Міністерство освіти і науки України

Національний університет «Києво-Могилянська академія»

Факультет інформатики

Кафедра інформатики

**Магістерська робота**

освітньо-кваліфікаційний рівень - магістр

на тему:

**«Розробка методу ітеративної побудови термінологій в колекціях наукових текстів»**

Виконав: студент 2-го року навчання

спеціальності

**8.05010101**«Інформаційні управляючі

системи та технології»

Решетньов Ігор Володимирович

Керівник: Глибовець А. М.,

кандидат технічних наук

Магістерська робота захищена

з оцінкою «\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_»

«\_\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_2014 р.

Київ 2014

Зміст

ВСТУП 4

РОЗДІЛ I. ОГЛЯД ІСНУЮЧИХ ПІДХОДІВ 6

1.1 Роль тезаурусу в інформаційному пошуку 6

1.2 Тезаурус як форма подання зв’язаної термінології 7

1.3 RDF як формат публікації тезаурусів 8

1.4 Огляд стандарту JSON-LD як конкретної специфікації RDF. 9

1.5 Дослідження існуючих автоматизованих методів побудови тезаурусів 10

1.5.1 Статистичні методи 10

1.5.1.1 Ранжування термінів за метрикою TFIDF 11

1.5.1.2 Метод спільного вжитку 12

1.5.1.3 Метод концептуального простору 12

1.5.2 Лексикографічні методи 14

1.5.2.1 Огляд лексикографічних відомостей щодо термінології 14

1.5.2.1 Метод генерації тезаурусу з використанням техніки браку знань 19

1.5.2.2 Гіпонімія як основа для пошуку ієрархічних зв’язків 21

Метод збирання гіпонімів з великих текстових колекцій 21

РОЗДІЛ II. РОЗРОБКА МЕТОДУ 26

2.1 Структурна схема алгоритму 26

2.1.1 Індексування і фільтрація важливих термінів 26

2.1.2 Зважування термінів за метрикою документарної частоти еталонної колекції 29

2.1.3 Метод пошуку зв’язків між термінами 30

2.1.4 Розширення тезауруса термінологічними словосполученнями 31

2.1.5 Методи пошуку збіжності за лексикографічним шаблоном 31

2.1.6 Інтерпретація зв’язків за збіжністю тексту з шаблоном 33

2.1.7 Підходи до отримання фразових іменникових словосполучень з тексту 34

2.1.8 Етапи експериментального застосування лексикографічних шаблонів 36

2.2 Математична модель і формалізація методу 36

2.2.1 Формальні позначення 36

2.2.2 Псевдокод розробленого методу 38

2.2.3 Формалізація правил збіжності з лексикографічним шаблоном 38

2.2.4 Перелік розроблених лексикографічних шаблонів 40

РОЗДІЛ III. ПРАКТИЧНА ЧАСТИНА 41

3.1 Збір даних для тестування розробленого методу 41

3.2 Розробка довідкового компоненту документарної частоти 41

3.2.1 Рішення на основі зовнішньої пошукової системи 41

3.2.2 Рішення на основі індексації еталонної колекції документів 42

3.3 Вибір готових рішень для реалізації кроків алгоритму. 43

3.4 Архітектура системи обробки документів для побудови термінології 45

3.5 Специфікація програмного інтерфейсу доступу до системи. 48

3.5.1 Точка доступу до RDF-тезаурусу 48

3.5.2 Точка доступу до керування індексами 49

3.6 Формат серіалізації тезаурусу на базі JSON-LD 49

РОЗДІЛ IV. ОБГОВОРЕННЯ РЕЗУЛЬТАТІВ 51

4.1 Схема тестування та оцінка результатів 51

4.2 Результати тестування методу на тематичних колекціях документів 53

4.3 Пропозиції щодо збільшення повноти і точності методу 54

ВИСНОВКИ 55

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ 57

ДОДАТКИ 61

# 

# 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], наукова україномовна література є найціннішим джерелом розширення нашої лексики, тому дана сфера в першу чергу вимагає дбайливого ставлення до створення відповідної термінології, її опису і уніфікації розроблених терміносистем у відповідних спеціалізованих словниках. Дуже часто в таких умовах темпи створення і актуалізації словників не встигають за поступом прогресу в найновіших дослідженнях, через об’єктивні причини складності досліджуваних сфер і мінливості понять з часом. 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.

Одним із ефективних способів покращення релевантності пошукової видачі таких систем є використання тезаурусу [35], що є довідником характеру і сили зв’язків між термінами. Серед методів побудови тезаурусів автоматизований метод якнайкраще підходить для сфери наукових досліджень через високі темпи оновлюваності інформації і пов’язаної високої собівартості долучення експертів до такої роботи. В рамках низки досліджень, проведених на кафедрі інформатики НаУКМА зі створення пошукової системи наукових документів, розробка компоненту автоматизовоної побудови тезауруса, що базується на методі вилучення термінології з вхідних наукових текстів, постає логічним кроком покращення якості пошукової системи, і є метою даної роботи.

Магістерська робота присвячена аналізу існуючих 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-сервісу для аналізу його ефективності на реальних даних наукової україномовної періодики, з подальшою можливістю застосування розробленого компоненту як складової пошукової системи наукових документів.

Актуальність даної проблеми спричинена також появою у відкритому доступі прикладних рішень для аналізу текстів українською мовою, з можливостями використання таких методів як лематизація і тегування за частинами мови. Під час розробки методу до уваги було прийнято обмеженість документарних колекцій випущених українською мовою, що вимагало врахування можливості ітеративного додавання наукових документів до термінологічної бази з наступним оновленням змісту тезауруса.

Фокус в роботі поставлено на вирішення прикладної задачі побудови термінології з описом зв’язків у форматі RDF з вхідних текстів у широковживаному форматі pdf.

# РОЗДІЛ I. ОГЛЯД ІСНУЮЧИХ ПІДХОДІВ

## Роль тезаурусу в інформаційному пошуку

Зазвичай, інформаційна потреба користувача пошукової системи не відповідає термінам, що зустрічаються в документах, або користувач неправильно розуміє в конкретний час в конкретному місці термінологію області знань, в якій він здійснює пошук. За таких умов, одним із методів покращення пошукової видачі є 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]. Інформаційні системи можуть використовувати тезауруси на етапі індексації документів, для того щоб правильніше класифікувати документи по категоріях, або під час пошуку, розширюючи пошуковий запит користувача пов’язаними термінами.

Тут варто згадати експерименти Фурне і Ландауера [22, c.17], які показали, що більшість користувачів тільки у 20% випадків вибирали однакові терміни для опису тих самих речей, що свідчить про необхідність для пошукових систем враховувати синонімію термінів, що в свою чергу задається саме в тезаурусах. Більш того, порівняльні заміри релевантності звичайної пошукової видачі з розширеною пошуковою видачею, до якої було додано інформацію з тезаурусу, за думкою Чена [22, c.178], показують можливість її покращення. Таким чином, можна вважати обґрунтованим використання тезаурусів в пошукових системах задля покращення їх якості.

Головна проблема складання тезаурусів полягає у тому, що для більшості комерційних баз даних, що поширюють наукову інформацію, вони складаються експертами з областей знань, а також фахівцями зі складання тезаурусів, тобто власноруч. Якщо ж розглядати окремі найновіші області знань, де відповідна термінологія тільки формується і випускається велика кількість нових публікацій, таких як біоінформатика або комп’ютерна інженерія, то в даному випадку термінологічні словники дуже швидко застарівають, і їх треба оновлювати частіше, тобто знову залучати 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. Як наслідок, маючи таку систему оновлювати термінологічні зв’язки стає значно простіше і дешевше. В публікації Моніки Лассі [35] було розглянуто основні методи автоматизованої побудови тезаурусів, такі як метод спільного вжитку, метод концептуального простору і метод баєсівських мереж. Дані методи мають як різну ефективність і часову оцінку складності, так і принципи - статистичний і лексикографічний. Розробці і застосуванню нового методу, що б використовував комбінацію ідей, що зустрічаються в цих підходах, і присвячено дану магістерську тезу.

## 1.2 Тезаурус як форма подання зв’язаної термінології

Тезаурусом називають керований словник, що містить семантичні зв’язки між термінами, і використовується з метою поліпшення процесу пошуку пов’язаних термінів. Означень тезауруса досить багато, починаючи від доволі короткого і загального, наведеного в роботі Шутца і Педерсона [39, c.307]: *“Ми визначаємо тезаурус як просте відношення між словами і іншими близько пов’язаними словами”,* уточнюючи далі що тезаурус має бути достатньо специфічним щоб надавати користувачу синоніми шуканих слів. На противагу, в роботі Міллера [37, c.489] подано більш точне означення тезауруса, що є *“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.”* Цікаво, що автор наголошує на принциповій нерозривності теоретичної моделі тезауруса і практичного використання програмних модулів із такою функціональністю.

Функціонально, зв’язки між термінами в тезаурусах прийнято позначати так: ширший термін (BT), вужчий термін (NT), переважний термін (USE), пов’язаний термін (RT) – для позначення зв’язків, для котрих неможливо точно встановити напрямок зв’язку.

Конвенційний спосіб складання тезаурусів – за допомогою залучення експертів з певних галузей знань – досить коштовний. Одним із способів зменшення вартості складання тезауруса є використання вже існуючих лексикографічних баз даних, таких як WordNet, що має серйозну проблему при застосуванні такого корпусу загальної лексики до доменно-специфічних областей знань [39, c.308]. Інший підхід – автоматизовані методи створення тезаурусів, що можуть підготувати досить точну основу для подальшого уточнення експертами.

## 1.3 RDF як формат публікації тезаурусів

Формат RDF [38] є одним із найпоширеніших способів подання даних і метаданих для технологій семантичного вебу. В основі даного формату, не вдаючись до подробиць абстрактного синтаксису і конкретних специфікацій, що ретельно розроблені у відповідних міжнародних стандартах, лежить ідея подання інформації у вигляді триплетів “суб’єкт”-“предикат”-“об’єкт”. Така загальна і проста на перший погляд модель може вдало задовольнити потреби тезауруса для опису його змісту. Однак поряд із задоволенням потреб на рівні узгодженості моделей постає інша надзвичайно важлива властивість міжнародного формату – широка всесвітня підтримка на рівні реалізації прикладних систем. Як зазначалося раніше, роль тезауруса визначається не лише точністю і об’ємом представлених термінологічних зв’язків, але й практичною застосовністю програмного модулю, простотою доступу і придатністю до машинної обробки. Саме завдяки можливості публікації оброблених за допомогою програмної частини тезауруса даних безпосередньо в мережі Інтернет у міжнародно-прийнятному форматі, нами було запропоновано формат RDF і підлеглу систему веб-сервісів з програмним інтерфейсом в якості кінцевого формату доступу до тезаурусу.

## 1.4 Огляд стандарту JSON-LD як конкретної специфікації RDF.

Серед конкретних специфікацій RDF, формат JSON-LD, поданий у стандарті ISO-25964 [31], на нашу думку найкращім чином відповідає поставленій задачі публікації ресурсів тезауруса у вигляді веб-сервісу. До визначених нами переваг формату належить:

* реалізація моделі *пов’язаних даних*, що є необхідною властивістю нами означеного тезауруса як системи доступу до даних про зв’язки термінів, і певним чином наслідує ідеї концепції RESTful HATEOAS [29] веб-сервісів, що на нашу думку є одним з найбільш прогресивних на сьогодні індустріальних підходів до підтримки машинної обробки веб-ресурсів;
* підтримка елемента *контексту* для опису доменних обмежень і типів полів тіла документа, що дозволяє значно скоротити запис без втрати в точності визначення обмежень;
* формат JSON є більш прийнятним для читання людиною з-поміж інших альтернатив;
* серіалізацію даних у формат JSON-LD за допомогою веб-сервісу досить легко реалізувати завдяки широкій підтримці базового формату JSON в прикладних бібліотеках з відкритим кодом.

До базових концепцій формату відносяться [32]:

1. IRI – інтернаціональні ідентифікатори ресурсів,
2. контекст, що слугує головним чином для задання скорочень до IRI,
3. Ідентифікатори вузлів і типізовані значення.

Запропонованих базових елементів формату має бути достатньо щоб подати дані тезаурусу у мінімально задовільному до стандарту вигляді.

## 1.5 Дослідження існуючих автоматизованих методів побудови тезаурусів

Методи автоматизованого складання тезаурусів можна поділити на два принципових класи: *статистичні*, що інтенсивно використовують частотні і позиційні характеристики термінів в документах в якості основи для різноманітних моделей виявлення зв’язків між термінами, і *лексикографічні,* що використовують відомості зі сфери обробки людської мови для здійснення синтаксичного, морфологічного і інших видів аналізу тексту з метою знаходження семантичних зв’язків на основі отриманої виключно з тексту інформації. При цьому в лексикографічних методах зазвичай набувають широкого вжитку зібрані експертами корпуси мов, що містять правила спільного вжитку слів, словоформи і синонімічні ряди, а також доступні програмні пакети для проведення початкового аналізу вільного тексту, зокрема утиліти тегування за частинами мови, лематизатори, стемери і поготів. В свою чергу, для статистичних методів такою основою є утиліти індексування і ранжування термінів. Подане розбиття методів на два класи за великим рахунком умоглядне і не є взаємовиключним, тому в більшості досліджень зустрічається спільне використання технік з різних підходів, з наданням переваги одному з класів методів.

### 1.5.1 Статистичні методи

Основою для багатьох статистичних методів пошуку залежностей між термінами є створення індексу термінів, що описують зміст документів найкращім чином. В свою чергу, побудова індексу зазвичай потребує ранжування термінів за ступенем важливості, і найбільш вживаними техніками зважування, що беруть свій початок в алгоритмах пошукових систем, є використання частоти терміну (**TF**), зворотньої документарної частоти (**IDF**), а також їх комбінацій. Розглянемо даний підхід детальніше.

#### 1.5.1.1 Ранжування термінів за метрикою 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.

Розглянемо специфічний зміст даних метрик. Відтак, використовуючи виключно частоту терміна в якості міри зважування, оптимальними термінами в індексі стають ті, що мають середню частоту в документі. Найчастіші слова в документі вважаються неприйнятними для відрізнення одного документу від інших в колекції, а найменш вживані слова – вважаються недостатньо впливовими для якісного опису змісту документа.

Натомість, метрика інвертованої документарної частоти бере до уваги появу слів у всій колекції, а не в окремо взятому документі. Таким чином, слова що зустрічаються в усіх або принаймні у багатьох документах колекції вважаються неприйнятними для відрізнення документів, і навпаки, ті, що зустрічаються в декількох документах, вважаються прийнятними для опису цих документів.

Комбінація наведених технік призводить до спільного використання сильних сторін обох методів: великої ваги набувають терміни, що зустрічаються з середньою частотою в документах і водночас з малою частотою в колекції.

Базові формули для обчислення таких метрик подані нижче.

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

*де ni відповідає кількості входжень шуканого слова в документ, а в знаменнику – кількість всіх слів у документі.*

\mathrm{IDF} =  \log \frac{|D|}{|(d_{i}\supset t_{i})|}

*де |D| - кількість документів в колекції, в знаменнику – кількість документів, що містить шуканий термін.*

#### 1.5.1.2 Метод спільного вжитку

Даний метод є одним із підходів в інформаційному пошуку до формування багатослівних термінів, що є статистичною альтернативою лексикографічного методу тегування за частинами мови. Основними елементами для обчислення в методі є частоти входження терміну у певні різні за розміром контекстні рамки, такі як цілий документ, глави документу, параграфи й інші менш дрібні елементи. При цьому чим ближче слова зустрічаються в контексті обраної рамки, тим більшою призначається міра спільного вжитку. В дослідженні Шутца і Педерсона [39] в основу для реалізації даного підходу покладено створення матриці *С*, елементи якої *Cij* показують кількість разів спільного вжитку між словами *i* та *j* в контекстній рамці розміру *k*. З даною реалізацією пов’язані очевидні проблеми ефективності через велику розмірність і тривалий час побудови такої матриці. Стосовно якості знайдених термінологічних зв’язків за даним методом теж виникають сумніви, як у статті Чена [23, c.178], що стверджує про неефективність складеного за даним методом тезаурусу при застосуванні до задач пошуку, і пропонує власний метод, що розглянуто далі.

#### 1.5.1.3 Метод концептуального простору

В цьому підході вводяться поняття *концептуального простору* як мережі термінів і зважених асоціацій між ними, що здатні відтворити концепти і зв’язки між ними у відповідному *інформаційному просторі,* представленому у вигляді колекції документів в базі даних.

Для даного підходу виділяють чотири кроки:

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 стоп-слів, 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 зв’язків між термінами базуються на приципі прямого зазначення зв’язку між словами за допомогою мовних засобів, причому характер зв’язку можна визначити, виходячи з синтаксичної та лексичної будови висловлювань. Перш ніж розглянути основні позиції даних методів, а також їх безпосередній зв’язок з утворенням термінології, необхідно подати декілька означень.

#### 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, c.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.

«Кожен термін є результатом розумової діяльності людини, зокрема діяльності, спрямованої на узагальнення і абстрагування».3

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, c.231].

В українській мові, крім поняття ***термінологія***, досі часто користуються терміном ***термінознавство***. Дефініція його така – «наука, яка займається загальновживаними питаннями терміна, термінології, номенклатури» [7, c.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** лексикографіі 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, c.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, c.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:

|  |
| --- |
| П р и к м е т н и к + і м е н н и к |

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

І м е н н и к + і м е н н и к

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

Тричленні термінологічні словосполучення можуть бути побудовані таким способом:

І м е н н и к + п р и к м е т н и к + і м е н н и к

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

П р и к м е т н и к + п р и к м е т н и к + і м е н н и к

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

П р и к м е т н и к + і м е н н и к + і м е н н и к

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

П р и к м е т н и к + і м е н н и к + п р и к м е т н и к + і м е н н и к

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

П р и к м е т н и к + і м е н н и к + і м е н н и к + і м е н н и к

(*аналітичне творення ступенів порівняння*)

І м е н н и к + п р и й м е н н и к + і м е н н и к + і м е н н и к

(*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, c.173].

Подальші схожі роздуми побачимо в методі Г. Грефенстета.

#### 1.5.2.1 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, c.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.

#### 1.5.2.2 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, c.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.

#### 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:

**NP0, такі як {NP1,NP2 ... , (або | і)} NPn**

де **NPi** – відповідає певному термінологічному іменниковому словосполученню, що потребує, щоб для всіх **NPi, 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 Aлшаві [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”)*

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

Набряки, гіматоми, рани і інші травми ...

*hyponym(“набряк”,”травма”), hyponym(“гіматома”,”травма”), hyponym(“рана”,”травма”)*

1. **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 [43].

Підсумовуючи основні здобутки даного методу, можна вказати на порівняльну дешевизну застосування даного методу для автоматизованого збору семантичних лексичних зв’язків в вільнотекстових 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 II. РОЗРОБКА МЕТОДУ

В даному розділі подано етапи розробки ітеративного методу побудови термінології за допомогою комбінації лексикографічних і статистичних методів.

## 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:

а) по-перше, видобування зі всіх слів, що зустрічаються в текстах документів, таких, що відповідають термінам в області знань відповідних документів;

б) 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. Грунтується 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, або повну збірку журналу “Записки НаУКМА” за декілька років.
* 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 підвантаження 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 віднайдення 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 оператори 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 = (NP0(MR<A,N>), EW(“*–*”,”-”), NP1(MR<N,N>))

Даний шаблон має задовільнити наступна фраза:

*“Соціологічне дослідження — система процедур для отримання наукових знань про*[*соціальні явища і процеси*](http://uk.wikipedia.org/wiki/%D0%A1%D0%BE%D1%86%D1%96%D0%B0%D0%BB%D1%8C%D0%BD%D0%B5_%D1%8F%D0%B2%D0%B8%D1%89%D0%B5)*”*

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 NP0 і NP1 встановлюється зв’язок BT;
* між представниками NP1 і NP0 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

1. Шаблони за Херстом, а саме:
   1. такий NP як {NP ,}\* {(і | або)} NP
   2. NP {, NP} \* {,} (або | і) інший NP
   3. NP {,} (включаючи | особливо) {NP,}\* {і | або} NP
2. 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.

1. 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

– множина текстових документів.

- множина лексикографічних шаблонів.

- множина термінів тезауруса.

– відсортований за метрикою і обмежений функцією список важливих однослівних термінів колекції

– множина багатослівних термінологічних словосполучень.

- множина зв’язків тезауруса,

- множина характеристичних фрагментів тексту для терміна *t.*

- множина речень характеристичного фрагменту *С.*

*–* множина лематизованих слів речення *S*

– послідовність збіжних з лексикографічним шаблоном термінологічних словосполучень.

– функція обмеження відсортованого списку термінів.

- функція видобування термінів з документу.

- функція, що будує послідовність відсортованих термінів документа за спаданням метрики TFIDF.

- функція обчислення частоту терміну в документі.

– функція, що ставить кожному терміну у відповідність його інвертовану документарну частоту з еталонної колекції.

- функція пошуку характеристичних фрагментів терміна.

- функція розбиття характеристичного фрагменту тексту на речення.

- функція видобуття послідовності лем з речення.

- функція задовільнення шаблону, що повертає множину послідовностей співпавших термінологічних словосполучень в порядку слідування позицій шаблону.

- функція встановлення зв’язків на множині послідовностей збігів до шаблону.

### 

### 2.2.2 Псевдокод розробленого методу

Рисунок 1. Формальний запис методу побудови тезауруса

|  |
| --- |
| 1. : 2. :    1. :       1. :          1. : якщо :          2. :             1. ; |

### 2.2.3 Формалізація правил збіжності з лексикографічним шаблоном

- множина лексикографічних шаблонів, задана як множина елементів шаблону.

– елементи шаблону.

; – множини команд пошуку термінологічних словосполучень, з вказанням головної (1) чи вторинної (0) ролі словосполучення у зв’язку в шаблоні.

– множина правил збіжності задана послідовностями тегів за частинами мови.

– множина команд пошуку прямої збіжності зі словом, задана на послідовностях альтернатив лем.

- множина команд пошуку вікон, що задана парами мінімальної і максимальної довжини вікна в кількості позицій лем у реченні.

- множина команд пошуку ітерацій, що задана на підпослідовностях елементів шаблону.

– множина фразових збігів, задана парами послідовностей лем і позиції першої леми.

– послідовність фразових збігів по операторам шаблону.

:PE - функція співставлення елементу шаблону з фразою, що ставить у відповідність множину фразових збігів.

### 

### 2.2.4 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* |

# 

# РОЗДІЛ III. ПРАКТИЧНА ЧАСТИНА

### 3.1 Збір даних для тестування розробленого методу

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. Тому для збору документів з відкритого доступу, зокрема випусків журналу “Наукові записки НаУКМА”, було розроблено спеціальний скрипт, що обходив сторінки сайтів архіву [2] і зберігав всі документи в директоріях по роках і темах. Розбиття по темах є необхідним для створення різних тематичних тезаурусів на етапі тестування. Всього з даної колекції було взято 2926 повнотекстових документів за 1996-2013 роки видання.

### 3.2 Розробка довідкового компоненту документарної частоти

#### 3.2.1 Рішення на основі зовнішньої пошукової системи

Документарну частоту по невеликій колекції, а там більше по одному документу, побудувати дуже важко. Набагато простіше взяти вже існуючу документарну частоту терміну з пошукової системи. В якості прикладу було обрано пошукову систему Google, що разом з результатами звичайного пошуку завжди повертає приблизну кількість результатів пошукової видачі. Даний показник і було взято за документарну частоту терміна. Під час побудови допоміжних методів щодо видобування даного показника з пошукової видачі, було прийнято рішення, що звичайний парсинг веб-сторінок пошукової видачі буде громіздким рішенням, оскільки задля одного числа прийдеться запитувати і розбирати цілу сторінку, що містить багато зайвих даних. Тому в рамках окремого програмного модулю дипломної роботи було написано клієнт сервісу Google Custom Search RESTful API [21], з налаштуванням всього необхідного в консолі сервісу Google. Як результат було отримано метод, що за даним терміном повертає його документарну частоту, причому кількість інтернет трафіку для таких запитів є мінімальною. Проте виникає інша проблема для використання Google API: квоти вільного доступу, що становлять 100 запитів на день. Звичайно, щоб обробляти великі документи і знайти, які терміни в них є важливими, необхідно дізнатись документарну частоту кожного. Тому було прийнято рішення, по-перше, зберігати в локальному кеші всі знайдені документарні частоти, і в першу чергу звертатись до нього, що з бігом часу призведе до досить повної бази термінів, і по-друге, залишити можливість обробляти звичайні веб-сторінки пошуку, на випадок вичерпання квоти.

Кеш, як і інші дані проміжної роботи алгоритму, було вирішено зберігати в документарній базі MongoDB.

#### 3.2.2 Рішення на основі індексації еталонної колекції документів

В якості еталонної колекції було обрано всі випуски журналу “Наукові записки НаУКМА”, отримані на етапі збору даних статей у відкритому доступі.

Попередньо структура каталогів еталонної колекції була вирівняна, і всі документи було зібрано в одній директорії. Для здійснення індексації колекції було використано рішення від Apache Lucene, так само як і в фінальному варіанті робочої системи. Доступ до програмного інтерфейсу бібліотеки індексації здійснювався з Java-коду, і спеціально для даної задачі створення початкового індексу документарних частот термінів була створена окрема конфігурація запуску програми. Основні особливості і кроки роботи даної програми наступні:

1. Зчитування параметрів директорії що містить документи колекції, а також окремої тимчасової конфігурації документарної бази даних MongoDB для зберігання проміжних результатів пошуку документарних частот.
2. Створення нового індексу Apache Lucene.
3. Подокументний розбір PDF і додавання текстів в індекс.
4. Перегляд індексу і зберігання документарних частот всіх термінів в документарну базу.
5. Дамп документарної бази для подальшого використання в якості початкового наповнення колекції документарних частот при звичайній роботі методу.

Зібраний дамп документарних частот зберігається в вихідних кодах програмної частини роботи у вигляді файлу даних із розширенням **.bson** і метаданих колекції термінів у файлі з розширенням **.json**. Дані файли мають бути імпортовані в нову базу на початку роботу алгоритму на новій системі.

Загалом після індексації колекції і фільтрації термінів, в дампі нараховується близько 143000 термінів.

### 3.3 Вибір готових рішень для реалізації кроків алгоритму.

Під час постановки задачі з розробки методу ітеративної побудови термінології, було передбачено існування готових рішень для вирішення елементарних кроків статистичних і лексикографічних методів, таких як розбір документів формату pdf, індексуваня, підрахунок частот термінів, пошук документів за фразою і тегування за частинами мови. Розглянемо обрані нами інструменти для вирішення даних прикладних задач.

Бібліотеки і утиліти *Apache Lucene* [14] – використано в якості основи для системи індексації вхідних текстових документів і розбиття на однослівні терміни, а також в якості базової системи пошуку характеристичних фрагментів тексту, що на основі побудованого індексу дозволяє знайти документи, в котрі входить шуканий термін, і в такий спосіб значно прискорюючи пошук порівняно з лінійним проходженням по всім документам.

*PDFBox* [15] – утилітарна бібліотека для розбору файлів у форматі pdf. Головною перевагою даного рішення є сумісність з форматом документа Apache Lucene, що дозволяє за один крок передавати розібраний документ одразу в підсистему індексації.

*Jlemmagen* [19] - один із ключових компонентів готових рішень для розбору україномовної термінології. Представляє собою колекцію утиліт для роботи в сфері обробки природної мови, зокрема містить лематизатор, що здатний приводити подані на вхід слова з тексту у нормальну форму відповідно до правил мови. Дане рішення є достатньо новим, і його основною перевагою є підтримка української мови, а також сумісність з мовними аналізаторами бібліотек індексації Apache Lucene.

*JlanguageTool* [34] – бібліотека з набором утилітарних методів роботи з текстом, що містить компонент тегування за частинами мови, підтримує українську мову, також може приводити слова у нормальну форму, а також синтезувати словоформи за вказаними тегами. Це рішення було використано в першу чергу для реалізації лексикографічних методів співставлення з шаблонами.

*Google Custom Search API* [21] *–* сервіс, що було використано для одного з підходів отримання репрезентативної документарної частоти терміна з зовнішньої пошукової системи.

*MongoDB* [13] – документарна база даних, що була використана в якості основного сховища розробленої системи. В базі зберігаються як самі терміни з обчисленими документарними частотами, так і зв’язки між термінами, що складають інформаційну основу тезуарусу. Вибір документарної бази обумовлений сумісністю форматів – обране рішення спроектовано для підтримки формату JSON, що в свою чергу є базою для формату JSON-LD як конкретної специфікації RDF, обраного нами для публікації тезаурусу. Таким чином, з архітектурної точки зору в системі буде присутній лише один формат даних як для збереження, так і для публікації даних клієнтам через веб-сервіси, що має забезпечити масштабованість і підтримуваність такої системи в майбутньому.

*Spring REST* [20] – java-фреймворк, призначений для підтримки розробки програмних систем у вигляді RESTful веб-сервісів. Використано як базову технологію під час розробки програмного прикладного інтерфейсу, що надає доступ до скачування готового тезаурусу, додавання нових документів в колекції, і навігації по термінах і зв’язках різних тезаурусів.

## 3.4 Архітектура системи обробки документів для побудови термінології

В якості системи, що мала б надати зручний доступ до функціональності розробленого методу, було вирішено побудувати веб-сервіс з можливістю побудови з наявних в користувача документів тезаурусу в форматі RDF.

Компоненту схему такої системи подано на рисунку 2.

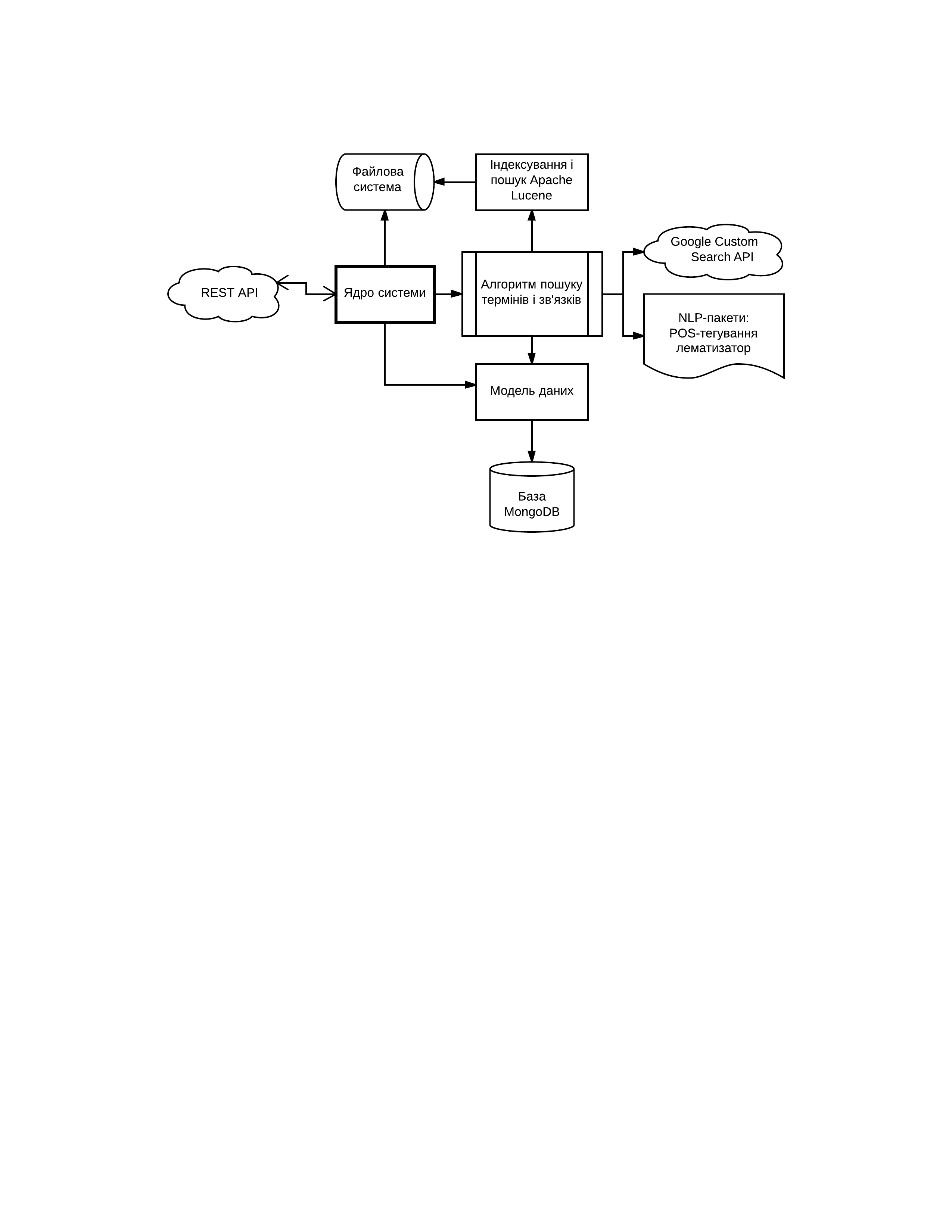
Відповідно до наведеної схеми, роль доступу до функціональності ядра системи відведено на рівень REST контролерів системи, функції ядра полягають в обробці запитів від програмного інтерфейсу і інтеграції розроблених компонентів системи і алгоритмів. Зокрема, ядро має доступ до файлової системи для збереження отриманих під час завантаження в колекцію документів, а також опосередковано через проміжну модель даних отримує доступ до документарної бази під час серіалізації тезаурусу в форматі RDF. До алгоритмів пошуку термінів і зв’язків, що в свою чергу тісно пов’язані з підсистемами індексації документів і допоміжними бібліотеками тегування за частинами мови, відносяться статистичні методи пошуку важливих на основі зважування, методи пошуку зв’язків за лексикографічними шаблонами, і допоміжні підходи, пов’язані з доступом до даних, пошуком характеристичних фрагментів тексту, узгодженням багатослівних словосполучень.

Рисунок 2. Схема компонентів системи побудови тезаурусів

Детальніше реалізацію компонентів системи можна представити у вигляді діаграми класів, поданої на рисунку 3.

Класи *IndexController* і *RDFController* відповідають за надання HTTP-інтерфейсу веб-сервісу до функцій керування колекціями, додавання нових документів і перегляду тезаурусів у декількох варіантах форматування.

Група класів *Termin*, *TermInDoc* і *TermRelation* належать до моделі даних і використовуються як на етапі застосування алгоритму пошуку, так і в якості специфікації структури відповідних сутностей документарної бази MongoDB. За допомогою абстракції репозиторію, що надається фреймворком Spring Data [41], ми маємо змогу застосувати статичну типізацію інтерфейсів відповідних *TerminRepository*, *TermInDocRepository* і *TermRelationRepository*, для того щоб пов’язати наведені проксі-класи з бібліотеками роботи з базою даних.

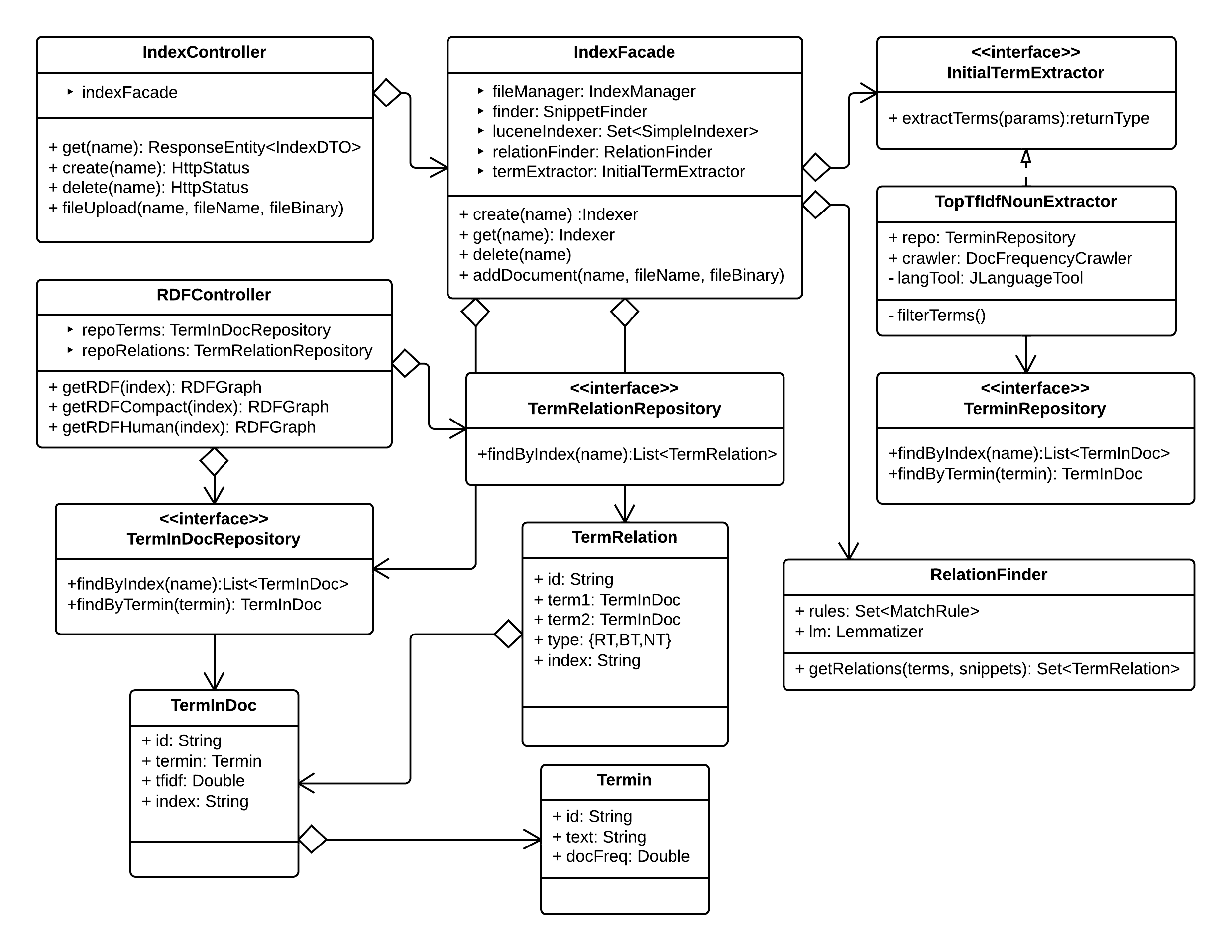


Рисунок 3. UML-діаграма основних класів системи побудови тезаурусів

Клас *IndexFacade* виконує роль ядра системи, поєднуючи компоненти системи в певні функціональні блоки для виконання основних операції, таких як додавання і індексація нового документа.

Серед класів, що безпосередньо містять методи пошуку термінів і зв’язків, ключову роль грають *TopTfIDfNounExtractor* і *RelationFinder*. Перший виконує інструкції статистичного етапу розробленого методу з пошуку важливих термінів, їх зважуванням, сортуванням і фільтрацією отриманого списку термінів. Другий безпосередньо містить статично задану колекцію лексикографічних шаблонів, і видобуває за їх допомогою зв’язки між термінами, почергово застосовуючи шаблони до списків запропонованих характеристичних фрагментів тексту.

Крім наведених на діаграмі класів, в програмному коді реалізації існують й інші утилітарні компоненти системи, пов’язані з інтеграцію з готовими рішеннями і вирішенням прикладних задач по обробці тексту і управління колекціями на рівні користувача.

## 3.5 Специфікація програмного інтерфейсу доступу до системи.

Прикладний програмний інтерфейс розробленої системи представляє собою спосіб доступу користувача або програми до можливостей створення і перегляду тезаурусів. Доступ до системи здійснюється за допомогою двох способів:

* RESTful API, що дозволяє звертатись до системи по протоколу HTTP;
* консольне застосування для локальної пакетної обробки колекцій документів, що зокрема використовується для тестування алгоритму.

### 3.5.1 Точка доступу до RDF-тезаурусу

Таблиця 2. Cпецифікація точки доступу до програмного інтерфейсу роботи з тезаурусом

|  |  |  |
| --- | --- | --- |
| HTTP-метод | URI | Призначення |
| GET | /rdf/*{indexName}*/ | Отримати тезаурус за іменем |
| GET | /rdf/*{indexName}*/compact | Доступ до тезауруса в компактній формі. До списку термінів включені тільки ті, для яких знайдено зв’язки |
| GET | /rdf/*{indexName}*/human | Вивід тезауруса в придатній для читання формі, зручній для перегляду зв’язків |

### 3.5.2 Точка доступу до керування індексами

Таблиця 3. Специфікація точки доступу до програмного інтерфейсу керування колекціями

|  |  |  |
| --- | --- | --- |
| HTTP-метод | URI | Призначення |
| GET | /index/ | Отримати список колекцій документів в системі |
| POST | /index/*{indexName}/* | Створення нової колекції з ім’ям |
| GET | /index/*{indexName}*/ | Перегляд статистичної інформації по колекції: назва, кількість документів, термінів і знайдених зв’язків. |
| DELETE | /index/*{indexName}* | Видалення колекції і пов’язаного тезаурусу. |
| POST | /index/*{indexName}*/upload | Завантаження файлу у форматі pdf до колекції документів. Даний метод ініціює запуск алгоритму перебудови тезаурусу із врахуванням даних нового документу. |

### 3.6 Формат серіалізації тезаурусу на базі JSON-LD

До основних секцій даного формату відносяться вузол **@context**, в котрому наводиться перелік скорочень до типів даних усередині тіла документу, і власне початковий вузол тіла документа, позначений як **@graph**. В рамках роботи над форматом представлення тезаурусів було проведено аналіз формату JSON-LD, особливо рекомендацій щодо позначення абстрактних концептів RDF для тезаурусів відповідно до стандарту ISO-25964 [31]. В результаті було визначено мінімальний набір полів і їх типів, що мають задовольнити наші потреби з серіалізації термінів і зв’язків. Посилання на відповідні типи даних зазначені у вузлі контексту.

Рисунок 4. Приклад серіалізованого у форматі JSON-LD тезаурусу

{

"@context" : {

"iso25964" : "http://www.niso.org/schemas/iso25964/iso25964-1\_v1.4.xsd#",

"thesaurus" : "iso25964:Thesaurus",

"concept" : "iso25964:ThesaurusConcept",

"relation" : "iso25964:HierarchicalRelationship",

"role" : "iso25964:role",

"baseConcept" : "iso25964:isHierRelConcept",

"depConcept" : "iso25964:hasHierRelConcept",

"lexicalValue" : "iso25964:lexicalValue"

},

"@graph" : {

"thesaurus" : {

"concept" : [

{

"@id" : "C1",

"lexicalValue" : "наука"

}, {

"@id" : "C2",

"lexicalValue" : "філософія"

}

],

"relation" : [

{

"baseConcept" : "C1",

"depConcept" : "C2",

"role" : "NT"

}

]

}

# 

Безпосередньо до вузла тезаурусу в тілі документа, згідно з зазначеним форматом, розміщуються колекції концептів і зв’язків. При цьому структура концептів пов’язує їх ідентифікатори з лексичними значеннями термінів, а зв’язки в свою чергу пов’язують ідентифікатори термінів між собою з вказанням типу зв’язку.

# РОЗДІЛ IV. ОБГОВОРЕННЯ РЕЗУЛЬТАТІВ

Застосування алгоритму для побудови термінології у вигляді RDF схеми з використанням синтетичних та реальних даних україномовної наукової періодики.

## 4.1 Схема тестування та оцінка результатів

Для тестування роботи алгоритму було вирішено розробити систему конфігурацій для зручного і гнучкого прогону різних варіацій алгоритму на різних даних, та водночас з необхідною оптимізацією швидкості таких тестувань, а також із заміром основних характеристик виконання окремих кроків алгоритму.

Найбільше для даного випадку підходить реалізація окремих інтеграційних тестів, що міститимуть незалежні конфігурації оточення, такі як реалізація документарної бази у пам’яті [40].

Схема тестування є необхідним компонентом системи як для дослідження точності знайдених термінологічних зв’язків між термінами, так і для розгляду модифікацій алгоритму для пошуку найбільш ефективного підбору параметрів алгоритму, що буде включено в фінальну реалізацію системи як веб-сервісу.

Під час розробки схеми тестування було проведено наступні роботи:

* розроблено єдиний формат тезаурусу для порівняння;
* зафіксовано модифікацію розробленого методу;
* налаштовано тестове середовище, що дозволяло запускати прогони алгоритмів;
* знайти сукупність реальних текстових наукових матеріалів для порівняння роботи на них алгоритмів;
* розроблено метрику точності пошуку.

Для перевірки результатів роботи алгоритму було вирішено побудувати декілька тезаурусів на основі різних тематичних розділів україномовного журналу “Наукові Записки НаУКМА”, зібравши колекції текстів по різних темах за декілька років, а також двох контрольних тезаурусів, не пов’язаних з даним журналом. Контрольні тезауруси планується наповнювати документами з періодичних видань на дві різних теми, не включених в еталонну колекцію. Такий контроль є необхідним для того щоб мінімізувати вплив початкового фільтру важливих термінів, що в даній роботі побудований на основі всіх тем журналу “Наукові записки НаУКМА” за всі роки, і тому завідомо краще працюючому, ніж з довільною колекцією.

В звідну таблицю, по кожному тезаурусу було зібрано наступну статистику:

* кількість документів в базовій колекції текстів,
* кількість знайдених термінів при початковому фільтруванні,
* кількість доданих термінологічних словосполучень і загальна кількість термінів,
* кількість знайдених зв’язків,
* кількість зв’язків по типах RT, BT і NT,
* виміряний коефіцієнт точності пошуку зв’язків.

Коефіцієнт точності методу вимірювався наступним чином. Зі сформованого тезаурусу обиралися випадково N знайдених зв’язків, і проводилась оцінка релевантності зв’язку між термінами, на основі звертання до первинного тексту.

Коефіцієнт точності мав наступне правило обрахунку:

Де:

* N1 відповідає кількості релевантних зв’язків з N, без урахування напрямку зв’язку;
* N2 – кількості правильно розпізнаного за напрямком зв’язків типу BT і NT відповідно.

Введені в формулу обрахунку вагові коефіцієнти мають підкреслити першочергову важливість знаходження простих зв’язків.

В додатках подано відформатовані для сприйняття людиною результати у вигляді побудованих тезаурусів у форматі RDF.

## 4.2 Результати тестування методу на тематичних колекціях документів

Таблиця 4. Результати тестування методу на колекціях наукових текстів

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **№** | **Назва колекції** | **Кількість документів** | **Знайдено термінів** | **Побудовано зв’язків** | | **Релевантність** | | | **Оцінка точності,** |
| **RT** | **BT,NT** | **N** | **N1** | **N2** |
| **1** | Комп’ютерні науки | 96 | 785 | 47 | 195 | 20 | 18 | 14 | 0.86 |
| **2** | Філософія і релігієзнавство | 45 | 1026 | 35 | 234 | 20 | 16 | 12 | 0.76 |
| **3** | Економічні науки | 58 | 890 | 20 | 195 | 20 | 15 | 15 | 0.75 |
| **4** | Юридичні науки | 66 | 748 | 51 | 47 | 20 | 13 | 10 | 0.62 |
| **5** | Біотехнології, контрольна колекція [1] | 35 | 464 | 10 | 142 | 20 | 12 | 11 | 0.59 |
| **6** | Соціологія, контрольна колекція [8] | 27 | 627 | 28 | 94 | 20 | 14 | 9 | 0.65 |

В середньому, точність розробленого методу на даних тестових запусків становить 70.5 %, і ми вважатимемо дану точність прийнятною для даного методу.

В результаті тестування на прикладах колекцій були помічені наступні закономірності:

* більшість збігів припадає на шаблон LP1, що позначає прямі дефініції в тексті;
* як і було передбачено, метод надав меншої точності пошуку на контрольних колекціях, проте деградація точності не набула суттєвої величини;
* шаблони типу LP2-4 знаходили досить мало зв’язків, близько 20 на 6000 текстових фрагментів.

Помічені закономірності свідчать про те, що лексикографічний метод є занадто чутливим до формального запису окремих шаблонів, а також вказує на недостатню повноту колекції за розміром і покриттям наукової сфери. Дійсно, колекції з 50-90 документів різнобічних статей, об’єднаних тільки широкою тематикою наукової галузі, що зустрічаються в журналі “Наукові записки НаУКМА” і в інших періодичних виданнях, не можна вважати достатньо великими і повними, щоб метод лексикографічних шаблонів за Хеарстом спрацював на повну потужність.

## 4.3 Пропозиції щодо збільшення повноти і точності методу

1. Використання стохастичного методу для усунення неоднозначностей в трактуванні тегів частин мови слів у контексті.
2. Використання більшої кількості граматичних правил узгодження слів в термінологічних словосполученнях під час приведення фраз до нормальної форми.
3. Збільшення кількості лексикографічних шаблонів і довжини синонімічних рядів визначальних лексем шаблону для досягнення більшої повноти пошуку зв’язків.
4. Застосовування більших колекцій документів для аналізу, а також для побудови еталонної колекції термінів документарних частот.

# ВИСНОВКИ

В рамках даної роботи було вирішено задачу ітеративної побудови термінології в колекціях наукових текстів. Результати роботи представлено у вигляді розробленого веб-сервісу з можливостями побудови тезаурусів у форматі RDF з вихідних текстів формату pdf. Формат тезаурусу JSON-LD було обрано з врахуванням можливості публікації отриманих термінологічних зв’язків у стандартизованому вигляді мережного доступу до ресурсів, і з позицій розуміння тезауруса як повноцінного програмного модулю пошукової системи наукових матеріалів. 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 розробленого модулю побудови тезаурусів покладено розвинений 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. В якості такої колекції було використано архів україномовної періодики “Наукові записки НаУКМА”, на основі якого було побудовано довідковий індекс документарних частот термінів.

Другий етап розробленого методу пов’язаний із застосуванням лексикографічних шаблонів для пошуку гіпонімічних зв’язків у вихідних текстах. Під час пошуку вдалої реалізації було використано відкрите 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.

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>

1. Українська мова: енциклопедія. – Вид-во" Українська енцклопедія" ім. МП Бажана, 2004.
2. "Agile and Scalable." *MongoDB*. Web. 05 April 2014. <http://www.mongodb.org/>.
3. "Apache Lucene Core." *Apache Lucene -*. Web. 08 April 2014. <http://lucene.apache.org/core/>.
4. "Apache PDFBox - A Java PDF Library."*Apache PDFBox*. Web. 08 June 2014. <http://pdfbox.apache.org/>.
5. 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.
6. 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.
7. Alshawi H. Processing dictionary definitions with phrasal pattern hierarchies //Computational Linguistics. – 1987. – Т. 13. – №. 3-4. – С. 195-202.
8. "Bitbucket." *Hlavki / JLemmaGen —*. Web. 08 April 2014. <https://bitbucket.org/hlavki/jlemmagen>.
9. "Building a RESTful Web Service."*Getting Started ·*. Web. 08 April 2014. <https://spring.io/guides/gs/rest-service/>.
10. "Custom Search." *— Google Developers*. Web. 08 April 2014. <https://developers.google.com/custom-search/json-api/v1/overview>.
11. 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.
12. Chen H. et al. Automatic thesaurus generation for an electronic community system. – 1995.
13. Chen H. et al. Generating, integrating, and activating thesauri for concept-based document retrieval //IEEE expert. – 1993. – Т. 8. – №. 2. – С. 25-34.
14. 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.
15. DOSTÁL J. Termín a jeho definice ve výkladových, terminologických a naučných slovnících a encyklopediích //Modernizace výuky v technicky orientovaných předmětech a oborech. – С. 57.
16. "Designing and Implementing RESTful Web Services with Spring." *Tutorial ·*. Web. 12 May 2014. <https://spring.io/guides/tutorials/rest/2/>.
17. Grefenstette G. Automatic thesaurus generation from raw text using knowledge-poor techniques. – Rank Xerox Research Centre, 1993.
18. “HATEOAS”. *Wikipedia*. Wikimedia Foundation. Web. 03 April 2014. <http://en.wikipedia.org/wiki/HATEOAS>.
19. 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.
20. "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/>.
21. JSON-LD 1.0. Web. 08 June 2014. <http://www.w3.org/TR/json-ld/>.
22. Kryslová M. Stanovlennja ukrajinskoji linhvistyčnoji terminolohiji. – 2009.
23. "LanguageTool Wiki." *Java API -*. Web. 08 April 2014. <http://wiki.languagetool.org/java-api>.
24. Lassi M. Automatic thesaurus construction //University Collage of Boras, Sweden. – 2002.
25. Manola F., Miller E., McBride B. RDF primer, 2004 //URL <http://www.w3.org/TR/rdf-primer>. – 2010.
26. Miller U. Thesaurus construction: problems and their roots //Information Processing & Management. – 1997. – Т. 33. – №. 4. – С. 481-493.
27. "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-rdf11-concepts-20140109/>>.
28. 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.
29. "Spring MongoDB Tutorial." *Spring MongoDB Tutorial*. Web. 08 April 2014. <http://bits-and-kites.blogspot.com/2014/01/spring-mongodb-tutorial.html>.
30. "Spring Data MongoDB." *Spring Data MongoDB*. Web. 08 April 2014. <http://projects.spring.io/spring-data-mongodb/>.
31. Strzalkowski T. Natural language information retrieval //Information Processing & Management. – 1995. – Т. 31. – №. 3. – С. 397-417.
32. "WordNet." *About  -*. Web. 08 June 2014. <http://wordnet.princeton.edu/>.

# 

# ДОДАТКИ

**Додаток А (довідниковий)**

Приклад утвореного тезаурусу по колекції текстів з тематики комп’ютерних наук у форматі JSON-LD

|  |
| --- |
| {  "@context": {  "iso25964": "http://www.niso.org/schemas/iso25964/iso25964-1\_v1.4.xsd#",  "thesaurus": "iso25964:Thesaurus",  "concept": "iso25964:ThesaurusConcept",  "relation": "iso25964:HierarchicalRelationship",  "role": "iso25964:role",  "baseConcept": "iso25964:isHierRelConcept",  "depConcept": "iso25964:hasHierRelConcept",  "lexicalValue": "iso25964:lexicalValue"  },  "@graph": {  "thesaurus": {  "concept": [  {"@id":"C1","lexicalValue":"технологія"},  {"@id":"C2","lexicalValue":"квантові обчислення"},  {"@id":"C3","lexicalValue":"бізнес"},  {"@id":"C4","lexicalValue":"царина"},  {"@id":"C5","lexicalValue":"суспільство"},  {"@id":"C6","lexicalValue":"цифрова нерівність"},  {"@id":"C7","lexicalValue":"критерій"},  {"@id":"C8","lexicalValue":"інформативність"},  {"@id":"C9","lexicalValue":"частинка"},  {"@id":"C10","lexicalValue":"молекула"},  {"@id":"C11","lexicalValue":"освіта"},  {"@id":"C12","lexicalValue":"відео"},  {"@id":"C13","lexicalValue":"графік"},  {"@id":"C14","lexicalValue":"атом"},  {"@id":"C15","lexicalValue":"середовище"},  {"@id":"C16","lexicalValue":"нахабна поведінка"},  {"@id":"C17","lexicalValue":"електронна бібліотека"},  {"@id":"C18","lexicalValue":"пошук"},  {"@id":"C19","lexicalValue":"здоров'я"},  {"@id":"C20","lexicalValue":"інтернет"},  {"@id":"C21","lexicalValue":"як конвергенція"},  {"@id":"C22","lexicalValue":"організаційні потреба"},  {"@id":"C23","lexicalValue":"стислість"},  {"@id":"C24","lexicalValue":"канал"},  {"@id":"C25","lexicalValue":"квантова фізика"},  {"@id":"C26","lexicalValue":"світ"},  {"@id":"C27","lexicalValue":"звук"},  {"@id":"C28","lexicalValue":"достовірність"},  {"@id":"C29","lexicalValue":"операція"},  {"@id":"C30","lexicalValue":"додавання"},  {"@id":"C31","lexicalValue":"компанія"},  {"@id":"C32","lexicalValue":"чистий зв'язок"},  {"@id":"C33","lexicalValue":"розпад родити"},  {"@id":"C34","lexicalValue":"поведінка"},  {"@id":"C35","lexicalValue":"чистий зміст"},  {"@id":"C36","lexicalValue":"кінокомпанія"},  {"@id":"C37","lexicalValue":"банда"},  {"@id":"C38","lexicalValue":"контроль"},  {"@id":"C39","lexicalValue":"дані"},  {"@id":"C40","lexicalValue":"комп'ютер"},  {"@id":"C41","lexicalValue":"розпад"},  {"@id":"C42","lexicalValue":"як"},  {"@id":"C43","lexicalValue":"вираз"},  {"@id":"C44","lexicalValue":"конструкція"},  {"@id":"C45","lexicalValue":"квантовий"},  {"@id":"C46","lexicalValue":"матеріал"},  {"@id":"C47","lexicalValue":"анотація"},  {"@id":"C48","lexicalValue":"анотацію курс"},  {"@id":"C49","lexicalValue":"метод"}  "relation": [  {"baseConcept":"C1","depConcept":"C2","role":"NT"},{"baseConcept":"C3","depConcept":"C4","role":"BT"},{"baseConcept":"C5","depConcept":"C6","role":"NT"},{"baseConcept":"C7","depConcept":"C8","role":"NT"},{"baseConcept":"C9","depConcept":"C10","role":"NT"},{"baseConcept":"C4","depConcept":"C11","role":"NT"},{"baseConcept":"C4","depConcept":"C3","role":"NT"},{"baseConcept":"C12","depConcept":"C13","role":"RT"},{"baseConcept":"C14","depConcept":"C10","role":"RT"},{"baseConcept":"C15","depConcept":"C16","role":"NT"},{"baseConcept":"C11","depConcept":"C4","role":"BT"},{"baseConcept":"C17","depConcept":"C18","role":"BT"},{"baseConcept":"C19","depConcept":"C11","role":"RT"},{"baseConcept":"C20","depConcept":"C21","role":"NT"},{"baseConcept":"C19","depConcept":"C22","role":"BT"},{"baseConcept":"C22","depConcept":"C19","role":"NT"},{"baseConcept":"C3","depConcept":"C11","role":"RT"},{"baseConcept":"C7","depConcept":"C23","role":"NT"},{"baseConcept":"C20","depConcept":"C24","role":"BT"},{"baseConcept":"C25","depConcept":"C26","role":"BT"},{"baseConcept":"C26","depConcept":"C25","role":"NT"},{"baseConcept":"C12","depConcept":"C27","role":"RT"},{"baseConcept":"C28","depConcept":"C7","role":"BT"},{"baseConcept":"C29","depConcept":"C30","role":"NT"},{"baseConcept":"C31","depConcept":"C32","role":"NT"},{"baseConcept":"C33","depConcept":"C34","role":"BT"},{"baseConcept":"C35","depConcept":"C36","role":"BT"},{"baseConcept":"C37","depConcept":"C38","role":"NT"},{"baseConcept":"C39","depConcept":"C27","role":"NT"},{"baseConcept":"C40","depConcept":"C1","role":"NT"},{"baseConcept":"C36","depConcept":"C35","role":"NT"},{"baseConcept":"C34","depConcept":"C41","role":"NT"},{"baseConcept":"C20","depConcept":"C42","role":"NT"},{"baseConcept":"C2","depConcept":"C1","role":"BT"},{"baseConcept":"C42","depConcept":"C20","role":"BT"},{"baseConcept":"C13","depConcept":"C39","role":"BT"},{"baseConcept":"C27","depConcept":"C12","role":"RT"},{"baseConcept":"C23","depConcept":"C7","role":"BT"},{"baseConcept":"C14","depConcept":"C9","role":"BT"},{"baseConcept":"C18","depConcept":"C17","role":"NT"},{"baseConcept":"C34","depConcept":"C33","role":"NT"},{"baseConcept":"C30","depConcept":"C29","role":"BT"},{"baseConcept":"C38","depConcept":"C37","role":"BT"},{"baseConcept":"C11","depConcept":"C19","role":"RT"},{"baseConcept":"C22","depConcept":"C11","role":"NT"},{"baseConcept":"C43","depConcept":"C44","role":"BT"},{"baseConcept":"C11","depConcept":"C3","role":"RT"},{"baseConcept":"C32","depConcept":"C31","role":"BT"},{"baseConcept":"C7","depConcept":"C28","role":"NT"},{"baseConcept":"C28","depConcept":"C23","role":"RT"},{"baseConcept":"C20","depConcept":"C5","role":"BT"},{"baseConcept":"C27","depConcept":"C39","role":"BT"},{"baseConcept":"C39","depConcept":"C13","role":"NT"},{"baseConcept":"C1","depConcept":"C45","role":"NT"},{"baseConcept":"C21","depConcept":"C20","role":"BT"},{"baseConcept":"C27","depConcept":"C13","role":"RT"},{"baseConcept":"C39","depConcept":"C12","role":"NT"},{"baseConcept":"C8","depConcept":"C7","role":"BT"},{"baseConcept":"C24","depConcept":"C20","role":"NT"},{"baseConcept":"C8","depConcept":"C23","role":"RT"},{"baseConcept":"C23","depConcept":"C28","role":"RT"},{"baseConcept":"C45","depConcept":"C1","role":"BT"},{"baseConcept":"C12","depConcept":"C39","role":"BT"},{"baseConcept":"C23","depConcept":"C8","role":"RT"},{"baseConcept":"C1","depConcept":"C40","role":"BT"},{"baseConcept":"C8","depConcept":"C28","role":"RT"},{"baseConcept":"C13","depConcept":"C27","role":"RT"},{"baseConcept":"C10","depConcept":"C14","role":"RT"},{"baseConcept":"C16","depConcept":"C15","role":"BT"},{"baseConcept":"C5","depConcept":"C20","role":"NT"},{"baseConcept":"C28","depConcept":"C8","role":"RT"},{"baseConcept":"C6","depConcept":"C5","role":"BT"},{"baseConcept":"C41","depConcept":"C34","role":"BT"},{"baseConcept":"C44","depConcept":"C43","role":"NT"},{"baseConcept":"C11","depConcept":"C22","role":"BT"},{"baseConcept":"C13","depConcept":"C12","role":"RT"},{"baseConcept":"C10","depConcept":"C9","role":"BT"},{"baseConcept":"C9","depConcept":"C14","role":"NT"},{"baseConcept":"C46","depConcept":"C47","role":"NT"},{"baseConcept":"C48","depConcept":"C46","role":"BT"},{"baseConcept":"C46","depConcept":"C48","role":"NT"},{"baseConcept":"C40","depConcept":"C49","role":"NT"},{"baseConcept":"C47","depConcept":"C46","role":"BT"},{"baseConcept":"C49","depConcept":"C40","role":"BT"}  ]  }  }  } |

**Додаток Б (довідниковий)**

Лістинг програми збору даних архіву журналу “Наукові записки НаУКМА”. Файл *naukmapdf.groovy*

|  |
| --- |
| @Grab(group='org.ccil.cowan.tagsoup',  module='tagsoup', version='1.2' )  class NaukmaGrabber {  static File logfile = new File("/Users/user/naukma/grabbing.log")  static String basePath = "/Users/user/naukma/pdfs"  static File baseDir = new File(basePath)  static int docCount = 0;  static def tagsoupParser = new org.ccil.cowan.tagsoup.Parser() as Object  static def slurper = new XmlSlurper(tagsoupParser)  public static void main(String[] args) {  long startTime = System.currentTimeMillis()  if(!logfile.exists())logfile.createNewFile();  if(!baseDir.exists())baseDir.mkdir();  logfile.withWriterAppend {f -> f << new Date().getDateTimeString() +" \_\_ " + "Start pdfs grabbing"+"\n"}  def text = ("http://nz.ukma.edu.ua/index.php?option=com\_content&task=section&id=10&Itemid=47").toURL().getText();  def xml = slurper.parseText(text)  xml."\*\*".findAll { it.@class.toString().contains("category")}.eachWithIndex {  elem, index ->  String year = elem.text().replaceAll("\\s+","")  processYear(year, elem.@href)  }  long endTime = System.currentTimeMillis()  logfile.withWriterAppend {f -> f << new Date().getDateTimeString() +" \_\_ " + "TIME SPENT: "+ (endTime-startTime)/1000.0 + " seconds"+"\n"}  logfile.withWriterAppend {f -> f << new Date().getDateTimeString() +" \_\_ " +"End grabbing pdfs"+"\n"}  logfile.withWriterAppend {f -> f << new Date().getDateTimeString() +" \_\_ " +"PDFs collected: " + docCount + "\n"}  }  public static void processYear(year, link){  String yearPath = basePath + "/" + year;  File yearDir = new File(yearPath)  if(!yearDir.exists()) yearDir.mkdir()  logfile.withWriterAppend {f -> f << new Date().getDateTimeString() +" \_\_ " + "Start year: "+ year +"\n"}  def text = (link).toURL().getText("windows-1251");  def xml = slurper.parseText(text)  xml."\*\*".findAll { it.@class.toString().contains("sectiontableentry")}.eachWithIndex {  elem, index ->  String topicText = elem.td.text().replaceAll("\\s+","\_");  String linkTo = elem.td.a.@href;  processTopic(yearPath, topicText, linkTo)  }  }  public static void processTopic(yearPath, topic, link){  String topicPath = yearPath + "/" + topic;  File topicDir = new File(topicPath)  if(!topicDir.exists()) topicDir.mkdir()  logfile.withWriterAppend {f -> f << new Date().getDateTimeString() +" \_\_ " + "Start topic: "+ topic +"\n"}  def text = (link).toURL().getText("windows-1251");  def xml = slurper.parseText(text)  xml."\*\*".findAll { it.@target.equals("\_blank") && it.text().toString().contains("Повний текст")}.eachWithIndex {  elem, index ->  String linkTo = elem.@href  if(linkTo.endsWith(".pdf")){  String filename = linkTo.substring(linkTo.lastIndexOf("/") + 1)  String filePath = topicPath + "/" + filename  logfile.withWriterAppend {f -> f << new Date().getDateTimeString() +" \_\_ " + "Save document: "+ filePath +"\n"}  saveFileFromURL(linkTo, filePath)  docCount++  } else {  processDocument(topicPath, linkTo)  }  }  }  public static void processDocument(topicPath, link){  logfile.withWriterAppend {f -> f << new Date().getDateTimeString() +" \_\_ " + "Start document: "+ link +"\n"}  try{  def text = (link).toURL().getText();  def xml = slurper.parseText(text)  xml."\*\*".findAll { it.@target.equals("\_blank") && it.@href.toString().endsWith(".pdf") && it.text().toString().endsWith(".pdf")}.eachWithIndex {  elem, index ->  String linkTo = elem.@href  if(linkTo.endsWith(".pdf")){  String filename = linkTo.substring(linkTo.lastIndexOf("/") + 1)  String filePath = topicPath + "/" + filename  logfile.withWriterAppend {f -> f << new Date().getDateTimeString() +" \_\_ " + "Save document: "+ filePath +"\n"}  saveFileFromURL("http://www.ekmair.ukma.kiev.ua" + linkTo, filePath)  docCount++  }  }  } catch (exc){  logfile.withWriterAppend {f -> f << new Date().getDateTimeString() +" \_\_ " + "Error: "+ exc +"\n"}  }  }  public static saveFileFromURL(link, file){  if(!new File(file).exists()){  try{  def f = new File(file).newOutputStream()  f << new URL(link).openStream()  f.close()  } catch (exc){  logfile.withWriterAppend {f -> f << new Date().getDateTimeString() +" \_\_ " + "Error: "+ exc +"\n"}  }  }  }  } |

**Додаток В (довідниковий)**

Лістинг підпрограми пакетної обробки колекції документів для побудови тезаурусу. Файл *FullProcessCollectionApplication.java*

|  |
| --- |
| @Configuration  @PropertySource(value = "classpath:init.properties")  @Import(MongoConfiguration.class)  @ContextConfiguration(classes = FullProcessCollectionApplication.class)  @ComponentScan(basePackages = {  "edu.naukma.reshet.core.model",  "edu.naukma.reshet.core.algorithm",  "edu.naukma.reshet.core.help",  "edu.naukma.reshet.repositories"  })  @EnableAutoConfiguration  public class FullProcessCollectionApplication implements CommandLineRunner{  static String path = "/Users/user/naukma/";  @Bean  public static PropertySourcesPlaceholderConfigurer getConfigurator(){  return new PropertySourcesPlaceholderConfigurer();  }  public static PdfDirectoryIndexer indexer(String collectionPath, String name){  return new PdfDirectoryIndexer(path + collectionPath, path + "index/", name);  }  public static void main(String args[]){  SpringApplication.run(FullProcessCollectionApplication.class, args);  }  @Autowired  TerminRepository repo;  @Autowired  TermInDocRepository repoTermInDoc;  @Autowired  TermRelationRepository repoRelation;  @Autowired  TopTfIdfInitialTerminologyNounExtractor extractor;  private Lemmatizer lemmatizer;  private void extractCollectionRelations(String indexName){  long startTime = System.currentTimeMillis();  Searcher2 searcher = new AdvancedTextSearcher(path + "index/"+indexName+"/lucene/");  SnippetsFinder snipFinder = new SnippetsFinder();  RelationFinder relFinder = new RelationFinder();  relFinder.setLm(lemmatizer);  snipFinder.setLm(lemmatizer);  snipFinder.setSearcher(searcher);  List<TermInDoc> termsAll = repoTermInDoc.findByIndex(indexName);  List<Snippet> snippets = snipFinder.findSnippets(termsAll);  Set<TermRelation> relations = relFinder.getRelations(termsAll, snippets);  Set<TermRelation> validRelations = prepareRelations(relations, indexName);  repoRelation.save(validRelations);  long endTime = System.currentTimeMillis();  System.out.println("Time spent: " + (endTime - startTime) / 1000 + " sec.");  }  private Set<TermRelation> prepareRelations(Set<TermRelation> relations, String index){  final Set<String> uniqueRelationsPool = Sets.newHashSet();  final Set<TermRelation> validRelations = Sets.newHashSet();  for(TermRelation relation: relations){  String hash = relation.getRelationType() + "|" +  relation.getTerm1().getTermin().getText() + "|" +  relation.getTerm2().getTermin().getText();  if (!uniqueRelationsPool.contains(hash)) {  uniqueRelationsPool.add(hash);  String termin1 = relation.getTerm1().getTermin().getText();  String termin2 = relation.getTerm2().getTermin().getText();  Termin term1 = repo.findByText(termin1);  Termin term2 = repo.findByText(termin2);  if (term1 == null) {  repo.save(relation.getTerm1().getTermin());  term1 = repo.findByText(termin1);  }  if (term2 == null) {  repo.save(relation.getTerm2().getTermin());  term2 = repo.findByText(termin2);  }  relation.getTerm1().setTermin(term1);  relation.getTerm2().setTermin(term2);  TermInDoc t1 = repoTermInDoc.findByTermin(term1);  TermInDoc t2 = repoTermInDoc.findByTermin(term2);  if(t1 == null){  repoTermInDoc.save(relation.getTerm1());  t1 = repoTermInDoc.findByTermin(term1);  }  if(t2 == null){  repoTermInDoc.save(relation.getTerm2());  t2 = repoTermInDoc.findByTermin(term2);  }  TermRelation validRelation = new TermRelation(  t1,  t2,  relation.getRelationType(), index);  validRelations.add(validRelation);  }  }  return validRelations;  }  private Map<String, Double> getCollectionTerms(String indexPath){  System.out.println("Reference collection documentary frequency extractor application");  SimpleTextSearcher searcher = new SimpleTextSearcher(path + indexPath, "science");  Map<String, Double> map = searcher.getTermsIDFs();  Map<String, Double> filtered\_map = filterTerms(map);  System.out.println("Original map size:" + map.size());  System.out.println("Filtered map size:" + filtered\_map.size());  return filtered\_map;  }  private void saveReferenceTerms(){  Map<String, Double> map = getCollectionTerms("index/science/lucene/");  List<String> list = sortByValue(map);  for(String key: list){  repo.save(new Termin(key, map.get(key)));  }  System.out.println("IDFs calculated.");  }  private void extractCollectionTerms(String indexName){  SimpleTextSearcher searcher =  new SimpleTextSearcher(path + "index/"+indexName+"/lucene/", indexName);  int totalDocs = searcher.getIndexReader().numDocs();  List<TermInDoc> allTerms = Lists.newLinkedList();  for(int i = 0; i < totalDocs; i++){  List<TermInDoc> terms = extractor.extractValuableTerms(searcher, i);  System.out.println("Terms count: " + terms.size());  allTerms.addAll(terms);  }  List<TermInDoc> top = Ordering  .natural()  .reverse()  .immutableSortedCopy(allTerms);  System.out.println("All Terms count: " + top.size());  List<TermInDoc> topSmall = limitList(top, 0.3);  System.out.println("Small terms count: " + topSmall.size());  repoTermInDoc.save(topSmall);  }  private static List<TermInDoc> limitList(List<TermInDoc> list, double factor){  final Set<String> uniqueTerms = Sets.newHashSet();  List<TermInDoc> uniqueList = FluentIterable.from(list)  .filter(new Predicate<TermInDoc>() {  @Override  public boolean apply(@Nullable TermInDoc termInDoc) {  if(termInDoc == null) return false;  if (!uniqueTerms.contains(termInDoc.getTermin().getText())) {  uniqueTerms.add(termInDoc.getTermin().getText());  return true;  }  return false;  }  }).toList();  return FluentIterable.from(uniqueList)  .limit((int) Math.round(list.size() \* factor))  .toList();  }  private static Map<String, Double> filterTerms(Map<String, Double> map){  return ImmutableMap.copyOf(Maps.filterEntries(map, new Predicate<Map.Entry<String, Double>>() {  @Override  public boolean apply(@Nullable Map.Entry<String, Double> entry) {  if (entry.getKey().length() <= 2) return false;  if (!entry.getKey().matches("^[А-ЯІа-яі\u0027-]+$")) return false;  return true;  }  }));  }  public static List<String> sortByValue(final Map m) {  List<String> keys = new ArrayList<String>();  keys.addAll(m.keySet());  Collections.sort(keys, new Comparator<String>() {  public int compare(String o1, String o2) {  Object v2 = m.get(o1);  Object v1 = m.get(o2);  if (v1 == null) {  return (v2 == null) ? 0 : 1;  } else if (v1 instanceof Comparable) {  return ((Comparable) v1).compareTo(v2);  } else {  return 0;  }  }  });  return keys;  }  private void indexPdfCollection(String collection){  System.out.println(collection + " collection documentary frequency builder application");  PdfDirectoryIndexer indexer = indexer("thesauri/"+collection+"/", collection);  indexer.indexDirectoryWithPdfs();  }  @Override  public void run(String... strings) throws Exception {  try {  lemmatizer = LemmatizerFactory.getPrebuilt("mlteast-uk");  } catch (IOException e) {  e.printStackTrace();  }  if(strings.length>0){  String collection = strings[0];  indexPdfCollection(collection);  extractCollectionTerms(collection);  extractCollectionRelations(collection);  } else {  System.out.println("Abort: please enter colleciton name");  }  }  } |

**Додаток Г (довідниковий)**

Лістинг підпрограми пошуку важливих термінів колекції. Файл *TopTfIdfInitialTerminologyNounExtractor.java*

|  |
| --- |
| package edu.naukma.reshet.core.algorithm;  @Component("noun")  public class TopTfIdfInitialTerminologyNounExtractor implements InitialTerminologyExtractor {  @Autowired  @Qualifier("local")  private DocumentaryFrequencyCrawler crawler;  @Autowired  private TerminRepository terminRepo;  private JLanguageTool langTool;  public TopTfIdfInitialTerminologyNounExtractor(){  try {  langTool = new JLanguageTool(new Ukrainian());  } catch (IOException e) {  e.printStackTrace();  }  }  @Override  public List<TermInDoc> extractValuableTerms(Searcher searcher, Integer docId) {  List<TermInDoc> terms = Lists.newArrayList();  Map<String, Integer> map = filterTerms(searcher.getFrequencies(docId));  for(String term: map.keySet()){  try {  List<AnalyzedSentence> analyzed = langTool.analyzeText(term);  if(analyzed.isEmpty()) continue;  AnalyzedSentence sentence = analyzed.get(0);  boolean hasNoun = sentence.getTokensWithoutWhitespace()[1].hasPartialPosTag("noun");  if (hasNoun) {  Integer frequency = map.get(term);  Double docFreq = crawler.getDocumentaryFrequency(term);  Double totalFreq = 1.0\*frequency\*Math.log(1.0/docFreq);  Termin termin = terminRepo.findByText(term);  if(termin == null){  terminRepo.save(new Termin(term, 1D));  termin = terminRepo.findByText(term);  }  terms.add(new TermInDoc(termin,totalFreq, searcher.getIndexName()));  }  } catch (IOException e) {  e.printStackTrace();  }  }  return FluentIterable.from(  Ordering  .natural()  .reverse()  .immutableSortedCopy(terms)  ).limit((int)Math.round(terms.size()\*0.2)).toList();  }  private static Map<String, Integer> filterTerms(Map<String, Integer> map){  return ImmutableMap.copyOf(Maps.filterEntries(map, new Predicate<Map.Entry<String, Integer>>() {  @Override  public boolean apply(@Nullable Map.Entry<String, Integer> entry) {  if (entry.getKey().length() <= 2) return false;  if (!entry.getKey().matches("^[А-ЯІа-яі\u0027-]+$")) return false;  return true;  }  }));  }  public void setCrawler(DocumentaryFrequencyCrawler crawler) {  this.crawler = crawler;  }  public void setTerminRepo(TerminRepository terminRepo) {  this.terminRepo = terminRepo;  }  } |

**Додаток Г (довідниковий)**

Лістинг підпрограми пошуку зв’язків між термінами колекції. Файл *RelationFinder.java*

|  |
| --- |
| package edu.naukma.reshet.core.algorithm;  @Component  public class RelationFinder {  private Lemmatizer lm;  private BreakIterator iterator = BreakIterator.getSentenceInstance(new Locale("uk-UA"));  public final static MatchRule [] MATCH\_RULES\_1\_9 = {  new MatchRule(1, POSTag.ADJ, POSTag.NOUN, POSTag.NOUN, POSTag.NOUN),  new MatchRule(1, POSTag.ADJ, POSTag.NOUN, POSTag.ADJ, POSTag.NOUN),  new MatchRule(2, POSTag.ADJ, POSTag.NOUN, POSTag.NOUN),  new MatchRule(2, POSTag.ADJ, POSTag.ADJ, POSTag.NOUN),  new MatchRule(0, POSTag.NOUN, POSTag.ADJ, POSTag.NOUN),  new MatchRule(1, POSTag.ADJ, POSTag.NOUN),  new MatchRule(1, POSTag.NOUN, POSTag.NOUN),  new MatchRule(0, POSTag.NOUN)  };  public final static MatchRule [] MATCH\_RULES\_1\_4 = {  new MatchRule(0, POSTag.NOUN, POSTag.ADJ, POSTag.NOUN),  new MatchRule(1, POSTag.ADJ, POSTag.NOUN),  new MatchRule(1, POSTag.NOUN, POSTag.NOUN),  new MatchRule(0, POSTag.NOUN)  };  public final static MatchPattern LP1 = new MatchPattern(  new NounPhraseElement(true, MATCH\_RULES\_1\_9),  new ExactWordElement("—","це","є","вважається","слід розуміти","являється"),  new NounPhraseElement(false, MATCH\_RULES\_1\_9)  );  public final static MatchPattern LP2\_1 = new MatchPattern(  new ExactWordElement("такий"),  new NounPhraseElement(true, MATCH\_RULES\_1\_4),  new ExactWordElement(","),new ExactWordElement("як"),  new NounPhraseElement(false, MATCH\_RULES\_1\_4),  new ExactWordElement("і","або","й","та"),  new NounPhraseElement(false, MATCH\_RULES\_1\_4)  );  public final static MatchPattern LP2\_2 = new MatchPattern(  new ExactWordElement("такий"),  new NounPhraseElement(true, MATCH\_RULES\_1\_4),  new ExactWordElement(","),new ExactWordElement("як"),  new NounPhraseElement(false, MATCH\_RULES\_1\_4), new ExactWordElement(","),  new NounPhraseElement(false, MATCH\_RULES\_1\_4),  new ExactWordElement("і","або","й","та"),  new NounPhraseElement(false, MATCH\_RULES\_1\_4)  );  public final static MatchPattern LP2\_3 = new MatchPattern(  new ExactWordElement("такий"),  new NounPhraseElement(true, MATCH\_RULES\_1\_4),  new ExactWordElement(","),new ExactWordElement("як"),  new NounPhraseElement(false, MATCH\_RULES\_1\_4), new ExactWordElement(","),  new NounPhraseElement(false, MATCH\_RULES\_1\_4), new ExactWordElement(","),  new NounPhraseElement(false, MATCH\_RULES\_1\_4),  new ExactWordElement("і","або","й","та"),  new NounPhraseElement(false, MATCH\_RULES\_1\_4)  );  public final static MatchPattern LP2\_4 = new MatchPattern(  new ExactWordElement("такий"),  new NounPhraseElement(true, MATCH\_RULES\_1\_4),  new ExactWordElement(","),new ExactWordElement("як"),  new NounPhraseElement(false, MATCH\_RULES\_1\_4), new ExactWordElement(","),  new NounPhraseElement(false, MATCH\_RULES\_1\_4), new ExactWordElement(","),  new NounPhraseElement(false, MATCH\_RULES\_1\_4), new ExactWordElement(","),  new NounPhraseElement(false, MATCH\_RULES\_1\_4),  new ExactWordElement("і","або","й","та"),  new NounPhraseElement(false, MATCH\_RULES\_1\_4)  );  public final static MatchPattern LP3\_1 = new MatchPattern(  new NounPhraseElement(false, MATCH\_RULES\_1\_4),  new ExactWordElement(","), new NounPhraseElement(false, MATCH\_RULES\_1\_4),  new ExactWordElement("і","або","й","та"),new ExactWordElement("інший"),  new NounPhraseElement(true, MATCH\_RULES\_1\_4)  );  public final static MatchPattern LP3\_2 = new MatchPattern(  new NounPhraseElement(false, MATCH\_RULES\_1\_4),  new ExactWordElement(","), new NounPhraseElement(false, MATCH\_RULES\_1\_4),  new ExactWordElement(","), new NounPhraseElement(false, MATCH\_RULES\_1\_4),  new ExactWordElement("і","або","й","та"),new ExactWordElement("інший"),  new NounPhraseElement(true, MATCH\_RULES\_1\_4)  );  public final static MatchPattern LP3\_3 = new MatchPattern(  new NounPhraseElement(false, MATCH\_RULES\_1\_4),  new ExactWordElement(","), new NounPhraseElement(false, MATCH\_RULES\_1\_4),  new ExactWordElement(","), new NounPhraseElement(false, MATCH\_RULES\_1\_4),  new ExactWordElement(","), new NounPhraseElement(false, MATCH\_RULES\_1\_4),  new ExactWordElement("і","або","й","та"),new ExactWordElement("інший"),  new NounPhraseElement(true, MATCH\_RULES\_1\_4)  );  public final static MatchPattern LP3\_4 = new MatchPattern(  new NounPhraseElement(false, MATCH\_RULES\_1\_4),  new ExactWordElement(","), new NounPhraseElement(false, MATCH\_RULES\_1\_4),  new ExactWordElement(","), new NounPhraseElement(false, MATCH\_RULES\_1\_4),  new ExactWordElement(","), new NounPhraseElement(false, MATCH\_RULES\_1\_4),  new ExactWordElement(","), new NounPhraseElement(false, MATCH\_RULES\_1\_4),  new ExactWordElement("і","або","й","та"),new ExactWordElement("інший"),  new NounPhraseElement(true, MATCH\_RULES\_1\_4)  );  public final static MatchPattern LP4 = new MatchPattern(  new NounPhraseElement(true, MATCH\_RULES\_1\_9),  new ExactWordElement(","),new ExactWordElement("включаючи","зокрема","особливо"),  new IterationElement(new NounPhraseElement(false, MATCH\_RULES\_1\_9), new ExactWordElement(",")),  new ExactWordElement("і","або","й","та"),  new NounPhraseElement(false, MATCH\_RULES\_1\_9)  );  public final static MatchPattern LP4\_1 = new MatchPattern(  new NounPhraseElement(true, MATCH\_RULES\_1\_9),  new ExactWordElement(","),new ExactWordElement("включаючи","зокрема","особливо"),  new NounPhraseElement(false, MATCH\_RULES\_1\_9),  new ExactWordElement("і","або","й","та"),  new NounPhraseElement(false, MATCH\_RULES\_1\_9)  );  public final static MatchPattern LP4\_2 = new MatchPattern(  new NounPhraseElement(true, MATCH\_RULES\_1\_9),  new ExactWordElement(","),new ExactWordElement("включаючи","зокрема","особливо"),  new NounPhraseElement(false, MATCH\_RULES\_1\_9), new ExactWordElement(","),  new NounPhraseElement(false, MATCH\_RULES\_1\_9),  new ExactWordElement("і","або","й","та"),  new NounPhraseElement(false, MATCH\_RULES\_1\_9)  );  public final static MatchPattern LP4\_3 = new MatchPattern(  new NounPhraseElement(true, MATCH\_RULES\_1\_9),  new ExactWordElement(","),new ExactWordElement("включаючи","зокрема","особливо"),  new NounPhraseElement(false, MATCH\_RULES\_1\_9), new ExactWordElement(","),  new NounPhraseElement(false, MATCH\_RULES\_1\_9), new ExactWordElement(","),  new NounPhraseElement(false, MATCH\_RULES\_1\_9),  new ExactWordElement("і","або","й","та"),  new NounPhraseElement(false, MATCH\_RULES\_1\_9)  );  public final static MatchPattern LP4\_4 = new MatchPattern(  new NounPhraseElement(true, MATCH\_RULES\_1\_9),  new ExactWordElement(","),new ExactWordElement("включаючи","зокрема","особливо"),  new NounPhraseElement(false, MATCH\_RULES\_1\_9), new ExactWordElement(","),  new NounPhraseElement(false, MATCH\_RULES\_1\_9), new ExactWordElement(","),  new NounPhraseElement(false, MATCH\_RULES\_1\_9), new ExactWordElement(","),  new NounPhraseElement(false, MATCH\_RULES\_1\_9),  new ExactWordElement("і","або","й","та"),  new NounPhraseElement(false, MATCH\_RULES\_1\_9)  );  public final static MatchPattern P1\_1 = new MatchPattern(  new NounPhraseElement(true, MATCH\_RULES\_1\_9),  new ExactWordElement(","),new ExactWordElement("а"),new ExactWordElement("саме"), new ExactWordElement(":","—"),  new NounPhraseElement(false, MATCH\_RULES\_1\_9), new ExactWordElement(","),  new NounPhraseElement(false, MATCH\_RULES\_1\_9)  );  public final static MatchPattern P1\_1a = new MatchPattern(  new NounPhraseElement(true, MATCH\_RULES\_1\_9),  new ExactWordElement(","),new ExactWordElement("а"),new ExactWordElement("саме"), new ExactWordElement(":","—"),  new NounPhraseElement(false, MATCH\_RULES\_1\_9), new ExactWordElement("і","або","й","та"),  new NounPhraseElement(false, MATCH\_RULES\_1\_9)  );  public final static MatchPattern P1\_2 = new MatchPattern(  new NounPhraseElement(true, MATCH\_RULES\_1\_9),  new ExactWordElement(","),new ExactWordElement("а"),new ExactWordElement("саме"), new ExactWordElement(":","—"),  new NounPhraseElement(false, MATCH\_RULES\_1\_9), new ExactWordElement(","),  new NounPhraseElement(false, MATCH\_RULES\_1\_9)  );  public final static MatchPattern P1\_3 = new MatchPattern(  new NounPhraseElement(true, MATCH\_RULES\_1\_9),  new ExactWordElement(","),new ExactWordElement("а"),new ExactWordElement("саме"), new ExactWordElement(":","—"),  new NounPhraseElement(false, MATCH\_RULES\_1\_9), new ExactWordElement(","),  new NounPhraseElement(false, MATCH\_RULES\_1\_9),  new ExactWordElement("і","або","й","та"),  new NounPhraseElement(false, MATCH\_RULES\_1\_9)  );  public final static MatchPattern S0 = new MatchPattern(  new NounPhraseElement(true, MATCH\_RULES\_1\_9),  new ExactWordElement(","),new ExactWordElement("наприклад","як-от"),  new NounPhraseElement(false, MATCH\_RULES\_1\_9)  );  public final static MatchPattern S1 = new MatchPattern(  new NounPhraseElement(true, MATCH\_RULES\_1\_9),  new ExactWordElement(","),new ExactWordElement("наприклад","як-от"),  new IterationElement(new NounPhraseElement(false, MATCH\_RULES\_1\_9), new ExactWordElement(",")),  new NounPhraseElement(false, MATCH\_RULES\_1\_9)  );  public final static MatchPattern S2 = new MatchPattern(  new NounPhraseElement(true, MATCH\_RULES\_1\_9),  new ExactWordElement(","),new ExactWordElement("наприклад","як-от"),  new IterationElement(new NounPhraseElement(false, MATCH\_RULES\_1\_9), new ExactWordElement(",")),  new NounPhraseElement(false, MATCH\_RULES\_1\_9),  new ExactWordElement("і","або","й","та"),  new NounPhraseElement(false, MATCH\_RULES\_1\_9)  );  private final List<MatchPattern> isAPatterns = new ImmutableList.Builder<MatchPattern>()  .add(P1\_1,P1\_1a,P1\_2,P1\_3)  .add(LP1)  .add(LP2\_4,LP2\_3,LP2\_2,LP2\_1)  .add(LP3\_4,LP3\_3,LP3\_2,LP3\_1)  .add(LP4\_4,LP4\_3,LP4\_2,LP4\_1)  .add(S0,S1,S2)  .build();  public Set<TermRelation> getRelations(List<TermInDoc> terms, List<Snippet> snippets){  Set<TermRelation> termRelations = Sets.newHashSet();  this.addIsA(termRelations, terms, snippets);  return termRelations;  }  private void addAssociations(Set<TermRelation> relations, List<TermInDoc> terms, List<Snippet> snippets){  Function<TermInDoc,String> termToString = new Function<TermInDoc, String>() {  @Nullable  @Override  public String apply(@Nullable TermInDoc input) {  return input.getTermin().getText();  }  };  for(Snippet snip:snippets){  List<TermInDoc> relatedTerms = getTermsListFromSentence(snip.getText(), terms);  String str = Joiner.on(",").join(Lists.transform(relatedTerms,termToString));  System.out.println(str);  for(TermInDoc term:relatedTerms){  if(!term.getTermin().getText().equals(snip.getTerm().getTermin().getText())){  relations.add(new TermRelation(snip.getTerm(),term,"association", "science"));  relations.add(new TermRelation(term,snip.getTerm(),"association", "science"));  }  }  }  }  private void addIsA(Set<TermRelation> relations, List<TermInDoc> terms, List<Snippet> snippets){  System.out.println("Snippets to process: " + snippets.size());  int snipCounter = 0;  Set<Integer> processedSentences = Sets.newHashSet();  for(Snippet snip:snippets){  iterator.setText(snip.getText());  int start = iterator.first();  for (int end = iterator.next();  end != BreakIterator.DONE;  start = end, end = iterator.next()) {  String sentence = snip.getText().substring(start, end);  if(!processedSentences.contains(sentence.hashCode())){  relations.addAll(getRelationsFromSentence(sentence, terms));  processedSentences.add(sentence.hashCode());  }  }  snipCounter++;  if(snipCounter % 50 == 0){  System.out.println("Snippets processed: " + snipCounter + "/" + snippets.size());  }  }  }  private List<TermRelation> getRelationsFromSentence(String sentence, List<TermInDoc> terms){  List<TermRelation> relations = Lists.newLinkedList();  for(MatchPattern pattern: isAPatterns){  List<List<NounPhraseMatch>> matches = pattern.matchAll(sentence);  for(List<NounPhraseMatch> match: matches){  relations.addAll(pattern.getRelations(match));  }  }  if(!relations.isEmpty()){  System.out.println(relations);  }  return relations;  }  private List<TermInDoc> getTermsListFromSentence(String sentence, List<TermInDoc> allTerms){  List<TermInDoc> terms = Lists.newArrayList();  String [] words = sentence.split("(?=[,.])|\\s+");  for (String word : words) {  if (word.trim().length() > 1) {  final String lemma = lm.lemmatize(word.trim()).toString().toLowerCase();  Optional<TermInDoc> foundTerm = Optional.fromNullable(Iterables.find(allTerms, new Predicate<TermInDoc>() {  @Override  public boolean apply(@Nullable TermInDoc termInDoc) {  return termInDoc.getTermin().getText().equals(lemma);  }  }, null));  if(foundTerm.isPresent()){  terms.add(foundTerm.get());  }  }  }  return terms;  }  public void setLm(Lemmatizer lm) {  this.lm = lm;  }  } |

**Додаток Д (довідниковий)**

Лістинг підпрограми реалізації програмного інтерфейсу до серіалізації тезаурусу. Файл *RDFController.java*

|  |
| --- |
| package edu.naukma.reshet.resource;  @Controller  @RequestMapping("/rdf")  public class RDFController {  @Autowired  TermInDocRepository repoTerms;  @Autowired  TermRelationRepository repoRelations;  @RequestMapping(value = "/{name}", method = RequestMethod.GET)  @ResponseBody  public HttpEntity<RdfGraph> getRDF(@PathVariable String name){  List<TermInDoc> terms = FluentIterable  .from(repoTerms.findByIndex(name))  .toList();  List<TermRelation> relations = FluentIterable  .from(repoRelations.findAll(new PageRequest(0,200)).getContent())  .toList();  RdfGraph graph = new RdfGraph(terms,relations, Thesaurus.Type.BASIC);  return new ResponseEntity<RdfGraph>(graph, HttpStatus.OK);  }  @RequestMapping(value = "/{name}/compact", method = RequestMethod.GET)  @ResponseBody  public HttpEntity<RdfGraph> getRDFcompact(@PathVariable String name){  List<TermInDoc> terms = FluentIterable  .from(repoTerms.findByIndex(name))  .toList();  List<TermRelation> relations = FluentIterable  .from(repoRelations.findByIndex(name))  .toList();  RdfGraph graph = new RdfGraph(terms,relations, Thesaurus.Type.COMPACT);  return new ResponseEntity<RdfGraph>(graph, HttpStatus.OK);  }  @RequestMapping(value = "/{name}/human", method = RequestMethod.GET)  @ResponseBody  public HttpEntity<RdfGraph> getRDFhuman(@PathVariable String name){  List<TermInDoc> terms = FluentIterable  .from(repoTerms.findByIndex(name))  .toList();  List<TermRelation> relations = FluentIterable  .from(repoRelations.findByIndex(name))  .toList();  RdfGraph graph = new RdfGraph(terms,relations, Thesaurus.Type.HUMAN);  return new ResponseEntity<RdfGraph>(graph, HttpStatus.OK);  }  } |