2, NT), коли\ (pe_i \in NP_0, pe_j \in NP_1) \\ або\ r = (T_1, T_2, RT), коли\ (pe_i, pe_j \in NP_0, або\ pe_j, pe_j \in NP_1) \end{array} \right\}$$

## 2.3 Перелік розроблених лексикографічних шаблонів

Таблиця 1. Перелік розроблених лексикографічних шаблонів у формальній нотації

| Назва | Формальний запис правил шаблону |
|---|---|

| | |
|---|---|
| MR1-9 | *MR<NPNN>,MR<ANNN>,MR<ANAN>,MR<ANN>, MR<AAN>,MR<NAN>,MR<NN>,MR<AN>,MR<N>* |
| LP1 | *NP1,EW<'–'\|'-'>,EW< 'це' \| 'є' \| 'вважається' \| 'означає' >,NP0* |
| LP2 | *EW<'такий'>,NP1,EW<'як'>,IT{NP0,EW<',">},EW<''і\|'або'\|'й'\|'та'>,NP0* |
| LP3 | *NP0,IT{EW<',">,NP0},EW<'і'\|'або'\|'й'\|'та'>,EW<'інший'>,NP1* |
| LP4 | *NP1,EW<',">, EW<'включаючи'\|'зокрема'\|'особливо'>,IT{NP0, EW<',">},EW<'і'\|'або'\|'й'\|'та'>,NP0* |
| LP5 | *NP0,W<0,3>, EW<'бути частиною' \| 'входити в'>,W<0,3>,NP1* |
| LP6 | *NP1,W<0,3>, EW<'складатися з'\|'підрозділятися на'>,W<0,3>, IT{NP0,EW<',">}, EW<'і'\|'або'\|'й'\|'та'>,NP0* |

## 2.4. Висновки до розділу 2

Отже, підсумовуючи вище викладене, можна визначити, що вбудований в Android Studio інструментарій автоматизації Gradle дозволяє полегшити процес збірки проєкту та забезпечує детальною інформацією про хід і результати виконаної збірки. Це дозволяє аналіз та, за потреби, виявити та усунути існуючі недоліки. Мобільний додаток тачпада MobileMouse використовує два елементи компонента Intent-Activity та один об'єкт.

# РОЗДІЛ 3

# ПРАКТИЧНА ЧАСТИНА

## 3.1 ЗБІР ДАНИХ ДЛЯ ТЕСТУВАННЯ РОЗРОБЛЕНОЇ СИСТЕМИ

Етап початкового збору даних є необхідним для побудови початкової довідкової системи документарних частот термінів, а також знадобиться під час проведення тестування алгоритму на точність складання тезаурусів. Для тестування застосунку використовувалась локальна версія популярного веб-ресурсу «Wikipedia». Всього з даного ресурсу було проімпортовано більше 1000 текстів, які містили інформацію про різні теми зв'язані з математикою. Загальна кількість термінів склала більше 1000 штук. Кількість означень для кожного терміна варіюється від 1 до 5.

## 3.2 РЕГУЛЯРНИЙ ВИРАЗ ДЛЯ ВИДІЛЕННЯ ТЕРМІНІВ ТА ЇХ ОЗНАЧЕНЬ В ТЕКСТІ

В програмуванні, регулярний вираз (від англ. regular expression, скорочено regex або regexp, а іноді ще й називають rational expression ) — це рядок, що описує або збігається з множиною рядків, відповідно до набору спеціальних синтаксичних правил. Вони використовуються в багатьох текстових редакторах та допоміжних інструментах для пошуку та зміни тексту на основі заданих шаблонів. Багато мов програмування підтримують регулярні вирази для роботи з рядками. Наприклад, Perl та Tcl мають потужний механізм для роботи, вбудований безпосередньо в їх синтаксис. Завдяки набору утиліт (включаючи редактор sed та фільтр grep), що входили до складу дистрибутивів Юнікс регулярні вирази стали відомими та поширеними.

Регулярні вирази базуються на теорії автоматів та теорії формальних мов. Ці розділи теоретичної кібернетики займаються дослідженням моделей обчислення (автомати) та способами описання та класифікації формальних мов.

Регулярний вираз (часто називається шаблон) є послідовністю, що описує множину рядків. Ці послідовності використовують для точного описання множини без перелічення всіх її елементів. Наприклад, множина, що складається із слів «грати» та «ґрати» може бути описана регулярним виразом «[гґ]рати». В більшості формалізмів, якщо існує регулярний вираз, що описує задану множину, тоді існує нескінченна кількість варіантів, які описують цю множину.

В якості виділення термінів та їх визначень з статей ресурсу запропоновано регулярний вираз:

\.*\s*[А-Я][а-яА-Я́ \s\-]+\s*(\([^\)]*\))?\s+([—])\s[^\.\n]*[\.\n]\s*

Він виділяє речення, в який йде визначення терміну з явним показом того, що це термін, використовуючи знак «тире».

Але нам також потрібно виділити сам термін з цього речення. Для вирішення цієї проблеми було запропоновано ще один регулярний вираз, який виконує пошук терміна в реченні з його визначенням:

[А-Я][а-яА-Я́ \s\-]+

## 3.3 СТЕМІНГ ТЕРМІНІВ

Слова в українській мові мають відмінкові, родові та множинні форма написання слова. Таким чином один термін в різних формах пишеться по різному. При пошуку терміна в текстах важливо впевнитись, що якесь слово є саме відмінковою, родовою або множинною формою якогось терміна. Для вирішення цієї задачі існує стемінг слів.

Стемінг (англ. stemming) - це процес скорочення слова до основи шляхом відкидання допоміжних частин, таких як закінчення чи суфікс. Результати

стемінгу іноді дуже схожі на визначення кореня слова, але його алгоритми базуються на інших принципах. Тому слово після обробки алгоритмом стемінгу (стематизації) може відрізнятися від морфологічного кореня слова. Стемінг застосовується в лінгвістичній морфології та в інформаційному пошуку. Багато пошукових систем використовують стемінг для об'єднання слів у яких збігаються форми після стематизації, вони вважають такі слова синонімами. Цей процес називають злиттям.

Комп'ютерна програма, що реалізує алгоритм стемінгу іноді має назву стемер.

Вперше алгоритм стемінгу був опублікований en:Julie Beth Lovins in 1968.[1] На свій час це була передова робота яка мала великий вплив на подальші дослідження у цьому напрямку.

Пізніше алгоритм стемінгу був написаний Мартіном Портером та опублікований в липні 1980 у журналі Program. Цей алгоритм набув широкого поширення та став де-факто стандартним алгоритмом стемінгу для англійської мови. Доктор Портер отримав en:Tony Kent Strix award у 2000 році за свою роботу над стемінгом та внесок у галузь пошуку інформації.

Існує досить багато реалізацій Портерівського алгоритму, що вільно поширюються у програмному забезпеченні, але деякі з них мають певні вади. Як результат, не всі алгоритми стемінгу видають результат, який від них очікують. Щоб зменшити подібні помилки Мартін Потер створив офіційну вільну реалізацію алгоритму у 2000 році. А наступні кілька роки присвятив побудові Snowball, спеціального середовища для написання алгоритмів стемінгу, яке призначене для вдосконалення стемінгу англійської мови та написанню алгоритмів стемінгу ще для кількох мов.

Для вирішення задачі стемінгу термінів в роботі був використаний саме алгоритм «Snowball». Він якнайкраще підходить для цієї задачі.

## 3.4 СЕРВЕРНА ЧАСТИНА ЗАСТОСУНКУ

В якості серверної частини застосунку для вирішення поставленої задачі була використана технологія ASP.NET WebAPI для створення REST сервісів для клієнтської частини. В якості бази даних був вибраний продукт компанії Microsoft – MSSQL. Також була використана технологія Entity Framework. Це об’єктно орієнтована модель для роботи з БД.

REST (скор. англ. Representational State Transfer, «передача репрезентативного стану») — підхід до архітектури мережевих протоколів, які забезпечують доступ до інформаційних ресурсів. Був описаний і популяризований у 2000 році Роєм Філдінгом (Roy Fielding), одним із творців протоколу HTTP. Найвідомішою системою, побудованою переважно за архітектурою REST, є сучасна Всесвітня павутина.

Дані повинні передаватися у вигляді невеликої кількості стандартних форматів (наприклад HTML, XML, JSON). Мережевий протокол (як і HTTP) повинен підтримувати кешування, не повинен залежати від мережевого прошарку, не повинен зберігати інформацію про стан між парами «запит-відповідь». Стверджується, що такий підхід забезпечує масштабування системи і дозволяє їй еволюціонувати з новими вимогами.

Антиподом REST є підхід, заснований на виклику віддалених процедур (Remote Procedure Call, RPC). Підхід RPC дозволяє використовувати невелику кількість мережевих ресурсів з великою кількістю методів і складним протоколом. При підході REST кількість методів і складність протоколу суворо обмежені, що призводить до того, що кількість окремих ресурсів має бути великою.

Web API представляє інший спосіб побудови програми ASP.NET дещо відмінний від ASP.NET MVC. Web API представляє собою веб-службу, яка може взаємодіяти з різними додатками. При цьому додаток може бути веб-

додатком ASP.NET, або може бути мобільним або звичайним десктопних додатком.

Також треба відзначити, що платформа Web API 2 не є частиною фреймворка ASP.NET MVC і може бути задіяна як в зв'язці з MVC, так і в поєднанні з Web Forms. Тому в Web API є своя система версій. Так, перша версія з'явилася з .net 4.5. А разом з .NET 4.5.1 і MVC 5 вийшла Web API 2.0.

Контролери в Web API приймають запит у вигляді об'єкта HttpRequestMessage, обробляють його за допомогою одного з методів і посилають у відповідь результат обробки у вигляді об'єкта HttpResponseMessage.

Microsoft SQL Server — комерційна система керування базами даних, що розповсюджується корпорацією Microsoft. Мова, що використовується для запитів — Transact-SQL, створена спільно Microsoft та Sybase. Transact-SQL є реалізацією стандарту ANSI/ISO щодо структурованої мови запитів (SQL) із розширеннями. Використовується як для невеликих і середніх за розміром баз даних, так і для великих баз даних масштабу підприємства. Багато років вдало конкурує з іншими системами керування базами даних.

Базовий код MS SQL Server (до версії 7.0) ґрунтувався на коді Sybase SQL Server. Це дозволило Microsoft вийти на ринок баз даних для підприємств, де конкурували Oracle, IBM, і, пізніше, сама Sybase. Microsoft, Sybase і Ashton-Tate спочатку об'єдналися для створення і випуску на ринок першої версії програми, що отримала назву SQL Server 1.0 для OS/2 (близько 1989 року), яка фактично була еквівалентом Sybase SQL Server 3.0 для Unix, VMS та ін. Microsoft SQL Server 4.2 був випущений у 1992 році та входив до складу операційної системи Microsoft OS/2 версії 1.3. Офіційний реліз Microsoft SQL Server версії 4.21 для ОС Windows NT відбувся одночасно з релізом самої Windows NT (версії 3.1). Microsoft SQL Server 6.0 був першою версією SQL

Server, створеною виключно для архітектури NT і без участі в процесі розробки Sybase.

До того часу, як вийшла на ринок ОС Windows NT, Sybase і Microsoft розійшлися та створювали вже власні моделі цього програмного продукту. Microsoft намагалася отримати виняткові права на всі версії SQL Server для Windows. Пізніше Sybase змінила назву свого продукту на Adaptive Server Enterprise щоб уникнути плутанини з Microsoft SQL Server. До 1994 року Microsoft отримала від Sybase три повідомлення про авторські права як натяк на походження Microsoft SQL Server.

Після розділення компанії зробили декілька самостійних релізів програм. SQL Server 7.0 був першим сервером баз даних зі справжнім графічним інтерфейсом адміністрування. Для усунення претензій з боку Sybase у порушенні авторських прав, весь успадкований код в сьомій версії був переписаний. Це забезпечило також й успіх SQL Server 2000, який був першою редакцією, орієнтованою на архітектуру IA-64.

Протягом подальших шести років корпорація Microsoft працювала над вдосконаленням вже існуючої версії SQL Server 2000 доки не збудувала зручнішу систему Microsoft SQL Server 2005. Були вдосконалені продуктивність, клієнтські інструменти інтегрованого середовища розробки, а також у декількох додаткових системах, що встановлюються разом із SQL Server 2005. Змінено: інструментарій процесів керування сховищами даних (SQL Server Integration Services або SSIS), сервер звітів, сервер OLAP та інтелекутального аналізу даних (Analysis Services), а також декілька технологій повідомлень, особливо Service Broker та Notification Services.

ADO.NET Entity Framework (EF) - об'єктно-орієнтована технологія доступу до даних. Вона є object-relational mapping (ORM) рішенням для .NET Framework від Microsoft. Надає можливість взаємодії з об'єктами як за допомогою LINQ у вигляді LINQ to Entities, так і з використанням Entity SQL.

Для полегшення побудови web-рішень використовується як ADO.NET Data Services (Astoria), так і зв'язка з Windows Communication Foundation і Windows Presentation Foundation, що дозволяє будувати багаторівневі додатки, реалізуючи один з шаблонів проектування MVC, MVP або MVVM.

# ВИСНОВКИ ТА ПРОПОЗИЦІЇ

В рамках даної роботи було вирішено задачу ітеративної побудови термінології в колекціях наукових текстів. Результати роботи представлено у вигляді розробленого веб застосунку з можливостями побудови тезаурусів з вхідних текстів формату txt. Серед типів зв'язків між термінами для пошуку було віддано перевагу зв'язкам "загальне-часткове", що визначалися за допомогою лексикографічного аналізу речень текстів на предмет вмісту гіпонімічних зв'язків між термінами. В основу описаного модулю побудови тезаурусів покладено розглянутий в даній роботі на основі даних попередніх досліджень метод пошуку важливих термінів і зв'язків у тексті. Перший етап даного методу, що пов'язаний з пошуком важливих термінів в колекціях документів, було вирішено за допомогою запропонованого в даній роботі методу зважування, сортування і фільтрації термінів документів за допомогою метрики документарної частоти еталонної колекції. Другий етап методу пов'язаний із застосуванням лексикографічних шаблонів для пошуку гіпонімічних зв'язків у вихідних текстах. Під час пошуку вдалої реалізації було використано передове програмне забезпечення для вирішення утилітарних задач лематизації термінів і тегування слів речень за частинами мови, а також адаптовано до україномовних правил слововжитку лексикографічні шаблони, запропоновані в дослідженні Хеарста [30]. В роботі було розроблено розширюваний програмний пакет з функціональністю управління застосуванням лексикографічних шаблонів. Для збільшення методів виділення термінів потрібно лише описати програмний клас, який реалізує пошук термінів. В веб застосунку реалізовано метод на основі регулярних виразів для знаходження термінів описаних найпопулярнішим методом з використанням «тире».

Тестування реалізації запропонованого методу на тематичних колекціях наукових текстів продемонструвало ефективність першого етапу алгоритму, а також достатню точність другого етапу в межах розроблених шаблонів. Обмеження лексикографічного методу пошуку гіпонімії на дають змоги досягти повноти пошуку зв'язків у тексті через однозначність вживаних в шаблонах контекстів термінологічних зв'язків і низьку статистичну частоту їх появи в тексті, і дану проблему запропоновано обходити в методі шляхом збільшення кількості шаблонів, розширенням синонімічних рядів визначальних шаблон слів, що вимагає залучення експертів з лексикографії, а також за допомогою покращення методу тегування за частинами мови з використанням стохастичних методів усунення неоднозначності в визначенні частин мови окремих слів.

Отриманий в результаті роботи програмний модуль продемонстрував свою прикладну застосовність на тестових колекціях даних, і може бути використаний в подальшому як складова пошукової системи наукових матеріалів.

## СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ


1. Архів журналу “Біотехнологія”// Веб. 14.04.2014 – Доступний з: http://www.biotechnology.kiev.ua/
2. Архів журналу “Наукові записки НаУКМА”. Веб. 12.04.2014 Доступний з: <http://nz.ukma.edu.ua/index.php?option=com_content&task=section&id=10&Itemid=47> і http://www.ekmair.ukma.kiev.ua/
3. Коваль А. П. Науковий стиль сучасної української мови: Структура наукового тексту //К.: Вища шк. – 1970.
4. Кочерган М. П. Вступ до мовознавства //К.: Академія. – 2000. — 368 с.
5. Кочерган М. П. Лексико-семантична система //Українська мова: Енциклопедія. – 2000. – С. 305-306.
6. Лексикографія. Вікіпедія. // Веб. 08.04.2014 – Доступно з: <http://uk.wikipedia.org/wiki/Лексикографія>.
7. Панько Т. І., Кочан І. М., Мацюк Г. П. Українське термінознавство //Львів: Світ. – 1994. – Т. 216.
8. Публікації інституту соціології НАНУ// Веб. 15.04.2014 – Доступний з: http://i-soc.com.ua/institute/el_library.php
9. Словники: мистецтво та ремесло лексикографії Сідні І. Лендау// Київ: К.І.С. – 2012. – 480с.
10. Стандарт ISO 25694. // Веб. 04.03.2014 – Доступно з: http://www.niso.org/schemas/iso25964/iso25964-1_v1.4.xsd
11. Типи зв’язків у тезаурусі. // Веб. 10.05.2014 – Доступно з: http://publish.uwo.ca/~craven/677/thesaur/main06.htm
12. Українська мова: енциклопедія. – Вид-во" Українська енцклопедія" ім. МП Бажана, 2004.

13."Agile and Scalable." MongoDB. Web. 05 April 2014. <http://www.mongodb.org/>.
14."Apache Lucene Core." Apache Lucene -. Web. 08 April 2014. <http://lucene.apache.org/core/>.
15."Apache PDFBox - A Java PDF Library."Apache PDFBox. Web. 08 June 2014. <http://pdfbox.apache.org/>.
16. Adamson G. W., Boreham J. The use of an association measure based on character structure to identify semantically related pairs of words and document titles //Information Storage and Retrieval. – 1974. – Т. 10. – №. 7. – С. 253-260.
17. Anderson J. D., Pérez-Carballo J. The nature of indexing: how humans and machines analyze messages and texts for retrieval. Part I: Research, and the nature of human indexing //Information Processing & Management. – 2001. – Т. 37. – №. 2. – С. 231-254.
18. Alshawi H. Processing dictionary definitions with phrasal pattern hierarchies //Computational Linguistics. – 1987. – Т. 13. – №. 3-4. – С. 195-202.
19."Bitbucket." Hlavki / JLemmaGen —. Web. 08 April 2014. <https://bitbucket.org/hlavki/jlemmagen>.
20."Building a RESTful Web Service."Getting Started ·. Web. 08 April 2014. <https://spring.io/guides/gs/rest-service/>.
21."Custom Search." — Google Developers. Web. 08 April 2014. <https://developers.google.com/custom-search/json-api/v1/overview>.
22. Chen H. et al. A concept space approach to addressing the vocabulary problem in scientific information retrieval: an experiment on the worm community system. – 1997.
23. Chen H. et al. Automatic thesaurus generation for an electronic community system. – 1995.

24. Chen H. et al. Generating, integrating, and activating thesauri for conceptbased document retrieval //IEEE expert. – 1993. – Т. 8. – №. 2. – С. 25-34.
25. Chen H., Lynch K. J. Automatic construction of networks of concepts characterizing document databases //Systems, Man and Cybernetics, IEEE Transactions on. – 1992. – Т. 22. – №. 5. – С. 885-902.
26. DOSTÁL J. Termín a jeho definice ve výkladových, terminologických a naučných slovnících a encyklopediích //Modernizace výuky v technicky orientovaných předmětech a oborech. – С. 57.
27. "Designing and Implementing RESTful Web Services with Spring." Tutorial ·. Web. 12 May 2014. <https://spring.io/guides/tutorials/rest/2/>.
28. Grefenstette G. Automatic thesaurus generation from raw text using knowledge-poor techniques. – Rank Xerox Research Centre, 1993.
29. "HATEOAS". Wikipedia. Wikimedia Foundation. Web. 03 April 2014. <http://en.wikipedia.org/wiki/HATEOAS>.
30. Hearst M. A. Automatic acquisition of hyponyms from large text corpora //Proceedings of the 14th conference on Computational linguistics-Volume 2. – Association for Computational Linguistics, 1992. – С. 539-545.
31. "ISO 25964 – the International Standard for Thesauri and Interoperability with Other Vocabularies." ISO 25964 Thesaurus Schemas. Web. 08 April 2014. <http://www.niso.org/schemas/iso25964/>.
32. JSON-LD 1.0. Web. 08 June 2014. <http://www.w3.org/TR/json-ld/>.
33. Kryslová M. Stanovlennja ukrajinskoji linhvistyčnoji terminolohiji. – 2009.
34. "LanguageTool Wiki." Java API -. Web. 08 April 2014. <http://wiki.languagetool.org/java-api>.

35. Lassi M. Automatic thesaurus construction //University Collage of Boras, Sweden. – 2002.
36. Manola F., Miller E., McBride B. RDF primer, 2004 //URL <http://www.w3.org/TR/rdf-primer>. – 2010.
37. Miller U. Thesaurus construction: problems and their roots //Information Processing & Management. – 1997. – Т. 33. – №. 4. – С. 481-493.
38. "RDF 1.1 Concepts and Abstract Syntax." RDF 1.1 Concepts and Abstract Syntax. Web. 12 May 2014. <http://www.w3.org/TR/2014/PR-rdf11concepts-20140109/>.
39. Schütze H., Pedersen J. O. A cooccurrence-based thesaurus and two applications to information retrieval //Information Processing & Management. – 1997. – Т. 33. – №. 3. – С. 307-318.
40. "Spring MongoDB Tutorial." Spring MongoDB Tutorial. Web. 08 April 2014. <http://bits-and-kites.blogspot.com/2014/01/spring-mongodbtutorial.html>.
41. "Spring Data MongoDB." Spring Data MongoDB. Web. 08 April 2014. <http://projects.spring.io/spring-data-mongodb/>.
42. Strzalkowski T. Natural language information retrieval //Information Processing & Management. – 1995. – Т. 31. – №. 3. – С. 397-417.
43. "WordNet." About - . Web. 08 June 2014. <http://wordnet.princeton.edu/>.

# ДОДАТКИ

***Додаток А***

**(довідниковий)**

**Інтерфейс користувача**


Зліва знаходиться список всіх документів. Можливий пошук по ним. Можливе додавання нового документу. Після додавання система автоматично знайде в ньому терміни та зв’яже їх з текстами де цей термін використовується.

*Додаток Б*

**(обов'язковий)**

# Код програми

## Моделі даних:

```
using System.Collections.Generic;

namespace TextStorage.Models
{

    public class Term
    {
        public int Id { get; set; }
        public string Name { get; set; }
        public string StemmedName { get; set; }

        public virtual ICollection<TermDescription> Descriptions { get; set; }

        public Term()
        {
            Descriptions = new List<TermDescription>();
        }
    }
}
```

```
namespace TextStorage.Models
{
    public class TermDescription
    {
        public int Id { get; set; }
        public string Description { get; set; }
        public int TermId { get; set; }
        public virtual Term Term { get; set; }
        public int TextId { get; set; }
        public virtual Text Text { get; set; }
    }
}
```

```
using System.Collections.Generic;

namespace TextStorage.Models
{
    public class Text
    {
        public int Id { get; set; }
        public string Name { get; set; }
        public virtual string TextContent { get; set; }
        public virtual string TextContentOriginal { get; set; }
    }
}
```

```
using System.Data.Entity;

namespace TextStorage.Models
{
    public class TextStorageContext : DbContext
    {
        public TextStorageContext() : base("TextStorage")
```

```
        {
        }

        public DbSet<Term> Terms { get; set; }
        public DbSet<Text> Texts { get; set; }
        public DbSet<TermDescription> Descriptions { get; set; }
    }
}
```

## Контролери WebAPI:

```
using System.Data;
using System.Linq;
using System.Web.Http;
using System.Web.Http.OData;
using TextStorage.Models;

namespace TextStorage.Controllers
{
    public class TermsController : ODataController
    {
        private TextStorageContext db = new TextStorageContext();

        // GET: odata/Terms
        [EnableQuery]
        public IQueryable<Term> GetTerms()
        {
            return db.Terms;
        }

        // GET: odata/Terms(5)
        [EnableQuery]
        public SingleResult<Term> GetTerm([FromODataUri] int key)
        {
            return SingleResult.Create(db.Terms.Where(term => term.Id == key));
        }

        // GET: odata/Terms(5)/Descriptions
        [EnableQuery]
        public  IQueryable<TermDescription>  GetDescriptions([FromODataUri]  int  key)
        {
            return db.Terms.Where(m => m.Id == key).SelectMany(m => m.Descriptions);
        }

        protected override void Dispose(bool disposing)
        {
            if (disposing)
            {
                db.Dispose();
            }
            base.Dispose(disposing);
        }

        private bool TermExists(int key)
        {
            return db.Terms.Count(e => e.Id == key) > 0;
        }
    }
}

using System.Data;
```

```
using System.IO;
using System.Linq;
using System.Net;
using System.Threading.Tasks;
using System.Web;
using System.Web.Http;
using System.Web.Http.OData;
using TextStorage.Models;
using TextStorage.TermSearcher;

namespace TextStorage.Controllers
{
	public class TextsController : ODataController
	{
		private TextStorageContext db = new TextStorageContext();

		// GET: odata/Texts
		[EnableQuery]
		public IQueryable<Text> GetTexts() {
			return db.Texts;
		}

		// GET: odata/Texts(5)
		[EnableQuery]
		public SingleResult<Text> GetText([FromODataUri] int key) {
			return SingleResult.Create(db.Texts.Where(text => text.Id == key));
		}

		// POST: odata/Texts
		public async Task<IHttpActionResult> Post() {
			var file = HttpContext.Current.Request.Files["UploadedText"];
			var fileName = HttpContext.Current.Request["UploadedTextName"];

			var streamReader = new StreamReader(file.InputStream);
			var textContent = streamReader.ReadToEnd();

			var newText = new Text() {
				Name = fileName,
				TextContent = textContent,
				TextContentOriginal = textContent
			};

			newText = db.Texts.Add(newText);
			await db.SaveChangesAsync();

			var newTerms = await RegexpSearcher.SaveTerms(newText, db);
			if (newTerms > 0) {
				TermsUtils.ApplyTerms(db);
				await db.SaveChangesAsync();
			}

			return Ok();
		}

		// DELETE: odata/Texts(5)
		public async Task<IHttpActionResult> Delete([FromODataUri] int key) {
			Text text = await db.Texts.FindAsync(key);
			if (text == null) {
				return NotFound();
			}

			db.Texts.Remove(text);
			await db.SaveChangesAsync();
```

```
                return StatusCode(HttpStatusCode.NoContent);
            }

            protected override void Dispose(bool disposing) {
                if (disposing) {
                    db.Dispose();
                }
                base.Dispose(disposing);
            }

            private bool TextExists(int key) {
                return db.Texts.Count(e => e.Id == key) > 0;
            }
        }
}

using System.Data;
using System.Linq;
using System.Web.Http;
using System.Web.Http.OData;
using TextStorage.Models;

namespace TextStorage.Controllers
{
  public class TermDescriptionsController : ODataController
  {
    private TextStorageContext db = new TextStorageContext();

    // GET: odata/TermDescriptions
    [EnableQuery]
    public IQueryable<TermDescription> GetTermDescriptions()
    {
      return db.Descriptions;
    }

    // GET: odata/TermDescriptions(5)
    [EnableQuery]
    public SingleResult<TermDescription> GetTermDescription([FromODataUri] int key)
    {
      return SingleResult.Create(db.Descriptions.Where(termDescription => termDescription.Id == key));
    }

    // GET: odata/TermDescriptions(5)/Term
    [EnableQuery]
    public SingleResult<Term> GetTerm([FromODataUri] int key)
    {
      return SingleResult.Create(db.Descriptions.Where(m => m.Id == key).Select(m => m.Term));
    }

    // GET: odata/TermDescriptions(5)/Text
    [EnableQuery]
    public SingleResult<Text> GetText([FromODataUri] int key)
    {
      return SingleResult.Create(db.Descriptions.Where(m => m.Id == key).Select(m => m.Text));
    }

    protected override void Dispose(bool disposing)
    {
      if (disposing)
      {
        db.Dispose();
```

```
        }
        base.Dispose(disposing);
    }

    private bool TermDescriptionExists(int key)
    {
        return db.Descriptions.Count(e => e.Id == key) > 0;
    }
  }
}
```

## Стеммер:

```
using Iveonik.Stemmers;
using System;
using System.Linq;

namespace TextStorage.TermSearcher
{
  public class Stemmer
  {
    public static string Stem(string s)
    {
      var stemmer = new RussianStemmer();
      var separators = new char[] { ' ', '\t', '\r' };
      var result = s.Replace("́", string.Empty)
        .Split(separators, StringSplitOptions.RemoveEmptyEntries)
        .Select(t => stemmer.Stem(t));
      return string.Join(" ", result);
    }
  }
}
```

## Клас пошуку термінів за регулярним виразом:

```
using TextStorage.Models;
using System.Linq;
using System.Text.RegularExpressions;
using System.Threading.Tasks;

namespace TextStorage.TermSearcher
{
  public class RegexpSearcher
  {
                const string REG_EXP_SENTENSE = @"\.*\s*[А-Я][а-яА-Я́ \s\-]+\s*(\([^\)]*\))?\s+([—
])\s[^\.\n]*[\.\n]\s*";
    const string REG_EXP_TERM = @"[А-Я][а-яА-Я́ \s\-]+";
    //тут є "Знак ударения" з кодом 769

    public static async Task<int> SaveTerms(Text textEntity, TextStorageContext dbContext)
    {
      var regex = new Regex(REG_EXP_SENTENSE, RegexOptions.CultureInvariant);
      int resultCount = 0;
      foreach (Match sentMatch in regex.Matches(textEntity.TextContentOriginal))
      {
        var value = sentMatch.Value.Trim('.', ' ', '\n', '\t', '\r');
        var termMatch = Regex.Match(value, REG_EXP_TERM, RegexOptions.CultureInvariant);
        var term = termMatch.Value.Trim();
        var termStemmed = Stemmer.Stem(term);
        if (term.Length <= 3) continue; //назва терміну закоротка
        var termEntity = dbContext.Terms.Where(e => e.StemmedName == termStemmed).FirstOrDefault();
        if (termEntity == null)
```

```
                termEntity = dbContext.Terms.Add(new Term()
                {
                    Name = term,
                    StemmedName = termStemmed
                });
                dbContext.Descriptions.Add(new   TermDescription()
                {
                    Description = value,
                    TermId = termEntity.Id,
                    TextId = textEntity.Id
                });
                await dbContext.SaveChangesAsync();
                resultCount++;
            }
            return resultCount;
        }
    }
}
```

## Алгоритм пошуку та виділення термінів в текстах:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using TextStorage.Models;

namespace TextStorage.TermSearcher
{
    public class TermsUtils
    {
        public static void ApplyTerms(TextStorageContext dbContext)
        {
            var terms = dbContext.Terms.OrderByDescending(t => t.StemmedName.Length).ToList();
            if (terms.Count == 0)
                return;
            var maxWordsInTerm = terms.Max(t => t.StemmedName.Split(new char[] { ' ' },
                StringSplitOptions.RemoveEmptyEntries).Length);
            string[] words = new string[maxWordsInTerm];
            string[] temp = new string[maxWordsInTerm];
            string textOriginal = "";
            int position = 0;
            int termStartPosition = 0;
            StringBuilder sb;
            bool hasWordFlag = false;
            int termId = -1;
            foreach (var text in dbContext.Texts)
            {
                sb = new StringBuilder();
                textOriginal = text.TextContentOriginal;
                for (position = 0; position < textOriginal.Length;)
                {
                    while (position < textOriginal.Length && IsSeparatorAll(textOriginal[position]))
                    {
                        sb.Append(textOriginal[position]);
                        position++;
                    }
                    words = new string[maxWordsInTerm];
                    temp = new string[maxWordsInTerm];
                    termStartPosition = position;
                    for (int i = 0; i < maxWordsInTerm && position < textOriginal.Length; i++)
                    {
                        while (position < textOriginal.Length && !IsSeparatorAll(textOriginal[position]))
```

```
                {
                    for (int j = i; j < maxWordsInTerm; j++)
                    {
                        hasWordFlag = true;
                        words[j] += textOriginal[position];
                        temp[j] += textOriginal[position];
                    }
                    position++;
                }
                while (position < textOriginal.Length && hasWordFlag && IsSeparator(textOriginal[position]))
                {
                    for (int j = i; j < maxWordsInTerm; j++)
                    {
                        temp[j] += textOriginal[position];
                    }
                    position++;
                }
                for (int j = i; j < maxWordsInTerm && hasWordFlag; j++)
                {
                    words[j] += " ";
                }
                hasWordFlag = false;
            }
            termId = -1;
            for (int i = maxWordsInTerm - 1; i >= 0; i--)
            {
                if (!string.IsNullOrWhiteSpace(words[i]))
                {
                    termId = isTerm(words[i], terms);
                    if (termId != -1)
                    {
                        sb.AppendFormat("<span id='term-{0}'>{1}</span>", termId, temp[i]);
                        position = termStartPosition + temp[i].Length;
                        break;
                    }
                }
            }
            if (termId == -1)
            {
                sb.Append(temp[maxWordsInTerm  -  1]);
            }
        }
        text.TextContent = sb.ToString();
    }
}
private static bool IsSeparatorAll(char c)
{
    var separators = new char[]
    {
        ' ', '\n', '\t', '\r', '(', ')', '[', ']',
        '—', '«', '»', '\'', '"', '.', ',', ';'
    };
    return separators.Contains(c);
}
private static bool IsSeparator(char c)
{
    var separators = new char[] { ' ', '\r', '\t' };
    return separators.Contains(c);
}
private static int isTerm(string s, List<Term> terms)
{
    var stemmedS = Stemmer.Stem(s);
    var term = terms.Where(t => t.StemmedName == stemmedS);
```

```
        if (term.Count() > 0) return term.First().Id;
        else return -1;
      }
    }
}
```

## Клієнтська частина застосунку:

```
<!DOCTYPE html>
<html>
<head>
        <title>Text Storage</title>
        <link  href="https://fonts.googleapis.com/css?family=Roboto"  rel="stylesheet"  type="text/css">
        <script  src="scripts/libs/jquery-1.12.4.min.js"></script>
        <script src="scripts/script.js"></script>
        <link rel="stylesheet" href="styles/main.css" />
</head>
<body>
        <div class="header div-group">
                <p class="textName">
                        Text Storage
                </p>
        </div>
        <div class="left div-group">
                <div class="newFileBlock">
                        <input type="file" id="textUpload" />
                        <label>
                                Text name:
         <input  name="newTextName"  id="newTextName"  />
                                <button name="loadNewText" id="loadNewTextButton">Upload</button>
                        </label>
                </div>
                <div class="searchBlock">
                        <label>
                                Text name:
         <input name="textName" id="textName" />
                                <button  name="searchTextNutton"  id="searchTextButton">Search</button>
                        </label>
                </div>
                 <div class="searchResultsBlock">
                        <ul id="searchResultsList" style="list-style-type: none"></ul>
                </div>
        </div>
        <div class="right div-group">
                <p class="textContent" id="textContent">
                </p>
        </div>
        <div id="termDescriptionsContainer">
                <div id="termDescriptionsInnerContainer">
                        <div id="hideTermDescriptionsContainer">X</div>
                        <div id="termDescriptionsList">
                        </div>
                </div>
        </div>
</body>
</html>
```

```
$(document).ready(function () {
   function clearTextsList() {
      $("#searchResultsList").empty();
   }
```

```
function loadSingleText(id, callback) {
    var url = "/odata/Texts(" + id + ")";
    $.getJSON(url, function (text) {
        callback(text);
    });
}
function  showTermDescriptionsList(termId)  {
    $("#termDescriptionsContainer").show();
}
$("#hideTermDescriptionsContainer").on("click",  function  ()  {
    $("#termDescriptionsContainer").hide();
});
function loadTextsList(name) {
    clearTextsList();
    var url = "/odata/Texts";
    var parameters = {
        "$select": "Id,Name"
    };
    if (name) {
        parameters["$filter"] = "substringof('" + name + "',Name)";
    }
    $.getJSON(url, parameters, function (texts) {
        var list = $("#searchResultsList");
        texts.value.forEach(function (text) {
            list.append("<li id='text-" + text.Id + "'>" + text.Name);
        });
        $("#searchResultsList > li").on("click", function () {
            loadSingleText(this.id.substring(5), function (text) {
                $("#textContent").html(text.TextContent);
                $(".textContent > span").on("click", function () {
                    showTermDescriptionsList(this.id.substring(5));
                });
            });
        });
    });
}

$("#loadNewTextButton").on("click", function () {
    var textName = $("#newTextName").val();
    if (!textName) {
        alert("Input text name");
        return;
    }
    var data = new FormData();
    data.append("UploadedTextName", textName);
    var files = $("#textUpload").get(0).files;
    if (files.length > 0) {
        data.append("UploadedText", files[0]);
    } else {
        alert("File is not selected");
        return;
    }

    var ajaxRequest = $.ajax({
        type: "POST",
        url: "/odata/Texts",
        contentType: false,
        processData:  false,
        data: data
    });

    ajaxRequest.done(function (xhr, textStatus) {
        alert(textStatus);
```

```
            $("#newTextName").val("");
            $("#textUpload").val("");
            var name = $("#textName").val();
            loadTextsList(name);
        });
    });
    $("#searchTextButton").on("click", function () {
        var name = $("#textName").val();
        loadTextsList(name);
    });

    loadTextsList();
});
```