МАГІСТЕРСЬКА РОБОТА

МР. ІПм - 54.00.00.000 ПЗ

Група ІПм-23-2

Остап’юк Олег

2024

Івано-Франківський національний технічний університет нафти і газу

Інститут інформаційних технологій

Кафедра інженерії програмного забезпечення

Остап'юк Олег Віталійович

(прізвище, ім'я, по батькові)

УДК 004.942

(індекс)

МАГІСТЕРСЬКА РОБОТА

Методи та засоби тематичного моделювання функцій локації

(назва роботи)

Інженерія програмного забезпечення

(назва освітньої програми)

121 - Інженерія програмного забезпечення

(шифр і назва спеціальності)

Остап'юк О.В.

(підпис, ініціали та прізвище здобувача освітнього ступеня)

Науковий керівник Крихівський Михайло Васильович, к.т.н., доцент

(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

Допущено до захисту

Завідувач кафедри

доц. Бандура В.В.

(посада) (підпис) (дата) (ініціали та прізвище)

Нормоконтроль

доц. Вовк Р.Б.

(посада) (підпис) (дата) (ініціали та прізвище)

Робота містить результати власних досліджень. Використання ідей, результатів і текстів інших авторів мають посилання на відповідне джерело

Івано-Франківськ – 2024

Івано-Франківський національний технічний університет нафти і газу

Інститут інформаційних технологій

Кафедра інженерії програмного забезпечення

Освітній рівень магістр

Спеціальність 121 – Інженерія програмного забезпечення

ЗАТВЕРДЖУЮ:
Зав. кафедрою **ІПЗ**

доц. В.В. Бандура

“ 04 ” вересня 2024 р.

ЗАВДАННЯ

НА МАГІСТЕРСЬКУ РОБОТУ СТУДЕНТУ

Остап’юку Олегу Віталійовичу

(прізвище, ім’я, по-батькові)

1. Тема магістерської роботи “ Методи та засоби тематичного моделювання функцій локації”

керівник проекту (роботи) Крихівський Михайло Васильович, к.т.н., доцент

затверджені наказом закладу вищої освіти від “ 22 ” листопада 2024 р. № 781 /7

2. Строк подання студентом проекту (роботи) 15 грудня 2024 р.

3. Вихідні дані до проекту (роботи) Теоретичні концепції та формальні моделі побудови та функціонування інформаційних та програмних технологій моделювання

4. Зміст розрахунково - пояснювальної записки(перелік питань, які потрібно розробити)

1. Дослідження предметної області проведення тематичного моделювання як методу NLP
2. Представлення моделей, методів та методології тематичного моделювання
3. Підготовка даних для тематичного моделювання локаційного визначення функції
4. Імплементація методів та засобів тематичного моделювання функцій локації

5. Перелік графічного матеріалу (з точним зазначенням обов’язкових креслень)

1. Архітектура RIPPLES (рис. 1.1)
2. Екранний граф RIPPLES для визначення місця розташування (рис. 1.2)
3. Графічний інтерфейс системи Grep (рис. 1.3)
4. Ілюстрація тематичного моделювання, застосованого до статті за допомогою LDA (рис. 1.4)
5. Графіки щільності [синій = низький, червоний = високий] (рис. 2.1)

6. Консультанти розділів проекту (роботи)

Розділ	Консультант	Підпис, дата
Перевірка на плагіат	доц., к.т.н. Вовк Р.Б.	

7. Дата видачі завдання 04 вересня 2024 р.

Керівник _____
(підпис)

Завдання прийняв до виконання_____
(підпис)

КАЛЕНДАРНИЙ ПЛАН

№ п/п	Назви етапів магістерської роботи	Строк виконання етапів роботи	Примітка
1	Підбір і вивчення літератури по темі магістерської роботи	15.09.2024	виконано
2	Аналіз концепцій та алгоритмів предметної області	29.09.2024	виконано
3	Дослідження предметної області проведення тематичного моделювання як методу NLP	15.10.2024	виконано
4	Представлення моделей, методів та методології тематичного моделювання	08.11.2024	виконано
5	Підготовка даних для тематичного моделювання локаційного визначення функції	20.11.2024	виконано
6	Імплементація методів та засобів тематичного моделювання функцій локації	01.12.2024	виконано
7	Затвердження пояснювальної записки роботи завідувачем кафедри	15.12.2024	виконано

Студент – магістр _____
(підпис)

Керівник роботи _____
(підпис)

АНОТАЦІЯ

Магістерська робота: 79 с., 21 рис., 11 табл., 52 джерела.

Тема: Методи та засоби тематичного моделювання функцій локації

Об'єкт дослідження: процеси обробки текстових даних з метою автоматизованого аналізу інформації.

Мета роботи: розробка та впровадження методів і засобів тематичного моделювання для визначення локаційного місця розташування функцій у текстових документах.

Предмет дослідження: методи, моделі та інструменти тематичного моделювання для визначення локаційного місця розташування функцій у текстових документах.

Результати дослідження

В роботі запропоновано методику інтеграції тематичного моделювання з сучасними інструментами управління задачами, зокрема модулем Jira.

Висновок

В магістерському дослідженні вдалося сформулювати теоретичну основу для тематичного моделювання функцій локації, запропонувати нові підходи до обробки текстових даних та підвищення продуктивності моделей NLP.

ТЕМАТИЧНЕ МОДЕЛЮВАННЯ, ОБРОБКА ПРИРОДНОЇ МОВИ, ЛОКАЦІЙНЕ РОЗТАШУВАННЯ ФУНКЦІЙ, ЛАТЕНТНИЙ РОЗПОДІЛ ДІРІХЛЕ (LDA), КЛАСИФІКАЦІЯ ТЕКСТІВ, МАШИННЕ НАВЧАННЯ, ВАЛІДАЦІЯ МОДЕЛЕЙ, УПРАВЛІННЯ ДАНИМИ.

ABSTRACT

Master Thesis: 79 pp., 21 fig., 11 tab., 52 sources.

Thesis Subject: Methods and tools for thematic modeling of location functions

Object of research: text data processing processes for the purpose of automated information analysis.

Purpose of work: development and implementation of methods and tools for thematic modeling to determine the location location of functions in text documents.

Subject of research: methods, models and tools for thematic modeling to determine the location location of functions in text documents.

Research results

The paper proposes a methodology for integrating thematic modeling with modern task management tools, in particular the Jira module.

Conclusion

The master's research managed to formulate a theoretical basis for thematic modeling of location functions, propose new approaches to text data processing and increase the productivity of NLP models.

TOPIC MODELING, NATURAL LANGUAGE PROCESSING, LOCAL FEATURE LOCATION, LATENT DIRICHLE DISTRIBUTION (LDA), TEXT CLASSIFICATION, MACHINE LEARNING, MODEL VALIDATION, DATA MANAGEMENT.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ СКОРОЧЕНЬ

FL - Feature Location

IR - Information Retrieval

ITS - Issue Tracking System

LDA - Latent Dirichlet Allocation

MRR - Mean Reciprocal Rank

NLP - Natural Language Processing

PA - Pachinko Allocation

RegEx - Regular Expression

VCS - Version Control System

ВСТУП

Актуальність теми.

У сучасному світі інформаційних технологій обсяги текстових даних стрімко зростають, що створює потребу у нових методах їх автоматизованого аналізу. Зокрема, завдання визначення локаційного місця розташування функцій у текстах є важливим для широкого спектра практичних застосувань: від управління проєктами та логістики до геоінформаційних систем та контекстного пошуку. Вирішення цієї проблеми дозволяє підвищити точність класифікації текстів, автоматизувати інформаційний пошук і прийняття рішень, що є критичним для великих корпоративних систем.

Методи тематичного моделювання, такі як Латентний розподіл Діріхле (LDA), зарекомендували себе як ефективні інструменти для аналізу великих текстових масивів, класифікації документів і виявлення прихованих структур у текстах. Однак застосування цих методів для визначення місця розташування функцій у текстових документах стикається з низкою викликів, зокрема:

- Потреба у високій точності тематичного розподілу;
- Адаптація моделей до специфічних наборів даних;
- Використання релевантних гіперпараметрів для оптимізації результатів.

Крім того, інтеграція тематичного моделювання з сучасними програмними платформами, такими як Jira, сприяє автоматизації управління процесами та підвищує ефективність обробки даних. Це є особливо актуальним у контексті цифрової трансформації, де важливість автоматизованого аналізу інформації постійно зростає.

Отже, розробка та вдосконалення методів тематичного моделювання для визначення локаційного місця розташування функцій є науково й практично значущою задачею, яка сприяє підвищенню ефективності

інформаційних систем, автоматизації бізнес-процесів та оптимізації роботи з текстовими даними.

Мета дослідження - розробка та впровадження методів і засобів тематичного моделювання для визначення локаційного місця розташування функцій у текстових документах.

Об'єкт дослідження - процеси обробки текстових даних з метою автоматизованого аналізу інформації.

Предмет дослідження - методи, моделі та інструменти тематичного моделювання для визначення локаційного місця розташування функцій у текстових документах.

Задачі дослідження:

- Провести аналіз предметної області тематичного моделювання, включаючи його концепції, методи та інструменти.

- Визначити особливості процесу видобування тексту та підготовки даних до тематичного моделювання.

- Розробити методику визначення локаційного місця розташування функцій на основі тематичного моделювання.

- Провести навчання та валідацію тематичних моделей, оцінити їх ефективність.

Методи дослідження

В роботі використано методи тематичного моделювання, зокрема Латентний розподіл Діріхле (LDA) та розподіл Пачінко, методи обробки природної мови (NLP) для попередньої обробки текстових даних, методи видобування тексту (Text Mining) для аналізу та підготовки текстових документів.

Наукова новизна отриманих результатів

Розроблено підхід до використання тематичного моделювання для визначення локаційного місця розташування функцій у текстах, який поєднує методи LDA, аналізу таксономій та обробки тексту та проведено детальний

аналіз впливу гіперпараметрів тематичних моделей на точність визначення місця розташування функцій.

Практичне значення результатів

Запропоновані методи та програмне рішення можуть бути використані для автоматизації аналізу текстових даних у великих обсягах та оптимізації пошуку інформації у корпоративних системах управління даними.

Структура магістерської роботи. Робота складається зі вступу, трьох розділів та висновків. Загальний обсяг роботи становить 79 сторінок, і містить 21 рисунок, 11 таблиць, список використаних джерел із 52 найменувань.

РОЗДІЛ 1. ДОСЛІДЖЕННЯ ПРЕДМЕТНОЇ ОБЛАСТІ ПРОВЕДЕННЯ ТЕМАТИЧНОГО МОДЕЛЮВАННЯ ЯК МЕТОДУ NLP

1.1. Опис проблеми дослідження

У розробці програмного забезпечення додавання нових функцій до системи не є завданням розробника програмного забезпечення, яке займає більшу частину часу та бюджету. Швидше від 60% до 80% бюджету витрачається на технічне обслуговування програмного забезпечення, яке є широкою діяльністю, включаючи роботу, яка виконується після того, як функція або вся програмна система починає працювати. Оскільки подальші дослідження показують, що близько 50% часу, витраченого на технічне обслуговування, витрачається на розуміння існуючого коду, інструменти, які підтримують цей процес, можуть мати значний вплив на продуктивність розробників. Зокрема, дослідження в цій галузі показали, що 88% ручних пошуків функції в незнайомому проекті не дають відповідних результатів.

Поки що ручний пошук у вихідному коді, підтримуваний такими інструментами, як grep і пошук текстового редактора, використовувався розробниками для цього завдання розуміння вихідного коду. У таких завданнях і коли потрібно знайти рішення проблеми обслуговування, ці інструменти вже дозволяють розробнику використовувати символи підстановки та канали для підключення пошуку на основі коду в кодовій базі.

Однак, крім цього, у поточних дослідженнях вивчаються нові методи, які інтегрують подальшу наявну інформацію в пул даних. Одним із відповідних полів є розташування функції як пошукова система для вихідного коду, оскільки вона може допомогти розробникам знайти відповідні частини у вихідному коді, просто ввівши пошуковий запит природною мовою в систему. Методи, які використовуються під капотом для забезпечення такої функціональності, можуть включати аналіз вихідного коду, пошук інформації або машинне навчання.

Для розробників, які не знайомі з кодовою базою, процес пошуку функцій, заснований виключно на текстурному пошуку у вихідному коді, показав низький рівень успіху. Тому передові методи визначення місця розташування історичних об'єктів раніше розглядалися як можливе рішення. Концепція була введена та обговорювалася в попередніх дослідженнях і спирається на тематичне моделювання, яке живиться даними з історичних даних кодової бази та описів, зібраних із систем відстеження проблем [1, 2].

Разом дані можуть формувати описи функцій, на основі яких тематичне моделювання може створювати кластери, що використовуються для зіставлення пошукових запитів із результатами пошуку. Проте немає готових до використання інструментів, які могли б зробити це надійно. Навіть ті прототипи, надані дослідниками, ще не генерують результатів, які могли б конкурувати з сучасними пошуковими системами, які популярні в певних контекстах без кодування. Крім того, цим прототипам поки що важко забезпечити відтворювані результати та контрольні показники, на основі яких можна розширити дослідження в цій галузі [3].

З цих причин необхідно обговорити питання про те, чи мають нові методи тематичного моделювання переваги над загальновживаним латентним розподілом Діріхле (LDA), і потрібно надати рішення для відтворюваних результатів і контрольних показників.

Мета цієї роботи та її дослідження базуються на даних, які описують еволюцію програмних систем. Такі дані включають історію вихідного коду, яку відстежують системи контролю версій, такі як Git, а також описи проблем із систем відстеження проблем, таких як Jira. У контексті цієї роботи такі дані будуть називатися даними історії кодової бази.

Робота побудована за двома цілями. У той час як перша ціль зосереджена на рішенні, яке може стати основою для порівняння між техніками тематичного моделювання в контексті розташування об'єктів, друга мета конкретно стосується порівняння двох вибраних технік, латентного розподілу Діріхле та розподілу Пачінко.

1) Розробка та реалізація набору інструментів, який підтримує застосування визначення розташування ознак до історичних даних бази коду. Частиною цієї мети є досягнення гнучкості через модульність для обміну конкретними технологіями, а також відтворюваність результатів за допомогою машиночитаних і людиночитаних результатів.

2) Порівняння розподілу Пачінко з розподілом Латентного Діріхле з точки зору продуктивності та точності, коли вони застосовуються в контексті розташування об’єктів. Ця мета містить вимогу дослідити вплив, який змінні конфігурації можуть мати на продуктивність і точність моделей. Крім того, частиною цієї мети є пряме порівняння між двома хорошими моделями з точки зору визначення місцезнаходження функцій в історичних даних кодової бази.

1.2. Поняття місця локаційного розташування функції на основі тематичного моделювання до документів

Обсяг цієї роботи вимагає застосування методів визначення місця розташування функцій на основі тематичного моделювання до документів, які описують ці функції в текстовій формі. Ці документи складаються з тексту природною мовою та змін, внесених до вихідного коду. Цей розділ розіб'є високорівневу ідею визначення місця розташування функцій на окремі кроки, які описують сучасний стан досліджень у цій галузі. Крім того, буде пояснено функцію цих кроків, а також їх мету в контексті роботи.

Починаючи з роз'яснення того, що таке визначення місця розташування функцій та як воно може бути використане розробниками та дослідниками для розуміння структури програмних проектів, будуть обговорені принципи, на яких воно базується. Вони включають пошук інформації, тематичне моделювання з розподілами Діріхле, а також видобуток та очищення тексту.

Визначення місця розташування функцій - це застосування обробки природної мови з конкретною метою визначення артефактів у текстових

документах, які відповідають пошуковому запиту. У цьому розділі буде коротко описано визначення місця розташування функцій у контексті визначення місця розташування функцій у вихідному коді, а також розглянуто доступні інструменти, які можуть виконувати це завдання. Крім того, представлено список ресурсів, які можуть допомогти під час оцінки результатів визначення місця розташування функцій [5].

1.2.1. Визначення і таксономія

Основною метою рефакторингу є видалення технічного боргу з кодової бази. Його можна визначити як іменник та як дієслово відповідно:

Рефакторинг (іменник): зміна, внесена до внутрішньої структури програмного забезпечення, щоб полегшити його розуміння та здешевити модифікацію без зміни його спостережуваної поведінки.

Рефакторинг (дієслово): реструктуризація програмного забезпечення шляхом застосування серії рефакторингів без зміни його спостережуваної поведінки.

Щоб мати можливість виконувати рефакторинг та працювати з вихідним кодом загалом, розробники повинні добре розуміти кодову базу. Визначення місця розташування функцій може бути першим кроком цього процесу, оскільки це дія з ідентифікації сутності або сутностей вихідного коду, які реалізують функцію [7]. Його можна виконати на основі чотирьох типів аналізу: динамічного, статичного, історичного та текстового [12]. Вільніше визначене, його можна описати як пошукову систему для функцій, де результатами є методи, класи або вихідні файли.

Згідно з дослідженням визначення місця розташування функцій у вихідному коді [13], динамічний аналіз проводиться під час виконання системи, спостерігаючи за функціями під час виконання програми. Дослідження показує, що їх можна класифікувати для збору трасування виконання програми, поки викликається функція, а потім або порівняти їх з трасуваннями, де функція не викликалась, або виконати аналіз на основі

частоти. Оскільки, за правильних сценаріїв, функції можна відобразити таким чином, дослідження показує, що вони є популярним вибором, який має лише той недолік, що спричиняє значні накладні витрати на час виконання.

Статичний аналіз описується в тому ж дослідженні як процеси, що досліджують структурну інформацію, таку як залежності в коді або потік даних у програмі. Хоча ця тактика може бути тісно пов'язана з тим, що робить розробник, коли він намагається зрозуміти код вручну, дослідження виявило високу ймовірність хибнопозитивних результатів, коли це автоматизовано.

Текстові підходи, досліджені дослідниками, ґрунтуються на ідеї агрегації ідентифікаторів та коментарів, які пов'язані з текстовими описами знань предметної області. Цей текст потім можна використовувати для пошуку функції за допомогою запиту в текстовій формі. Для полегшення реалізації такого аналізу дослідження визначило три методи: зіставлення зі зразком, пошук інформації та обробка природної мови, які будуть детальніше обговорені в наступному розділі.

Нарешті, метод визначення місця розташування функцій, описаний у дослідженні як історичний, спирається на видобуток даних із систем контролю версій і таким чином збирає відповідні рядки коду або артефакти, пов'язані з функцією.

Сучасні підходи в дослідженнях зазвичай поєднують принаймні два з цих методів, намагаючись усунути недоліки одного за допомогою переваг іншого [14]. Комбінація, на якій буде зосереджена ця дисертація, - це історичний аналіз, який також використовує текстові підходи до визначення місця розташування функцій за допомогою тематичного моделювання, яке, як обговорюється далі в розділі, є методом пошуку інформації.

1.2.2. Інструменти для визначення місця розташування функцій

Попередні дослідження вже створили деякі дослідні зразки та готові до виробництва інструменти для застосування визначення місця розташування

функцій до проектів. Цей розділ має на меті надати короткий огляд. Для двох найбільш релевантних методів, історичного та текстового аналізу, існують інструменти CVSSearch [14] та Hipikat [15], що дозволяють історичне визначення місця розташування функцій [13].

Інші інструменти для текстового визначення місця розташування функцій - це здебільшого інтеграція функцій текстового пошуку в інтегровані середовища розробки, такі як eclipse. Прикладами є Google Eclipse Search [12] та IRiSS [16]. Для інших, інструменти, такі як TraceGraph [18], STRADA [19] та Featureous [20], реалізують форми динамічного визначення місця розташування функцій, а Ripples [21] та Suade [22] реалізують методи статичного визначення місця розташування функцій.

Інструмент RIPPLES [21] витягує ASDG (Abstract Semantic Dependency Graph - абстрактний граф семантичної залежності) з коду C та підтримує як визначення місця розташування концепції, так і поширення змін. Архітектура RIPPLES показана на рисунку 1.1.

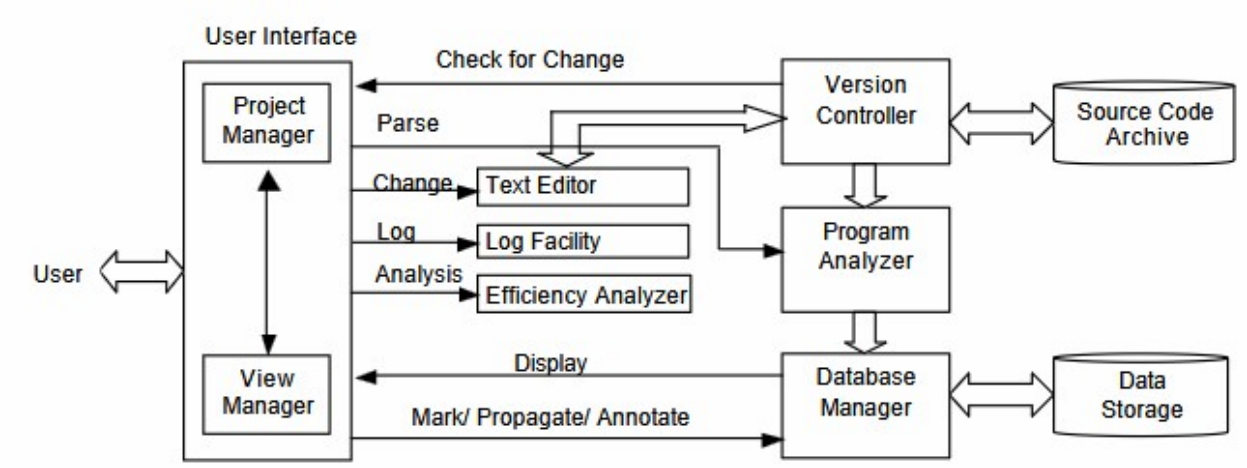


Рис. 1.1. Архітектура RIPPLES

RIPPLES реалізовано на Tcl/Tk. На рисунку 1.2 показано інтерфейс користувача RIPPLES.

Інструмент складається з таких основних компонентів:

- Контролер версій: керує архівом вихідного коду.

- Менеджер проектів: обробляє інформацію про проект, інформацію про файли, налаштування середовища тощо.

- Менеджер перегляду: керує масштабуванням, макетом та обробкою подій.

- Журнал: записує всі операції для інструментації, вимірювання ефективності та скасування.

- Аналізатор ефективності: вимірює ефективність RIPPLES.

- Аналізатор програм: аналізує код, будує ASDG та зберігає його в базі даних. Він використовує парсер Datrix C/C++.

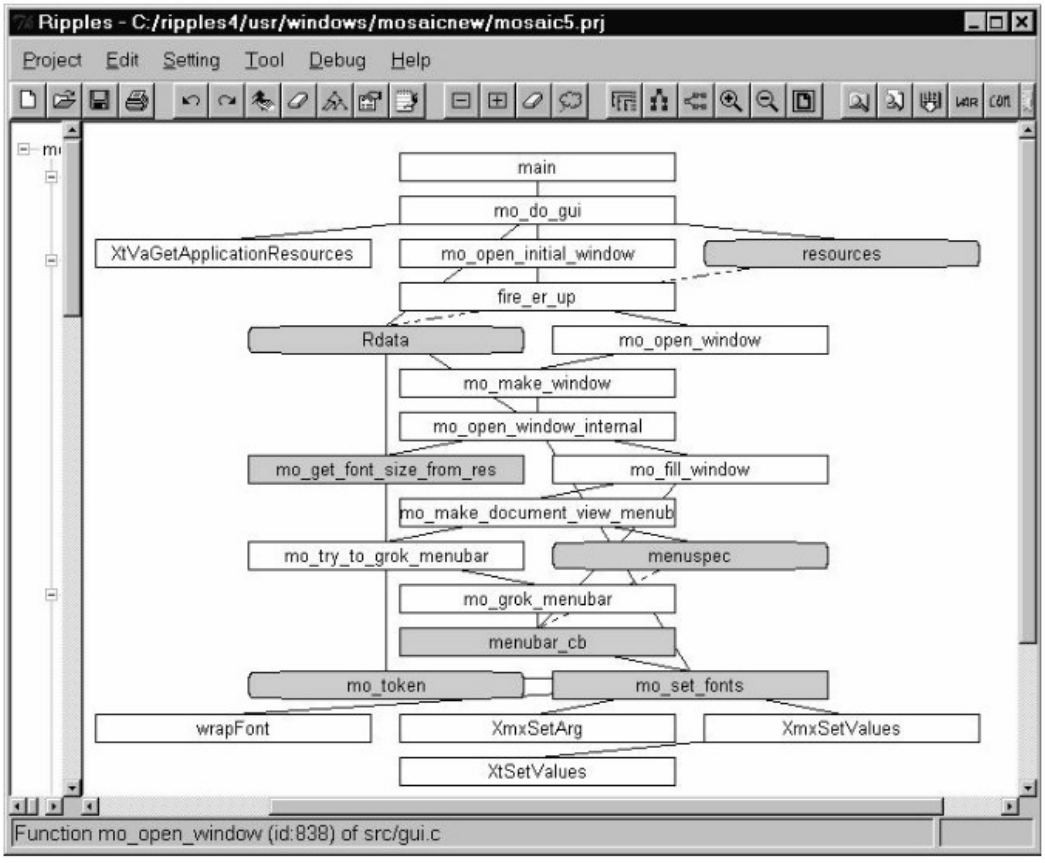


Рис. 1.2. Екранний граф RIPPLES для визначення місця розташування (виділені компоненти підсвічені)

1.2.3. Набори даних для порівняльного аналізу

Окрім інструментів, які певним чином дозволяють визначати місце розташування об’єктів, деякі дослідники також опублікували набори даних,

які мають допомогти у подальших дослідженнях, надаючи золотий набір зіставлень результатів запиту, які можна використовувати для оцінки результатів нових методів.

1.3. Процес видобування тексту (Text Mining)

Як зазначено в попередньому розділі, три методи сприяють текстовому аналізу для визначення місця розташування функцій: зіставлення зі зразком, пошук інформації та обробка природної мови [22]. Вони будуть представлені в цьому розділі.

В огляді технологій [23] автори описують зіставлення зі зразком як простий текстовий пошук за допомогою утиліт, таких як Grep. Він описується як відносно надійний, але не дуже точний, оскільки шанси на збіг термінів запиту зі словами, знайденими у вихідному коді, відносно низькі.

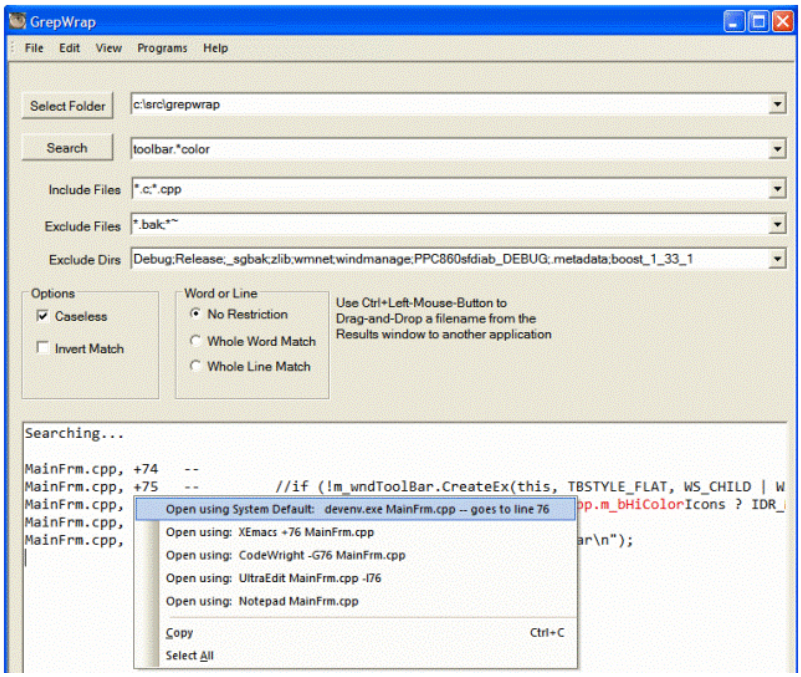


Рис. 1.3. Графічний інтерфейс системи Grep

До інших технологій, що зазвичай використовуються, належать регулярні вирази, які відповідають шаблонам, що складаються з одного або

кількох літералів символів, операторів або конструкцій [21]. Регулярні вирази будуть використовуватися як спосіб попередньої фільтрації даних під час видобутку тексту в цій роботії. Один із прикладів можна побачити в комбінації лістингу 1.1 та лістингу 1.2

Лістинг 1.1. Регулярний вираз, що наближено розпізнає імена методів Java.

```
(?:public|protected|private|static|\s) +[\w\<\>\[\]]+\s+(\w+) *\([^\)]*\) *(?:\{?|[^;])
```

Лістинг 1.2. Приклад зіставлення зі зразком за допомогою регулярного виразу для виявлення імені методу

```
/**
 * Add the specified scheme:auth information to this connection.
 */
public void addAuthInfo(String scheme, byte[] auth) {
            ↑ Method  ↑
    cnxn.addAuthInfo(scheme, auth);
}
```

Огляд [13] також описує методи пошуку інформації як статистичні методи, що використовуються для пошуку відповідного коду функцій на основі ідентифікаторів та коментарів, подібних до запиту. З точки зору технології, латентний розподіл Діріхле (LDA) [23] є серед тих, на які посилається огляд. Як форма тематичного моделювання, що базується на розподілах Діріхле, LDA буде оцінено поряд із спорідненим розподілом Пачінко [25] у цій роботі. Через свою актуальність у визначенні того, які частини вихідного коду відповідають запиту, розподіли Діріхле, а також LDA та розподіл Пачінко детальніше описані в наступних розділах, після аналізу загального огляду сучасного стану тематичного моделювання.

Обробка природної мови (NLP) - це широкий підхід, який можна використовувати для аналізу частин мови, що використовуються у вихідному

коді та пошуковому запиті, щоб зіставити їх точніше, ніж це можливо за допомогою зіставлення зі зразком, проте це також дорожче [33]. У цій дисертації будуть використані методи, пов'язані з NLP, такі як попередня фільтрація пошукових запитів шляхом токенізації речень на слова та видалення стоп-слів.

1.4 Концепція тематичного моделювання як методу пошуку інформації та обробки природної мови

Тематичне моделювання як метод пошуку інформації відіграватиме важливу роль у виконанні цього проекту. Тому в цьому розділі буде коротко описано сучасний стан тематичного моделювання.

Будучи методом навчання без учителя зі спеціалізацією на виявленні тем з текстових документів, тематичне моделювання має великий потенціал у сучасних інтернет-технологіях [37]. Його можна, наприклад, використовувати для класифікації оцифрованих документів в архіві, що в іншому випадку вимагало б ручного аналізу, щоб розподілити їх на групи для зручного перегляду. Для реалізації цього існує кілька типів моделей. Зосереджуючись на LDA та розподілі Пачінко, вони будуть пояснені в наступних розділах.

По суті, тематичне моделювання можна описати як ймовірнісне кластеризування для текстових документів. На нижчому рівні це означає, що модель може пов'язувати слова з подібним значенням у тематичні групи з однаковим контекстом і розрізняти використання слів з кількома значеннями [38]. Тому тема - це розподіл за фіксованим словником. У прикладі кластеризації статей про різні науки один кластер міститиме слова, які часто використовуються в генетиці з високою ймовірністю, тоді як інший може містити слова про еволюційну біологію [39]. Рисунок 1.4 графічно показує цей спосіб роботи зі статтями, проте слід підкреслити, що алгоритми LDA та розподілу Пачінко не мають інформації про теми тем, які вони створюють,

оскільки маркування не задіяне [32]. Отриману модель можна використовувати, наприклад, в інструменті для автоматичного групування документів або як пошукову систему для запитів користувачів. Останнє буде випадком використання, застосованим у цій роботі.

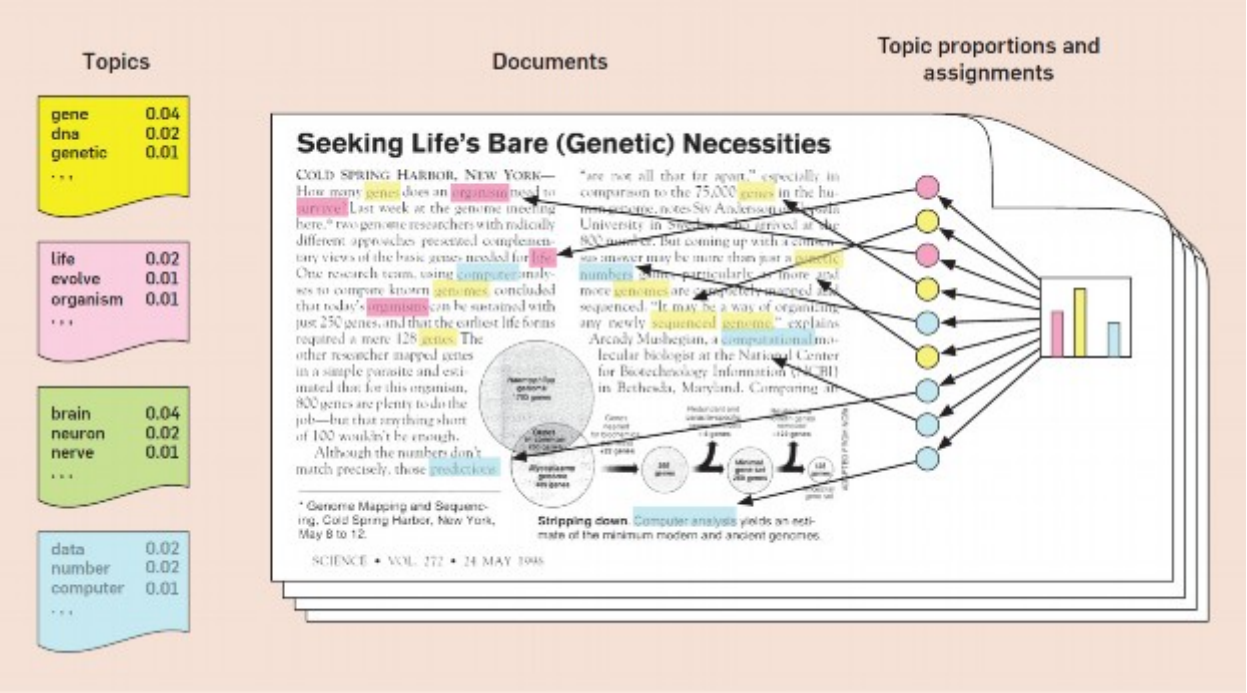


Рис. 1.4 Ілюстрація тематичного моделювання, застосованого до статті за допомогою LDA

Окрім вищезгаданої моделі розподілу Пачінко, дослідження породили багато альтернатив моделі LDA, яка є концептуальною основою для деяких з них [36]. LDA, в свою чергу, базується на розподілах Діріхле, обговорених у наступному розділі [37].

Навчання тематичних моделей, як правило, застосовується пакетно, додаючи кілька наборів даних до моделі, а потім завершуючи її, що унеможливлює додавання нових документів пізніше. Таким чином, це стосується і LDA, і моделі розподілу Пачінко, проте існує тенденція до застосування онлайн-тематичного моделювання. З цими онлайн-варіантами, заснованими на класичних моделях, документи можна додавати до корпусу, одночасно маючи придатну для використання модель [37]. Хоча це було б

корисним доповненням до інструменту, розробленого в цій роботі, дозволяючи йому надавати постійно оновлювану пошукову систему для функцій, реалізації таких тематичних моделей у бібліотеках або взагалі є рідкісними, а ті, що є загальнодоступними, обмежені LDA. Теорії для цих онлайн-варіантів обговорюються та порівнюються в таких статтях, як [39].

На відміну від кластеризації, методи тематичного моделювання не генерують чітких результатів, призначаючи документи одному кластеру (або темі), а багатьом. Це можна порівняти з нечіткою кластеризацією, підкатегорією звичайних алгоритмів кластеризації [Pr17]. Ця асоціація з кількома кластерами в тематичних моделях, таких як LDA та розподіл Пачінко, представлена у вигляді розподілів Діріхле, які будуть темою наступного розділу.

Висновки до розділу

У цьому розділі досліджуються основи тематичного моделювання як методу аналізу тексту в рамках обробки природної мови (NLP). Головна мета аналізу полягає у визначенні прихованих структур у текстах, які відображають ключові теми, що допомагають зрозуміти зміст документів та їх зв’язок із функціями розташування.

Описано проблему дослідження, що полягає в необхідності автоматизації роботи з великими обсягами текстових даних. Ручне опрацювання є неефективним через різноманітність і неструктурованість текстів, тому тематичне моделювання виступає незамінним інструментом для цієї задачі.

Тематичне моделювання особливо корисне для визначення функцій локації в документах. Розглянуто, як ключові слова й теми можуть бути використані для аналізу місця, контексту або функціонального призначення тексту. У цьому контексті важливим є визначення таксономії таких функцій,

яка включає різні типи текстових даних (корпуси новин, соціальні мережі, технічні документи).

Також у розділі описуються інструменти для аналізу, серед яких алгоритми на основі Latent Dirichlet Allocation (LDA), Biterm Topic Model (BTM), а також новітні методи на основі нейронних мереж. Для порівняння їхньої ефективності використовуються різноманітні набори даних, наприклад, твіти чи текстові повідомлення.

Процес видобування тексту (Text Mining) представлений як багатоетапний підхід, який включає очищення текстів, векторизацію та власне побудову моделей. Тематичне моделювання при цьому стає ключовим інструментом для пошуку та класифікації інформації, особливо в контексті обробки коротких текстів чи текстів із шумовими даними.

Нарешті, підкреслено значення тематичного моделювання в контексті інформаційного пошуку. Як метод NLP, воно відкриває широкі можливості для створення ефективних систем аналізу, здатних автоматично визначати теми, важливі для функцій локації, покращуючи розуміння текстів і їх класифікацію.

РОЗДІЛ 2. ПРЕДСТАВЛЕННЯ МОДЕЛЕЙ, МЕТОДІВ ТА МЕТОДОЛОГІЇ ТЕМАТИЧНОГО МОДЕЛЮВАННЯ

2.1. Розподіл Діріхле

Для глибокого розуміння розподілів Діріхле потрібні значні знання зі статистики. Для того, щоб слідувати аргументації та реалізації, представленим у цій дисертації, такі знання не потрібні, проте корисним буде загальне розуміння того, що описує розподіл Діріхле, і воно буде коротко викладено в цьому розділі. Для досягнення цієї мети пояснення з [21] будуть відображені на прикладі тригранного кубика.

Шестигранний кубик, на якому кожне число «1», «2» та «3» з'являється двічі, будемо називати тригранним кубиком, і його можна розглядати як функцію маси ймовірності, як і звичайний кубик [21]. Ці кубики, виготовлені вручну, не будуть рівномірно зважені, і тому матимуть випадковість результатів, яка не вважається справедливою. Цілком ймовірно, що буде виготовлено підтасований кубик, що призведе до несправедливого розподілу ймовірностей. Цю функцію маси ймовірності для тригранних кубиків можна записати як вектор у $\mathbb{R}3$, як показано нижче, де Q_{fair} символізує неможливий справедливий кубик, а $_{\text{Qloaded}}$ символізує несправедливий кубик, з яким значення «3» є найбільш ймовірним, а «1» - найменш ймовірним.

$$\theta_{fair} = \left(\frac{1}{3}, \frac{1}{3}, \frac{1}{3}\right)$$

$$\theta_{loaded} = \left(\frac{0.9}{3}, \frac{1}{3}, \frac{1.1}{3}\right)$$

Враховуючи мішок із 100 таких кубиків ручної роботи, випадковість функції маси ймовірності можна змоделювати за допомогою розподілу Діріхле. Даними передумовами є те, що результати кожної сторони Q = (Q1,

Q2, … , Q_k), у цьому випадку з k = 3, більше нуля і в сумі дорівнюють одиниці.

$$\theta_i \geq 0 \; for \; i = 1, 2, \ldots, k$$

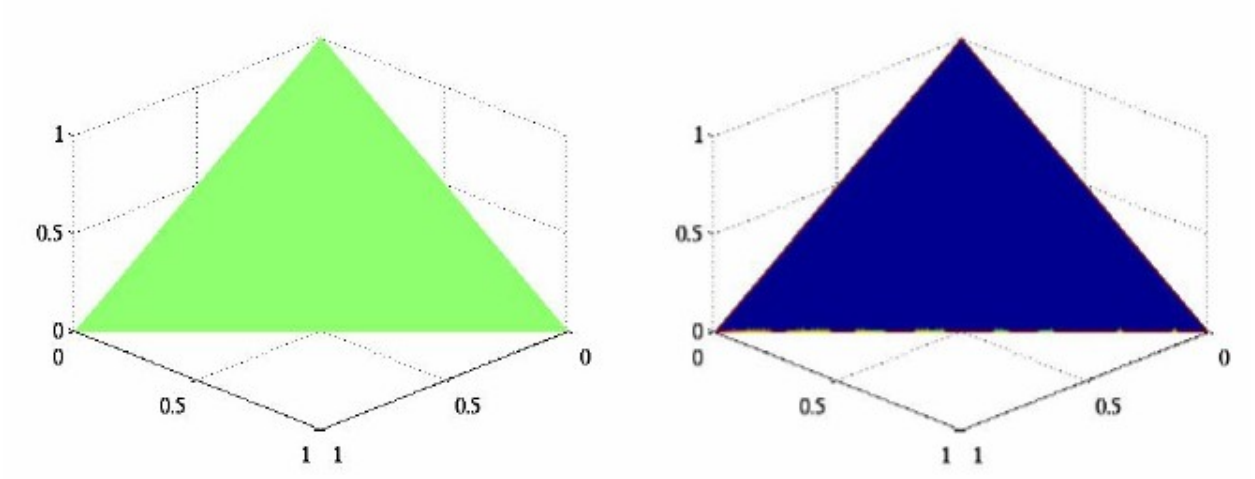
$$\sum_{i=1}^{k} \theta_i = 1$$

Також потрібно, щоб параметр α = (α_1, α_2, … , α_k) був визначений з α > 0 для i = 1, 2, … k.

При цьому кожну функцію маси ймовірності можна візуалізувати як точку на трикутнику в тривимірному евклідовому просторі. Тепер функцією розподілу Діріхле є забезпечення розподілу ймовірності по цьому простору. Для цього використовується наведена нижче функція:

$$p(\theta|\alpha) = \frac{\Gamma(\sum_{i=1}^{k} a_i)}{\prod_{i=1}^{k} \Gamma(a_i)} \prod_{i=1}^{K} \theta_i^{a_i - 1}$$

Поєднання їх потім призводить до графіків щільності, які мають різний центр щільності залежно від параметра α , який був використаний.



$\alpha = (1, 1, 1)$ $\alpha = (0.1, 0.1, 0.1)$

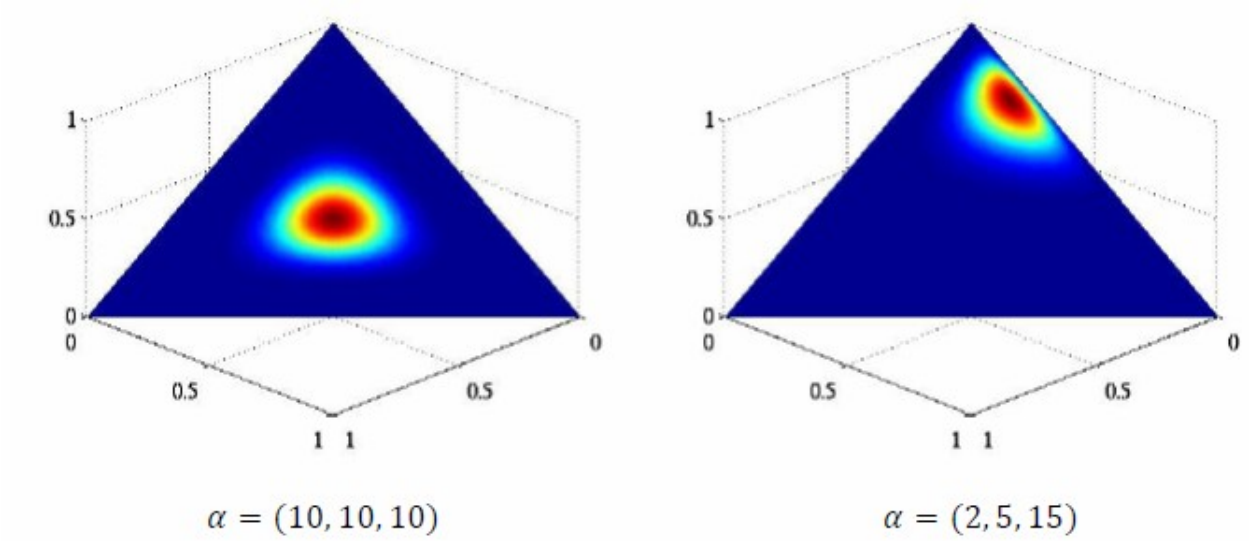


Рис. 2.1. Графіки щільності [синій = низький, червоний = високий]

Для рівномірного α = (c, c, c) з c > 1 графік щільності показаний на рисунку 2 у нижньому лівому куті: є центр щільності посередині трикутника. Інтерпретовано на прикладі тригранних кубиків це означає, що більшість результатів близька до Q_{fair} , що свідчить про передову та точну технологію виробництва. Для аналогічно рівномірного α з c = 1 застосовується особливий випадок у верхньому лівому куті рисунка 2.1. Тут усі результати однаково ймовірні, що призводить до появи всіх видів справедливих і несправедливих кубиків. Знову ж таки, рівномірний α з 0 < c < 1 створює кубики, які менш ймовірно будуть справедливими, ніж несправедливими, причому ступінь справедливості зменшується, коли *c* наближається до 0. Нарешті, для α, що є вектором, який не є рівномірним, центр переміщується до будь-якого кута трикутника, що означає, що виготовлений кубик буде завантажений у певному напрямку.

Цей розподіл Діріхле відіграватиме важливу роль як у налаштуванні латентного розподілу Діріхле, так і в розподілі Пачінко. Крім того, апріор буде актуальним як один з гіперпараметрів, який можна використовувати для впливу на точність моделей. Проте в сучасних дослідженнях це незвична

практика, коли майже всі дослідники обирають прості симетричні апріори Діріхле [44].

2.2. Латентний розподіл Діріхле та розподіл Пачінко

Як латентний розподіл Діріхле, так і розподіл Пачінко складаються з розподілів Діріхле. Оскільки обидва будуть активно використовуватися для створення тематичних моделей у цій дисертації, сучасні дослідження цих технологій будуть коротко викладені в цьому розділі.

Латентний розподіл Діріхле був вперше представлений у 2003 році і з тих пір використовується в багатьох дослідницьких роботах, які застосовували тематичне моделювання в усіх видах сценаріїв [39]. Деякі з них будуть згадані пізніше в цій дисертації через їх актуальність для їх зв'язку з темою.

Згідно з початковою статтею, список попередників LDA включає схему tf-idf, латентне семантичне індексування (LSI) та ймовірнісне латентне семантичне індексування як розширення LDA [44]. Причина, по якій LDA було введено на додаток до цих рішень, полягала в тому, щоб надати рішення, яке працює в рамках так званого «припущення про мішок слів», яке вважає порядок слів у статті несуттєвим [45].

tf-idf просто базується на підрахунку слів і термінів у документах, і з цього він створює матрицю термінів за документами, ефективно зводячи документи змінної довжини до списків чисел фіксованої довжини [14]. LSI та pLSI розширюють це, але мають власні проблеми, як обговорюється в [19].

Нарешті, LDA визначає приховані теми для захоплення прихованої семантики в текстових документах. З LDA кожен документ представлений розподілом за неспостережуваними (латентними) темами, причому кожна тема описується розподілом за словами [19, 25]. У цьому поясненні той факт, що теми не спостерігаються, означає, що вони не містять жодних позначок, які б їх описували, а скоріше те, що вони є набором пропорцій їхнього

вмісту. При цьому порядок слів не має значення, що додатково можна довести за допомогою так званої теореми представлення де Фінетті [46]. Візуальну абстракцію цього відношення можна побачити на рисунку 2.2.

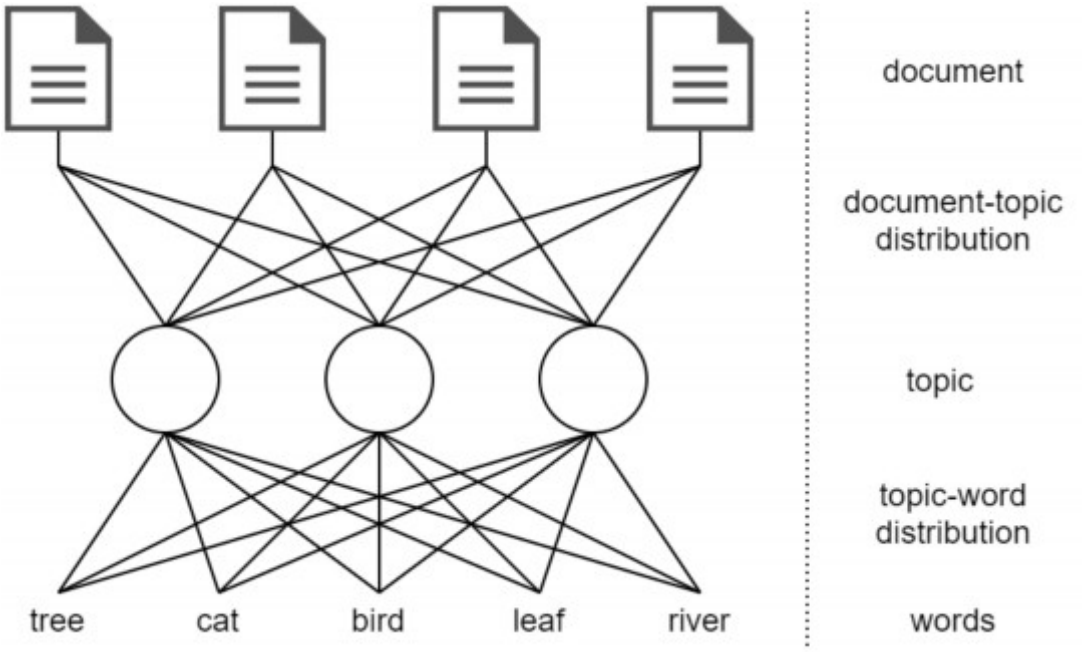


Рис. 2.2. Візуальна абстракція зв’язку між документами та словами в LDA

Представляючи свою нову модель, автори розподілу Пачінко порівнюють свою ідею з латентним розподілом Діріхле за допомогою візуалізації, показаної на рисунку 2.3, яка порівнюється з графіком на рисунку 2.2. Можна побачити обидва орієнтовані графіки LDA і розподілу Пачінко складається з вузла для вибірки документів (r), внутрішніх вузлів, що представляють теми, зроблені з розподілу Діріхле (S), і листків, що представляють слова (V) [48].

Різниця полягає в кількості внутрішніх шарів, яких у Pachinko Allocation два порівняно з одним. Це робиться для того, щоб розподіл Пачінко міг виявити кореляції між темами, для яких LDA не має способу ідентифікації [12]. Хоча можливо навіть більше рівнів, ця так звана чотирирівнева модель розподілу Пачінко була описана в початковій статті [12].

Таким чином, мотивацією для розподілу Пачінко було створення точних моделей, які можуть виявити велику кількість детальних тем і кореляції між цими темами.

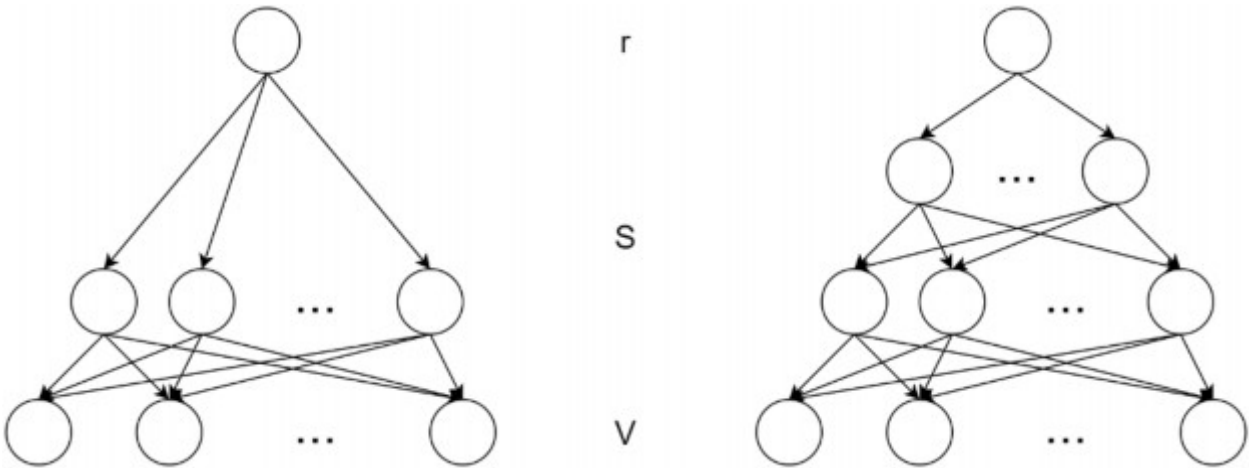


Рис. 2.3. Модельні структури LDA та чотирирівневого розподілу Пачінко

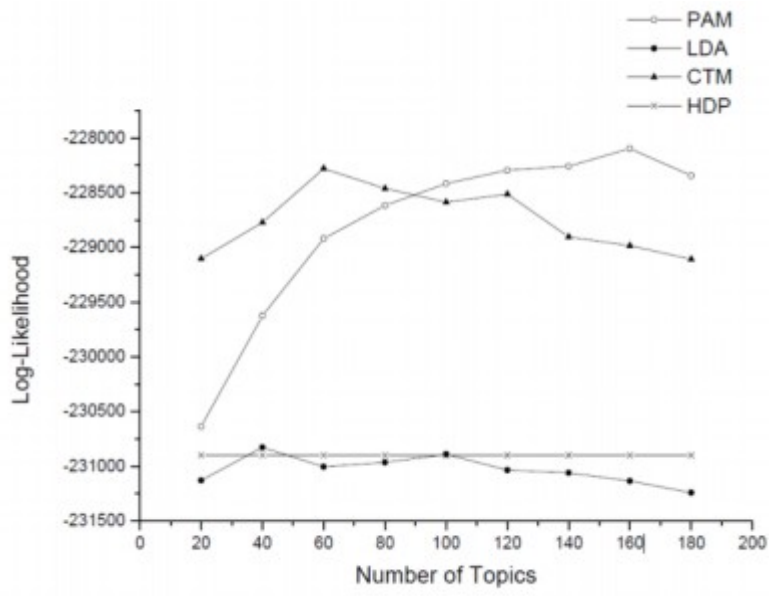
Документ, що визначає розподіл Пачінко, далі порівнює його з іншими моделями, включаючи порівняння точності класифікації з латентним розподілом Діріхле. Це наочно показано в таблиці 2.1. Результати показують, що розподіл Пачінко має незмінно вищу точність, ніж LDA.

Таблиця 2.1.

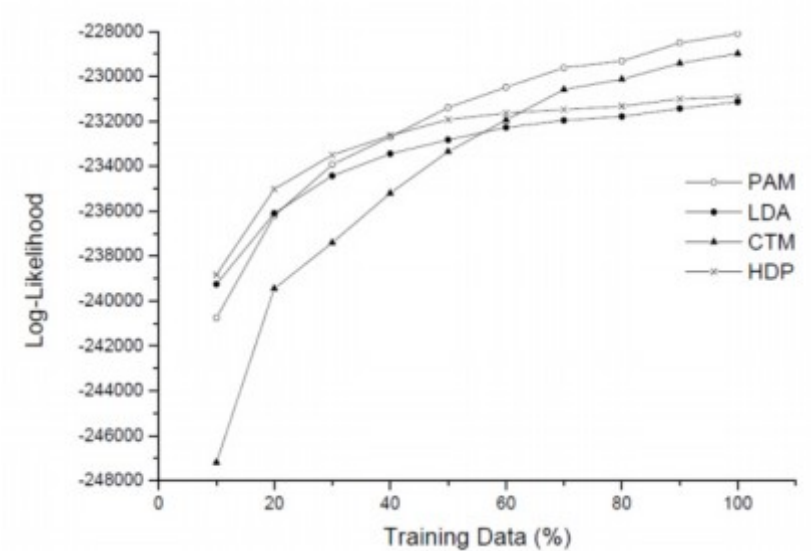
Точність класифікації документів

class	Number of docs	LDA	PAM
graphics	234	83.95	86.83
os	239	81.59	84.10
pc	245	83.67	88.16
mac	239	86.61	89.54
windows.x	243	88.07	92.20
total	1209	84.70	87.34

Тести, проведені в статті, також показують, що розподіл Пачінко може обробляти більшу кількість тем порівняно з LDA, зберігаючи при цьому вищу ймовірність. Те саме стосується більшого обсягу навчальних даних, що робить його альтернативою для розгляду під час роботи з тематичним моделюванням. Відповідні графіки показані на рисунку 2.4.



а) Pachinko працює краще з більшою кількістю тем



б) Пачінко працює краще з більшою кількістю тренувальних даних

Рис. 2.4. Порівняння між LDA, Pachinko Allocation (PAM) і двома іншими методами

2.3. Огляд методології дослідження

У цьому розділі буде детально описано методології, застосовані на етапі дослідження дисертації. Буде описано, за допомогою яких ідей було досягнуто рішення, як розроблені тести та як структурована оцінка результатів.

Основою для етапу дослідження є корпус, який використовується для навчання тематичних моделей. Цей процес на прикладі деталізації на рівні класу візуалізовано на рисунку 2.5. Він по суті виконує об'єднання інформації з двох джерел: історії вихідного коду та описів проблем, написаних природною мовою.

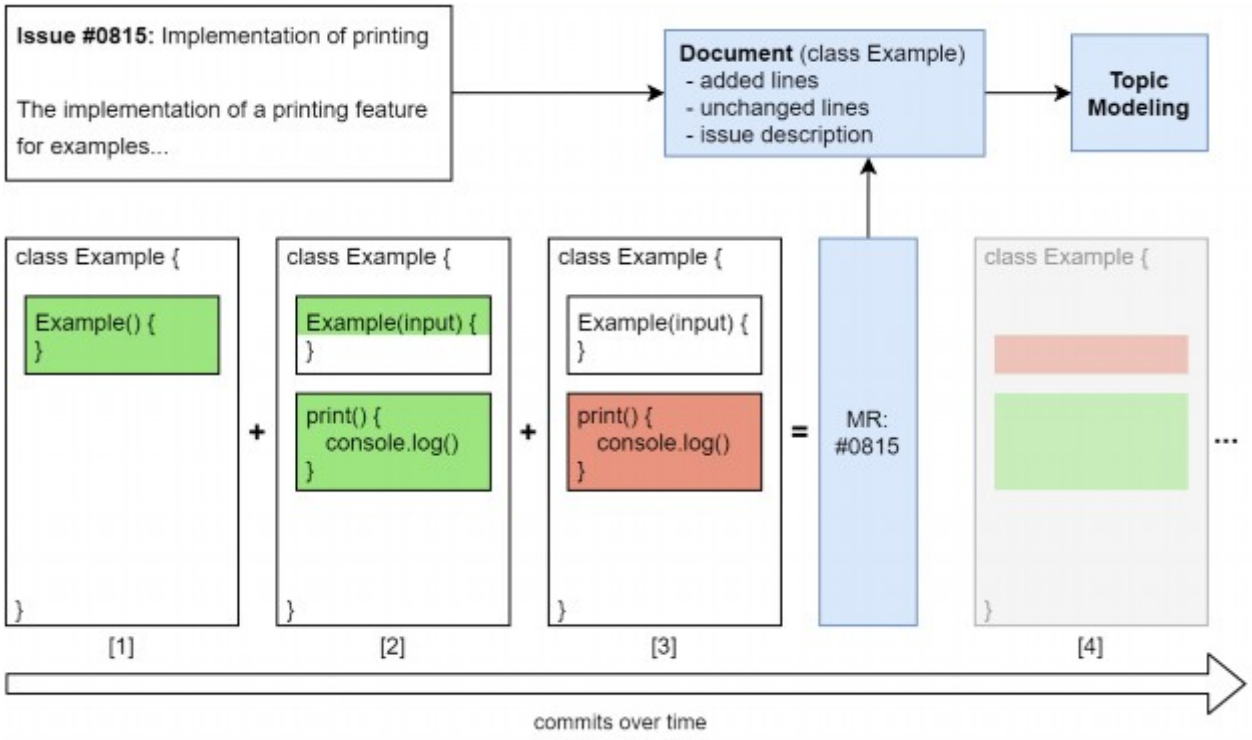


Рис. 2.5. Генерація корпусу з описів проблем та наборів змін на рівні класу

Кожен набір змін складається з кількох комітів, які, у випадку використання Git, агрегуються в запити на злиття, які потім застосовуються до основної гілки репозиторію. При використанні систем відстеження проблем, таких як Jira, ці набори змін пов'язані з описом проблеми за

допомогою ідентифікатора, який записується в поле повідомлення запиту на злиття. З цього створюються документи, які служать вхідними даними для тематичного моделювання. Детальні кроки для отримання даних та їх підготовки до використання описані в розділі 2.4.

Нарешті, попередньо оброблені текстові дані включаються в документ:

- Додані рядки коду
- Незмінені рядки коду
- Опис проблеми

Ця інформація додатково пов'язана з класом, з якого вони були вилучені, щоб пошукова система, яка базується на тематичному моделюванні, могла відображати їх як результати.

Методи, що використовуються для навчання та оптимізації тематичної моделі, детальніше пояснюються в наступному розділі, тоді як ті, що застосовуються для створення пошукової системи на основі моделей, є темою розділу 3.1.

2.4. Підготовка даних для тематичного моделювання локаційного визначення функції

Під час підготовки та виконання визначення місця розташування функцій на основі тематичного моделювання виконується кілька кроків для зчитування та перетворення даних. Методи, застосовані на цих етапах, пояснюються в наступних підрозділах.

2.4.1. Видобуток даних

Деталі реалізації частини видобутку даних інструменту, відповідального за імпорт даних контролю версій та відстеження проблем, пояснюються в наступному розділі. Крім того, вилучення ідентифікаторів проблем з наборів змін та подальше зв'язування артефактів вихідного коду з описами проблем також є темою цих підрозділів у розділі 3. Усі ці кроки

значною мірою залежать від текстової фільтрації на основі регулярних виразів, які зіставляють структурований текст із шаблонами та витягують необхідні токени.

2.4.2. Очищення тексту

Методи, що застосовуються з точки зору очищення тексту, належать до категорії видалення стоп-слів як з англійського тексту, так і з вихідного коду Java, а також видалення найпоширеніших слів як частини навчання LDA та розподілу Пачінко. Хоча останнє є простим параметром, який можна встановити в реалізації методів тематичного моделювання, що використовуються в цій дисертації, видалення стоп-слів є статичною частиною аналізу, яка застосовується в багатьох випадках та точках даних.

Хоча існують попередньо визначені списки слів для фільтрації стоп-слів з англійського тексту, для цієї дисертації було використано власний список. Цей список походить з даних золотого набору Корлі та був обраний для покращення порівнянності результатів з робіт [12, 29] пізніше в оцінці.

Аналогічним чином, стоп-слова для мови програмування Java також походять із золотого набору Корлі та включають загальні ідентифікатори, такі як «abstract», «interface» або «true» та «false».

Щоб завершити очищення тексту, з усіх текстових входів, включаючи описи проблем, артефакти вихідного коду, а також пошукові запити, також видаляються інші символьні символи, цифри та пробіли.

2.5. Тематичне моделювання

Тематичне моделювання з LDA та розподілом Пачінко - це процес, який вимагає оптимізації параметрів, щоб розпізнавати зв'язки між документами та словами, що містяться в них. У цьому розділі буде коротко описано ці доступні параметри та детально описано методи, що використовуються під час оптимізації.

2.5.1 Параметри тематичної моделі

Обидва методи тематичного моделювання, LDA та розподіл Пачінко, мають подібну структуру, як описано в на початку розділу. Вихідні дані, а також параметри, а також опції для налаштування результатів шляхом варіації цих параметрів, подібні для обох методів. У цьому розділі буде коротко описано відповідні налаштування, реалізовані в бібліотеці, що використовується для виконання методів тематичного моделювання, перш ніж перейти до представлення методу налаштування гіперпараметрів.

Кількість кластерів - це найпростіший метод, за допомогою якого можна безпосередньо впливати на модель, щоб вона повертала кращі результати. Тут доступні параметри відрізняються між LDA та розподілом Пачінко. LDA має можливість змінювати один рівень кластерів (k), а розподіл Пачінко, у своїй найпоширенішій формі, має два рівні ($k1$, $k2$).

Апріорні параметри задаються через розподіли Діріхле, які визначають зв'язки між вузлами як LDA, так і PA. Щоб створити кластери на крайніх точках простору, потрібно вибрати вектор з числами $0 < x < 1$, щоб отримати чіткі концентрації кластерів. Лише в попередніх дослідженнях ця опція рідко використовувалася.

У реалізаціях LDA та розподілу Пачінко є два таких параметри: α та η (альфа та ета, показані на рисунку 2.6). Параметр α відповідає за концентрацію тем, а параметр η відповідає за концентрацію слів у межах тем. Оскільки розподіл Пачінко має два рівні тем, α доповнюється α_{sub} для другого рівня.

Крім того, видалення слів, які є поширеними або занадто рідкісними, може бути полегшено інтерфейсами реалізацій LDA та розподілу Пачінко.

Вихідні дані обох методів, коли документ використовується як вхідні дані в модель, є розподілом тем. Методи, що застосовуються до цього для створення пошукової системи, обговорюються пізніше під заголовком «Пошуковий запит».

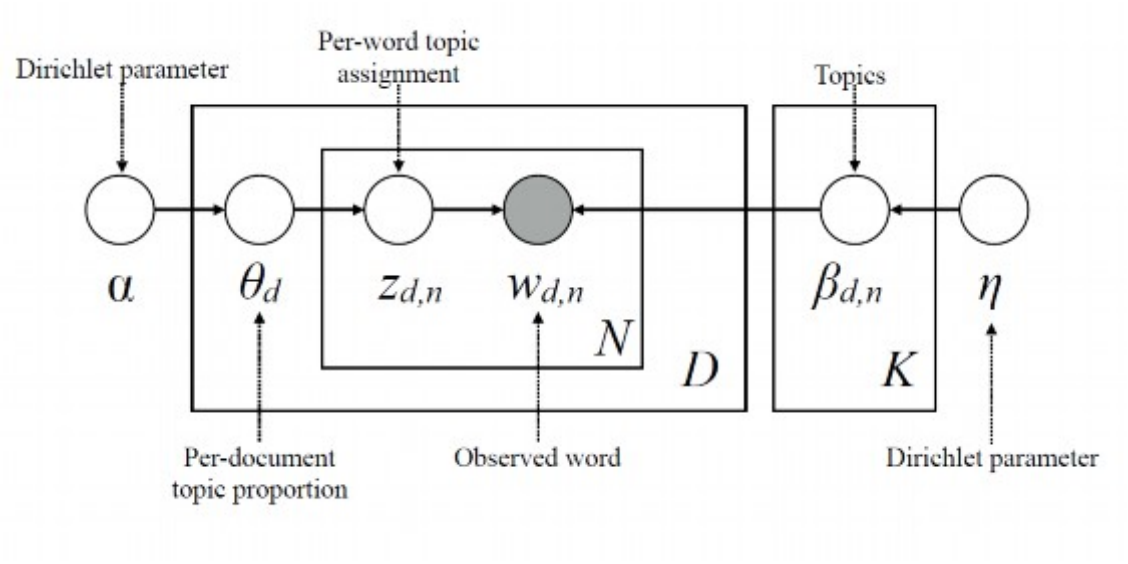


Рис. 2.6. LDA представлена у вигляді графічної моделі з прихованими параметрами та спостережуваними параметрами α та η

2.5.2. Налаштування гіперпараметрів

Налаштування гіперпараметрів - це практика зміни параметрів моделі для досягнення кращих результатів. В принципі, для цього можна використовувати всі параметри, згадані в попередньому розділі, наприклад, шляхом виконання пошуку по сітці або просто шляхом тестування випадкових значень. Зазвичай у дослідженнях на основі тематичного моделювання найбільше уваги приділяється зміні кількості тем, а інші зазвичай вибираються на основі найкращих практик, як попередній досвід.

Метод, застосований у цій роботі, виходить за рамки цього поширеного підходу, проте також починається з зосередження на пошуку кількості кластерів, яка найкраще підходить для даного завдання визначення місця розташування функцій у вихідному коді. Після того, як буде знайдено хорошу кількість кластерів, ця конфігурація буде використана для налаштування інших параметрів. Таким чином, уникається значний час виконання налаштування кількох параметрів одночасно. Параметри, що налаштовуються, крім кількості кластерів, - це обговорювані параметри Діріхле альфа та ета, а також кількість ітерацій над навчальними даними та коефіцієнт вигоряння для оптимізації початкових параметрів.

Дотримуючись найкращих практик, логарифмічна ймовірність використовується як ідентифікатор прогресу оптимізації, причому висока логарифмічна ймовірність представляє хорошу продуктивність моделі під рукою [43].

2.6. Визначення локаційного розташування функцій

Раніше пояснені методи вже натякали на ті, що застосовуються в контексті визначення місця розташування функцій. Відповідно, цей розділ підсумовує ці методи. У ньому детально описано тактику пошуку збігів між розподілами тем, щоб зіставити пошуковий запит із відповідними документами, метрики для вимірювання точності цих збігів та перевірку на основі золотого набору загалом.

2.6.1. Пошуковий запит

Враховуючи пошуковий запит, висновок моделі застосовується через бібліотеку tomotopy. Результатом є розподіл тем з одним або двома рівнями залежно від того, чи є модель моделлю LDA чи моделлю розподілу Пачінко. За допомогою цього розподілу тем, специфічного для документа, асоціацію з іншими документами можна обчислити або шляхом визначення найбільш ймовірної теми запиту та відповідного сортування інших документів, або шляхом обчислення відстані між розподілами тем. Для останнього можна використовувати відхилення, як описано формулою нижче:

$$\frac{1}{n}\sum_{i=1}^{n}|x_i - y_i|$$

У цій формулі x_i - це ймовірність того, що документ належить до теми, а y_i - це ймовірність того, що документ, пов'язаний із пошуковим запитом,

належить до тієї ж теми. Після обчислення для всіх існуючих документів список можливих збігів можна відсортувати за цим балом.

2.6.2. Метрики продуктивності

Для оцінки ефективності результатів визначення місця розташування функцій у попередніх роботах використовується метрика середнього зворотного рангу (MRR), яка застосовується до результатів пошуку. По суті, вона обчислює бал на основі позиції, на якій знаходиться перший релевантний елемент, у списку результатів пошуку. Її формула показана нижче:

$$MRR = \frac{1}{|Q|}\sum_{i=1}^{|Q|}\frac{1}{rank_i}$$

Однією з основних властивостей середнього зворотного рангу є його перевага високо релевантних результатів над незначно релевантними результатами. Це буде питання, яке буде описано в третьому розділі. Аргументи на користь зосередження на цих метриках включають те, що це є кращим рішенням у попередніх дослідженнях, коли є лише кілька або один релевантний документ для пошукових запитів, як це має місце з проектом та золотим набором, що використовуються для дослідження в цій роботі.

Додатковими метриками продуктивності, що використовуються для оцінки та обговорення результатів, є середній та медіанний ранг, а також тест знаків Вілкоксона.

Хоча середній та медіанний ранг є простими метриками для безпосередньої оцінки результату, результат тесту знаків Вілкоксона потребує деяких пояснень. Це пояснюється тим, що він не оцінює самі результати, а їх значущість у порівнянні між двома методами або двома парними наборами даних загалом. Його значення T можна обчислити за такою формулою:

$$T^- = \sum_{i=1}^{N^-} R_i^- \; and \; T^+ = \sum_{i=1}^{N^+} R_i^+ \; and \; T = \min(T^-, T^+)$$

У цій формулі $R_i -$, $R_i +$ - це знакові ранги різниць між результатами B та A, а N−, N+ - відповідна кількість різниць ≠ 0. Знакові ранги тут можна обчислити, упорядкувавши різниці за їх абсолютним значенням у порядку зростання, а потім призначивши значення рангів. У разі зв'язків між значеннями всім їм присвоюється середній ранг.

Враховуючи *T*, значущість між *B* та *A* можна визначити або шляхом порівняння з існуючою таблицею ймовірностей, або шляхом ручного обчислення Z-балу для даної конфігурації. Ручне обчислення можна виконати за наведеними нижче формулами:

$$Z = \frac{\mu - T - 0.5}{\sigma} \; and \; \mu = \frac{n(n+1)}{4} \; and \; \sigma = \sqrt{\frac{[2n+1]\mu}{6}}$$

Тепер для Z-балу, більшого за 1,96, різниця є значущою за рівнем значущості α = 0,05. Це значення відоме з інших тестів, таких як t-тест Стьюдента. Менші Z-бали означали б, що різниця між B та A недостатньо значуща.

Якщо значення рангів не надто відрізняються, Z-бал можна використовувати в контексті нормального розподілу для обчислення p-значення. За допомогою цього нормального розподілу Z-бал можна знайти на осі x, а пов'язані p-значення можна зчитати з осі y. Щоб знайти кумулятивну ймовірність того, що Z-бал знаходиться поза областю, в якій ми припускаємо, що різниці незначні $p < \alpha$, можна обчислити функцію кумулятивного розподілу для обох сторін нормального розподілу, що призведе до визначеної ймовірності (p-значення):

$$p = 2 * (1 - \phi(|Z|))$$

У цій формулі функція ϕ є функцією кумулятивного розподілу стандартного нормального розподілу. За допомогою p-значення можна провести пряме порівняння з рівнем значущості. Якщо p менше за α, різниця є значущою. В іншому випадку такий висновок про значущість не підтверджується даними.

2.6.3. Валідація на основі золотого набору

Окрім метрик продуктивності, метод перевірки, що використовується в цій дисертації, базується на золотих наборах, наданих попередніми дослідженнями. Це означає, що дані, що використовуються як вхідні дані для навчання тематичних моделей, не розділяються на навчальний та тестовий набори даних, як це прийнято в машинному навчанні, а натомість використовуються всі дані, які можна перевірити за допомогою золотого набору. Золотий набір тоді складається з тестових даних, які були надані разом з мітками, або в контексті пошукової системи разом з очікуваними результатами, так що можна обчислити метрики продуктивності для порівняння підходів один з одним. У контексті цієї роботи доступні два життєздатні золоті набори, з яких один буде обрано в наступних розділах, щоб увімкнути перевірку.

Висновки до розділу

У другому розділі представлено ключові концепції та підходи до тематичного моделювання, а також детально описано методологію аналізу текстових даних із метою визначення локаційних функцій. Основним акцентом є розгляд розподілу Діріхле, латентного розподілу Діріхле (LDA) і розподілу Пачінко як базових статистичних моделей для тематики.

Особливу увагу приділено підготовці даних, адже якість видобутку, очищення та нормалізації текстів безпосередньо впливає на результати моделювання. У процесі видобутку даних важливо забезпечити репрезентативність текстових корпусів. Тематичне моделювання розглядається через параметри моделей, такі як кількість тем, та вплив налаштування гіперпараметрів на точність результатів. Зокрема, пояснено методи оптимізації гіперпараметрів для забезпечення стабільності моделі й підвищення її продуктивності.

Для визначення локаційного розташування функцій важливим етапом є формування релевантного пошукового запиту. Це включає побудову запиту, що відображає тематичну структуру тексту, та використання метрик продуктивності для оцінки моделі. Крім того, наголошено на значенні валідації моделі за допомогою "золотого набору" даних, що дозволяє оцінити точність класифікації локацій.

Загалом у розділі підкреслено важливість узгодженості методологічного підходу, починаючи від видобутку даних і завершуючи аналізом результатів тематичного моделювання, особливо в контексті аналізу функцій, пов'язаних із розташуванням.

РОЗДІЛ 3. ІМПЛЕМЕНТАЦІЯ МЕТОДІВ ТА ЗАСОБІВ ТЕМАТИЧНОГО МОДЕЛЮВАННЯ ФУНКЦІЙ ЛОКАЦІЇ

3.1. Загальна стратегія рішення

Цей розділ магістерської роботи присвячений реалізації методів, описаних у розділі 2. Крім того, він описує архітектуру інструментів, які є результатом поєднання цих методів і середовища, в якому вони тестувалися.

Реалізація потрібна для досягнення двох цілей високого рівня: імпорту даних із відповідних джерел даних і виконання/оцінки розташування функції поверх цих агрегованих даних.

Цілі, визначені для функції імпорту, випливають з ідеї, що агрегація даних має бути автоматизованою та стабільною для повторного використання. Реалізація, яка забезпечує розташування функції, з іншого боку, використовуватиметься для експериментування з різними видами технік моделювання теми та потенційно фреймворків і, отже, є більш мінливою. Очікується, що він регулярно змінюватиметься під час розробки, оцінювання та перевірки та може ніколи не досягти остаточного стану. Через це було вирішено розділити проект на дві окремі програми, які спільно використовують базу даних для обміну даними. Таким чином, конкретні цілі розрізняють програму імпортера та програму визначення місця розташування.

Програму визначення локаційного місця розташування функцій слід розглядати більше як прототип або інструмент оцінки, ніж готову програму. Відповідний набір вимог до розташування функції провідності в будь-якій базі коду наведено нижче:

1) Програма визначення розташування функцій повинна мати можливість читати дані, збережені імпортером.

2) Текст має бути попередньо оброблений відповідно до найкращих практик, щоб створити текстовий корпус для кожного набору змін. Вони повинні містити описи функцій, а також сам змінений вихідний код.

3) Виконання процесу навчання для тематичної моделі має використовувати доступні бібліотеки і зберігати навчені моделі, а також таблиці пошуку корпусу для подальшого використання.

4) Оцінка запитів користувача щодо навчених моделей повинна бути реалізована через а інтерфейс командного рядка з результатами для друку на консолі або вихідні файли.

5) Голдсети з принаймні однієї попередньої дослідницької статті необхідно порівняти з нещодавно навченими моделями, а результати, отримані на основі цього, мають автоматично зберігатися як список результатів пошуку для кожного запиту голдсету.

6) Результати оцінених запитів goldset необхідно перевірити на відповідність очікуваним результатам goldset шляхом обчислення значення балу, яке робить можливим порівняння успіху між моделями.

Розробка додатків стримується факторами, пов’язаними із залежністю від зовнішніх джерел даних і технічними залежностями від інструментів сторонніх розробників.

Додаток визначення розташування функцій залежить від наявності доступу до корпусу описів функцій, які відображаються на класи або методи, що містяться в наборах змін. Такі дані можна знайти в системах контролю версій вихідного коду, таких як Git або Subversion, і в програмах для відстеження проблем, таких як GitHub або Jira. Однак через обмеження часу імпортер може реалізувати інтерфейси лише для кількох із цих систем.

Додаток визначення розташування залежить від бібліотек, які надають функціональні можливості для моделювання теми. Серед Java і Python, які мають аналогічні доступні бібліотеки, як мову програмування було обрано Python.

Для розробки як імпортера, так і додатка розташування функцій рішення щодо загальної стратегії розвитку повинні були бути прийняті заздалегідь до початку реалізації.

Вирішальною частиною цього є рішення про те, яку мову програмування використовувати. Це значною мірою залежить від оцінки бібліотек, які підтримують методи, розглянуті в розділі 2.

Додаток визначення місця розташування потребує спеціальних бібліотек для виконання методів моделювання теми, таких як прихований розподіл Діріхле та розподіл Пачінко. Оскільки, як згадувалося раніше у відповідних розділах, існує лише кілька бібліотек, що підтримують розподіл Pachinko, вибір мов програмування дуже обмежений. Порівняння популярних бібліотек моделювання тем наведено в таблиці 3.1.

Таблиця 3.1.

Огляд бібліотек тематичного моделювання

Library	Supported Topic Models	Interface languages
Tomotopy (v0.11.1[6])	✔ LDA (multiple variations) ✔ Pachinko Allocation — Others	Python 3
MALLET (v.2.0.8[7])	✔ LDA (multiple variations) ✔ Pachinko Allocation (hierarchical) — Others	Java
Gensim (v.4.0.0[8])	✔ LDA ✘ Pachinko Allocation — Others	Python 3
Top2Vec (v.1.0.24[9])	✘ LDA (multiple variations) ✘ Pachinko Allocation (hierarchical) — Top2Vec algorithm	Python 3

Gensim та Top2Vec не підходять для завдання порівняння LDA з Pachinko Allocation, оскільки їхня підтримка цих методів обмежена або

взагалі відсутня. Однак на обидві часто посилаються в літературі та онлайн-посібниках [34, 42]. Двома варіантами, які можна вважати придатними, є Tomotopy та MALLET, оскільки обидва вони підтримують принаймні один варіант LDA та Pachinko Allocation. Щоб прийняти рішення, аргументи за і проти цих бібліотек можуть ґрунтуватися на ступені деталізації, наданої в їхній документації, та на особливостях мови програмування, до якої вони надають інтерфейс.

Бібліотека MALLET надає короткий посібник із початку роботи, в якому викладено основні принципи використання інтерфейсу командного рядка. Але оскільки вона написана на Java, вона також надає Java API, який задокументовано за допомогою автоматизованого інструменту документування. Повний список методів тематичного моделювання, що підтримуються MALLET, наведено нижче:

- LDA
- Паралельна LDA
- DMR LDA
- Ієрархічна LDA
- Позначена LDA
- Багатомовна тематична модель
- Ієрархічна модель розподілу Pachinko (PAM)
- Зважена тематична модель
- LDA з інтегрованим виявленням фраз
- Вбудовування (word2vec) з використанням skip-gram з негативною вибіркою

Tomotopy не постачається з інтерфейсом командного рядка, а натомість надає API для Python 3. Хоча ця документація виявилася неповною в деяких незначних випадках, її важливі функції та параметри описані достатньо. Повний список підтримуваних методів наведено нижче:

- Латентне розміщення Діріхле
- Позначена LDA

- Частково позначена LDA
- Контрольована LDA
- Регресія Діріхле з багаточленним розподілом
- Узагальнена регресія Діріхле з багаточленним розподілом
- Ієрархічний процес Діріхле
- Ієрархічна LDA
- Розподіл Pachinko
- Ієрархічний PA
- Корельована тематична модель
- Динамічна тематична модель
- Тематична модель на основі псевдо документів

Оскільки Tomotopy має кращу документацію та підтримує більше варіацій відповідних методів тематичного моделювання, його було обрано як відправну точку для програми визначення місця розташування функцій, що вимагає Python 3 як мову програмування для проекту.

Програма імпорту не має прямих вимог до бібліотеки, яка має функціональність, що не вбудована в більшість поширених мов програмування. Тому вибір мови для реалізації відповідає вибору, зробленому для програми визначення місця розташування функцій.

Оскільки обидві програми будуть засновані на Python 3, як середовище розробки було обрано моно-репозиторій, що дозволяє містити проект в одній загальній системі контролю версій та залишає можливість для спільного використання коду. Цей репозиторій публічно розміщений на GitHub10 і вільно дотримується стратегії розгалуження GitHub Flow.

3.2. Реалізація програми імпорту

Цей розділ містить деталі реалізації програми імпорту. Після вступу контекст, область застосування та стратегії програми обговорюються в підрозділі 3.2.1, а потім детальний опис архітектури інструменту.

3.2.1. Контекст, область застосування та стратегія рішення

Відповідно до передумов, завданням програми імпорту є агрегація даних із систем контролю версій та систем відстеження проблем. Для цілей цієї роботи було вирішено зосередити увагу на одній системі контролю версій та одній системі відстеження проблем як джерелах даних, з можливістю включення більшої кількості в майбутньому. Щоб мати змогу використовувати раніше згадані золоті набори для перевірки результатів визначення місця розташування функцій, ці джерела даних повинні відповідати тим, які використовуються принаймні одним програмним проектом, який був проаналізований у попередніх дослідженнях і має результати, збережені в золотому наборі. Як згадувалося в другому розділі, було визначено два таких золотих набори: "Моделювання тем змін для визначення місця розташування функцій" на основі [43, 44] та "Емпірична оцінка базових методів визначення місця розташування функцій" на основі статті [45]. В таблиці 3.2 дано огляд проектів, що містяться в цих золотих наборах, і містить інформацію про те, який проект використовує яку систему контролю версій та систему відстеження проблем. Таким чином, її можна використовувати як основу для аргументації в процесі вибору однієї з цих систем для інтеграції в імпортер. Підсумок відповідної інформації можна знайти в таблиці 3.3 нижче.

Дані показують, що більшість проектів із двох золотих наборів в основному використовують Git як систему контролю версій. Вибір Git як джерела даних має ту перевагу, що це децентралізована система, і тому не вимагає постійного підключення до певних хостингових сервісів. Замість цього можна використовувати добре задокументований і широко використовуваний інтерфейс командного рядка Git для завантаження копії репозиторію з будь-якого розміщеного екземпляра Git і зчитування необхідних даних. Таким чином, підхід не обмежується GitHub або будь-яким хостинг-провайдером і може, наприклад, також працювати з GitLab або

власним рішенням, сумісним з Git. На цій основі було вирішено, що стратегія рішення повинна включати імпорт репозиторіїв Git через Git CLI.

Таблиця 3.2.

Порівняння зразкових наборів (Goldset)

Project	Lero	Corley	VCS / Issue Tracking
AgroUML	✔	✔	Prev.: SVN (Tigris) / Tigris (not available) 2019: Git (GitHub) / GitHub
Bookkeeper	✘	✔	Git (GitHub) / Jira
CommonsMath	✔	✘	Git (GitHub) / Jira
CommonsLang	✔	✘	Git (GitHub) / Jira
Derby	✔	✔	Git (GitHub) / Jira
Eclipse	✔	✘	Git (eclipse.org) / Bugzilla
Hibernate	✘	✔	Git (GitHub) / Jira
iBatis	✔	✘	Git (GitHub) / GitHub
JabRef	✔	✔	Prev.: Git (SourceForge) / SourceForge 2015: Git (GitHub) / GitHub
jEdit	✔	✔	SVN (SourceForge) / SourceForge
Lucene	✘	✔	Git (GitHub) / Jira
Mahut	✘	✔	Git (GitHub) / Jira
MuCommander	✔	✔	Git (GitHub) / GitHub
Mylyn	✔	✘	Git (eclipse.org) / Bugzilla
OpenJPA	✘	✔	Git (GitHub) / Jira
Pig	✘	✔	Git (GitHub) / Jira
Rhino	✔	✘	Git (GitHub) / GitHub
Solr	✘	✔	Git (GitHub) / Jira
Tika	✘	✔	Git (GitHub) / Jira
Zookeeper	✘	✔	Git (GitHub) / Jira

Таблиця 3.3.

Склад проектів у золотих наборах

VCS	
Git	18
SVN	2

Hosting	
GitHub	17
SF	1
Custom	2

Issue Tracking	
Jira	12
GitHub	5
Bugzilla	2
SF	1

Вибір Jira як першого рішення для відстеження проблем для інтеграції в імпортер знову було зроблено на основі даних у таблицях 3.2 і 3.3. Оскільки 12 з 18 проектів, які використовують Git як систему контролю версій, відстежують проблеми на екземплярі Jira, це вибір, який дає інструменту найвищу гнучкість, теоретично дозволяючи на етапі оцінки цієї дисертації проводити валідацію на основі більш ніж половини доступних золотих наборів.

Щоб мати змогу запускати імпорт кілька разів, не вимагаючи ручного налаштування середовища виконання з сумісними інсталяціями Git, Python та бібліотек Python, було також вирішено контейнеризувати програму імпорту за допомогою docker.

Для цієї роботи імпортер не повинен періодично оновлювати дані, імпортовані з Git, проте потрібне постійне сховище для даних, яке дозволяє іншим програмам отримувати до них доступ за допомогою стандартних галузевих рішень. Це сховище має бути доступним поза середовищем імпортера, оскільки інші програми можуть працювати на виділених машинах і не повинні самі надавати та запускати екземпляр імпортера. Для реалізації цієї стратегії необхідно налаштувати глобально доступний екземпляр бази даних. З точки зору технології використання реляційної бази даних було оцінено порівняно з використанням нереляційної бази даних, такої як MongoDB. Було обрано MongoDB, оскільки структури даних, які необхідно

зберігати, мають сильну зв'язність: наприклад, зміни в певних файлах можна зберігати в тому самому документі, що й самі файли.

Щоб забезпечити доступність з-за меж локальної мережі, було обрано безкоштовний рівень MongoDB Atlas як хостинг-провайдера для екземпляра бази даних. Щоб додатково дозволити імпорт кількох репозиторіїв в один і той самий екземпляр MongoDB, кожен репозиторій буде зберігатися у власній базі даних.

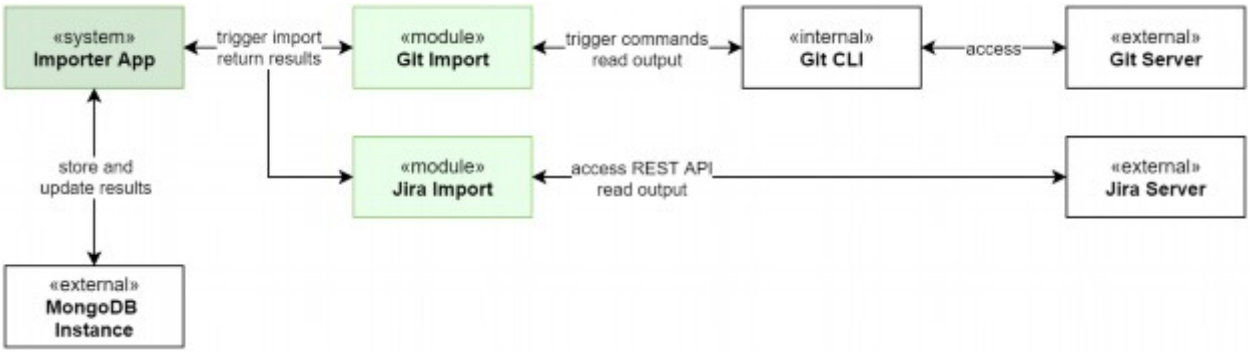


Рис. 3.1. Контекст програми імпорту

Загальний контекст імпортера можна візуалізувати, як показано на рисунку 3.1. Він складається з трьох модулів, один з яких є центральним системним модулем (темно-зелений), а два мають певну відповідальність, до якої звертається системний модуль (світло-зелений). Відображаються інші внутрішні та зовнішні системи (білим), які використовуються цими модулями.

Розглянемо компоненти та їх взаємодію. Importer App - головний компонент програми, який ініціює процес імпорту та отримує результати. Git Import - модуль, відповідальний за імпорт даних з Git, що запускає команди Git CLI. Git CLI взаємодіє з Git Server (сервером Git, наприклад, GitHub, GitLab) для отримання даних репозиторію. Jira Import - модуль для імпорту даних з Jira. Використовує REST API для доступу до Jira Server та отримання інформації про задачі. MongoDB Instance - зовнішній компонент - база даних MongoDB, де зберігаються імпортовані дані.

Потік даних (рис. 3.1):

- Importer App запускає імпорт даних.

- Git Import та Jira Import отримують дані з відповідних джерел.

- Importer App отримує результати імпорту від модулів.

- Результати зберігаються та оновлюються в MongoDB Instance.

Загалом, схема ілюструє процес імпорту даних з Git та Jira, керований Importer App, та зберігання цих даних в MongoDB. У таблиці 3.4 ці компоненти знову перераховані з описом їх призначення.

Таблиця 3.4.

Опис компонентів у контексті програми імпортера

Component	Description
Importer App	The starting point of the Python application. The main app is responsible for initiating processes and for passing on data between them and the data storage.
MongoDB	A globally available instance of the non-relational database MongoDB hosted on MongoDB Atlas.
Git Import	Python module of the *importer* application responsible for handling Git. It is the task of this module to trigger commands on the Git CLI that on the one hand pull data from the Git Server as well as initiating commands reading the Git CLI output. Its thereby crawling and interpreting the repository's commit history.
Git CLI	An instance of the Git CLI installed on the host operating system. It is accessed by the Git Import Python module.
Git Server	External component hosted on a computer that is available publicly on the internet or locally on a network.
Jira Import	Python module of the *importer* application responsible for handling the import of issues from a Jira instance, which it accesses through the corresponding REST-API.
Jira Server	External component hosted on a computer that is available publicly on the internet or locally on a network.

3.2.2 Вигляд структурних блоків

Відповідно до стратегії рішення, описаної в попередньому підрозділі, та визначеної логічної структури програми імпорту, в цьому розділі будуть

обговорені особливості реалізації. Як показано на рисунку 3.2, структурні блоки програми розділені на три високорівневі модулі: головний app.py, git та issues.

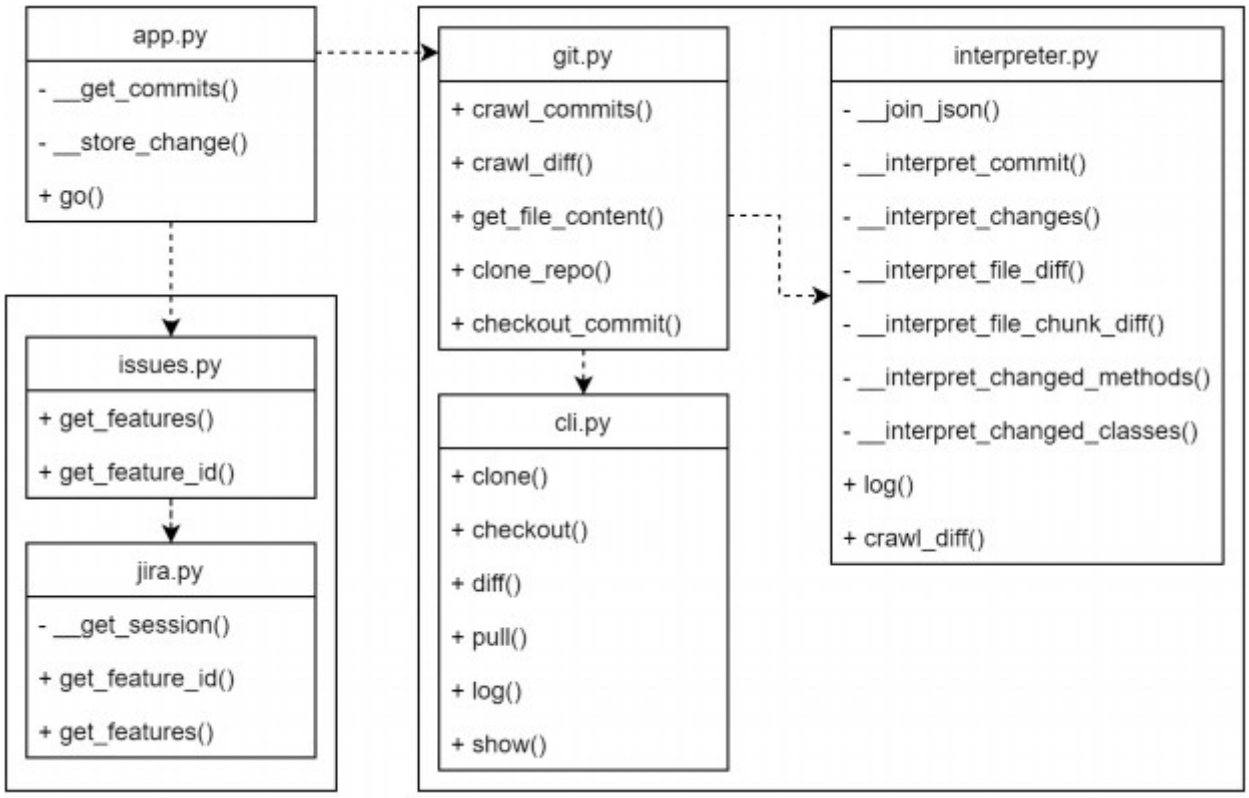


Рис. 3.2. UML-діаграма структурних блоків у програмі імпорту

У наступних розділах завдання та функціональність цих модулів будуть пояснені в технічному контексті.

Головна програма з файлом app.py відповідає за порядок виконання процесу імпорту, візуалізованого на рисунку 10. Вона також відповідає за доступ, об'єднання та збереження даних у базі даних. Це дозволяє іншим модулям діяти незалежно від конкретного типу сховища даних.

Перші два кроки запускають функціональність модуля Git. По-перше, сканується журнал Git (git log), щоб отримати інформацію про те, які файли були змінені та коли. По-друге, ці змінені файли досліджуються один за одним, зчитуючи кількість змінених рядків та пов'язуючи їх з класами та методами, до яких вони належать.

Рис. 3.3. Процес імпорту

Другий крок запускає функціональність модуля issues. Тут проблеми, пов'язані з раніше просканованими комітами, імпортуються з системи відстеження проблем проекту.

Нарешті, дані зберігаються: всі коміти зберігаються в одній таблиці, всі файли та інформація про їх зміни - в іншій, а всі завантажені проблеми - в третій. Загалом, діаграму бази даних можна намалювати, як показано на рисунку 3.4 нижче.

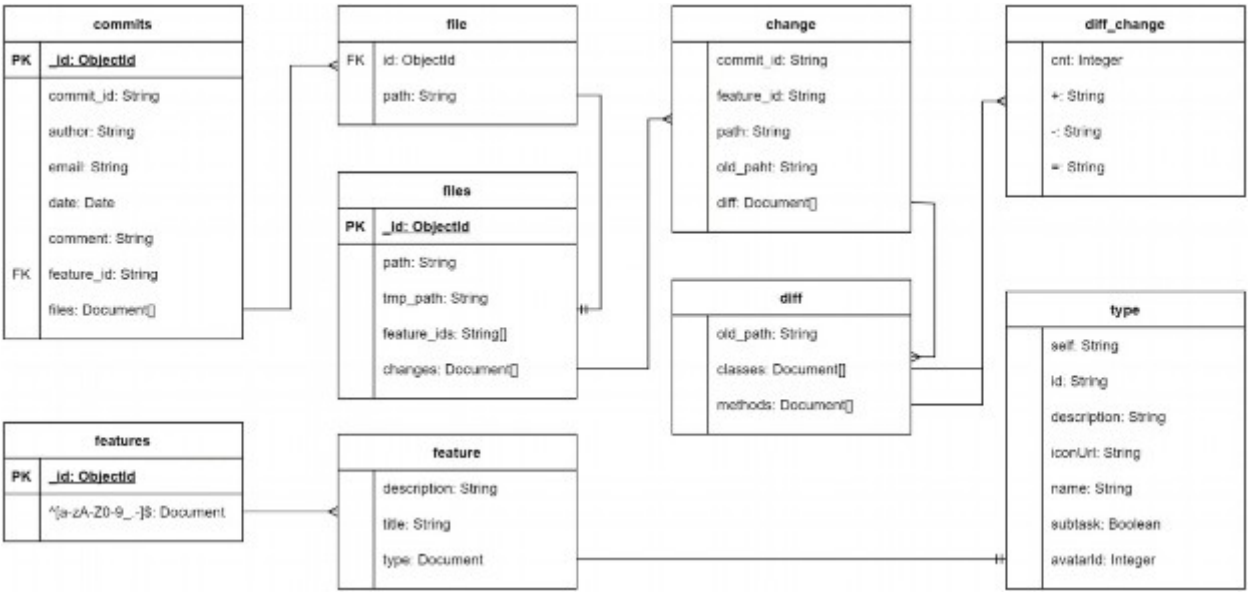


Рис. 3.4. Діаграма бази даних

Модуль Git відповідає за агрегацію відповідних даних з репозиторію Git, доступного в локальній мережі або в Інтернеті. У контексті головної програми та після того, як репозиторій став доступним офлайн, цей процес агрегації можна розділити на два кроки, які вже згадувалися в попередньому розділі: сканування журналу Git для отримання інформації про коміти та

сканування різниці Git (git diff) цих комітів для артефактів рівня методу та класу.

Реалізація цієї функціональності є розширенням форку існуючого вихідного коду, який був створений раніше в рамках дослідницького проекту [21], проведеного його автором.

У межах модуля є три компоненти: точка входу модуля git.py, адаптер до Git CLI cli.py та інтерпретатор виводу CLI interpreter.py.

3.2.3. Модуль задач Jira

Модуль задач відповідає за агрегування проблем із систем відстеження проблем і повернення їх до основної програми. На рисунку 3.3 цей крок позначено як сканування проблем. На даний момент він складається з файлу issues.py як точки входу модуля та адаптера для доступу до даних з екземпляра Jira, реалізованого у файлі jira.py.

Щоб встановити з'єднання з примірником Jira через HTTP, файл jira.py має отримати маркер від цього примірника під час створення нового сеансу. Після цього до окремих проблем можна отримати доступ через URL-адресу за схемою, наведеною в лістингу 3.1 нижче, де домен і параметри project і issueid потрібно змінити відповідно.

Лістинг 3.1. Зразок URL-адреси для доступу до проблеми з примірника Jira

```
http://issues.sample.com/{p roject}/rest/api/2/issue/{issue_id}
```

Під час процесу сканування всі коміти повторюються, і кожна окрема проблема завантажується з сервера. Потім описи проблем поміщаються до словника з ідентифікатором проблеми як ключем і описом як значенням.

Завдяки модульності рішення з файлом issues.py, який є точкою доступу перед адаптером Jira, інші системи відстеження проблем можна додати пізніше. Це також стосується вилучення ідентифікатора функції з

повідомлень комітів, що є методом, реалізованим у jira.py , який використовується Git-модулем interpreter.py через issues.py.

3.2.4. Фіналізація даних

Процес імпорту завершується об'єднанням агрегованих даних. До цього моменту було створено чотири окремі пули даних, як показано в таблиці 3.5.

Таблиця 3.5.

Огляд даних, створених етапами імпорту

Data pool	Description

Commits	A list of commits structured as JSON documents:
	`[{`
	`'commit_id': string,`
	`'author': string,`
	`'email': string,`
	`'date': date,`
	`'comment': string,`
	`'feature_id': string`
	`}]`

Changes	A list of changes on a file-level structured as JSON documents:
	`[{`
	`'commit_id': string,`
	`'feature_id': string,`
	`'path': string,`
	`'old_path': string`
	`}]`

Diffs	A dictionary of differential changes on a source code level structured as JSON documents with filenames as keys[12]:
	`{`
	`'?path': {`
	`'old_path': string,`
	`'classes': {`
	`'?class_name': {`
	`'cnt': int, '+': int, '-': int`
	`}, ...`
	`},`
	`'methods: {`
	`'?method_name': {`
	`'cnt': int, '+': int, '-': int`
	`}, ...`
	`}`

Features	A dictionary of issue descriptions structured as a JSON document with issue IDs as keys[12]:
	`{`
	`'?feature_id': {`
	`'description': string,`
	`'type': {`
	`'self': string,`
	`'id': string,`
	`...`
	`}`

Пули даних комітів та функцій зберігаються безпосередньо у відповідних колекціях бази даних MongoDB.

З іншого боку, зміни та диференціали (diffs) необхідно об'єднати та обробити. Спочатку диференціали додаються до змін на основі шляху до файлу. Після цього необхідно виконати відстеження перейменованих файлів, зіставляючи старі імена файлів вже збережених змін з новим ім'ям файлу поточної оцінюваної зміни.

Нарешті, імпорт завершено, і дані можуть бути зчитані іншими програмами, що відповідають специфікаціям діаграми бази даних на рисунку 3.4 на початку цього розділу.

3.3. Визначення локаційного місця розташування функцій

Цей розділ містить деталі реалізації програми визначення місця розташування функцій. Він має схожу структуру з попереднім розділом та розділений на підрозділи для обговорення контексту, області застосування та стратегії рішення, а також документації архітектури інструменту.

3.3.1. Контекст, область застосування та стратегія рішення

В рамках загальної стратегії рішення, описаної в попередньому розділі, програма визначення місця розташування функцій відповідає за виконання та порівняння визначення місця розташування функцій на основі тематичного моделювання. Через це вона повинна виконувати три різні завдання, які в цьому розділі будуть позначені як навчання (train), оцінка (evaluate) та валідація (validate).

Вибір реалізації завдань як програми командного рядка Python 3 також можна вивести з попереднього розділу. Тут окремі завдання повинні виконуватися незалежно, зберігаючи вивід у форматі, який, за необхідності, може бути зчитаний іншими.

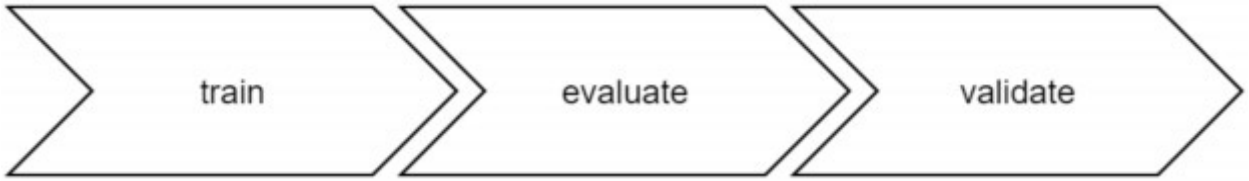


Рис. 3.5. Завдання процесу визначення місця розташування функцій

На відміну від програми імпорту, розробка цих завдань не йшла заздалегідь визначеним шляхом. Натомість технології та методи обмінювалися або видалялися під час реалізації та тестування, щоб адаптуватися до результатів тематичного моделювання, які спостерігалися. Тому стратегія рішення може містити перелік опцій, які потрібно дослідити та реалізувати, але вони можуть не потрапити до фінальної програми або пізніше будуть доступні лише через певні команди.

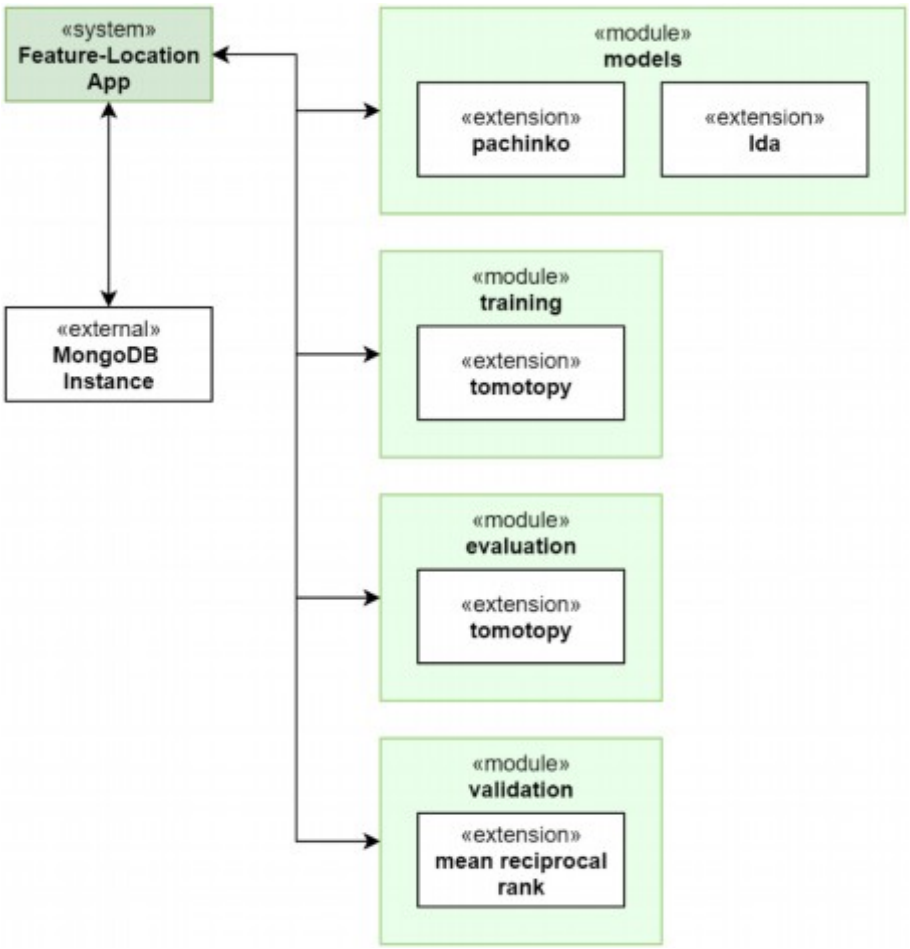


Рис. 3.6. Контекст програми визначення місця розташування функцій

На рисунку 3.6 показано контекст програми визначення місця розташування функцій. Він складається з основної системи (темно-зелений), зовнішнього екземпляра MongoDB (білий) та чотирьох модулів (зелений), які містять розширення, призначені для заміни, оскільки вони реалізують доступ до певних бібліотек або алгоритмів, де також можуть бути досліджені альтернативи.

Завдання навчання з його реалізацією в модулі навчання, природно, використовує базу даних, заповнену програмою імпорту як джерело даних. На основі того, що доступно з бази даних, необхідно знайти стратегії як для попередньої обробки даних, так і для загального формування корпусу, щоб використовувати його як вхідні дані для виконання тематичного моделювання.

Хоча база даних технічно надає три групи текстових джерел, які описують зміни, лише дві з них будуть використані в реалізації завдання навчання: описи проблем та ті рядки вихідного коду, які були змінені. Вони є основою для формування корпусу, як показано на рисунку 3.8, та для опису функцій. Повідомлення про коміти як група текстових джерел, з іншого боку, не використовуються, оскільки вони не вважаються такими, що надають інформацію про високорівневі функції. Це відповідає методам, що використовуються в інших роботах, які досліджують тематичне моделювання на основі змін.

Попередня обробка даних полягає в токенізації цих описів, а також у фільтрації стоп-слів. Для цього потрібно оцінити два варіанти: використання англійських стоп-слів з бібліотеки Python NLTK та альтернативний підхід з використанням власних стоп-слів як для англійської, так і для мови програмування Java, знайдених в реалізації інших досліджень [19, 39].

Формування корпусу має бути застосовано до ряду документів, які у вигляді файлів та історії їх змін зчитуються з бази даних. Тут користувач повинен мати можливість вибирати параметри, необхідні для навчання доступних моделей, через інтерфейс командного рядка програми. Як

обговорювалося раніше, для реалізації має використовуватися бібліотека tomotopy, яка підтримує принаймні модель LDA та модель розподілу Pachinko.

Результати навчання, яке застосовується до цього корпусу на основі документів, необхідно зберегти для подальших етапів. Вони повинні складатися з файлу моделі, який буде використаний для виведення нових документів, та таблиці пошуку, яка містить кожен документ, на якому була навчена модель, разом з пов'язаним з ним розподілом тем.

Таблиця 3.6.

Опис компонентів у контексті програми визначення місця розташування функцій

Component	Description
Feature-Location App	The starting point of the Python application. The main app is responsible for starting one or multiple of the available tasks (*train*, *evaluate*, *validate*). It also contains code that can train both LDA and Pachinko Allocation with a variating number of clusters. It provides a command-line interface for starting the application.
MongoDB	A globally available instance of the non-relational database MongoDB hosted on MongoDB Atlas. It is filled by the *importer* application and contains data on changes made to the repositories as well as features from the issue tracking systems. On that MongoDB instance, each imported repository has its own database.
Models	Python module containing the model-specific as well as the common implementation logic of LDA and Pachinko Allocation using the tomotopy library.
Training	Python module that provides the relevant functions for initiating the training process. For this, it is being accessed by the main application. It currently offers an implementation for training with tomotopy, but the library-specific logic is exchangeable.
Evaluation	Python module that is responsible for evaluating the goldset queries by inferring them against a trained model. Besides generic evaluation functionality, it also acts as a wrapper that supports tomotopy models.
Validation	Python module for validating previously evaluated goldset queries against the anticipated results from the same goldset.

На основі збережених моделей та таблиць пошуку як для LDA, так і для розподілу Pachinko, етап оцінки вимагає створення результатів пошуку або для запитів з золотого набору, або для окремих користувацьких запитів, зчитаних з інтерфейсу командного рядка. Для цього мають бути використані функції виведення бібліотеки tomotopy. Після того, як документ був виведений, застосовується стратегія пошуку відповідних результатів, що відповідають розподілу тем. Для цього було вирішено дослідити два варіанти: простий пошук кластера, який зазвичай має найвищу ймовірність, та оцінку відстані між розподілами ймовірностей. Ці результати пошуку мають бути збережені як документи JSON для подальшого аналізу.

3.3.2. Вигляд структурного блоку

Специфіка впровадження, описана в цьому розділі, випливає зі стратегії вирішення вище, а також з інших попередніх розділів.

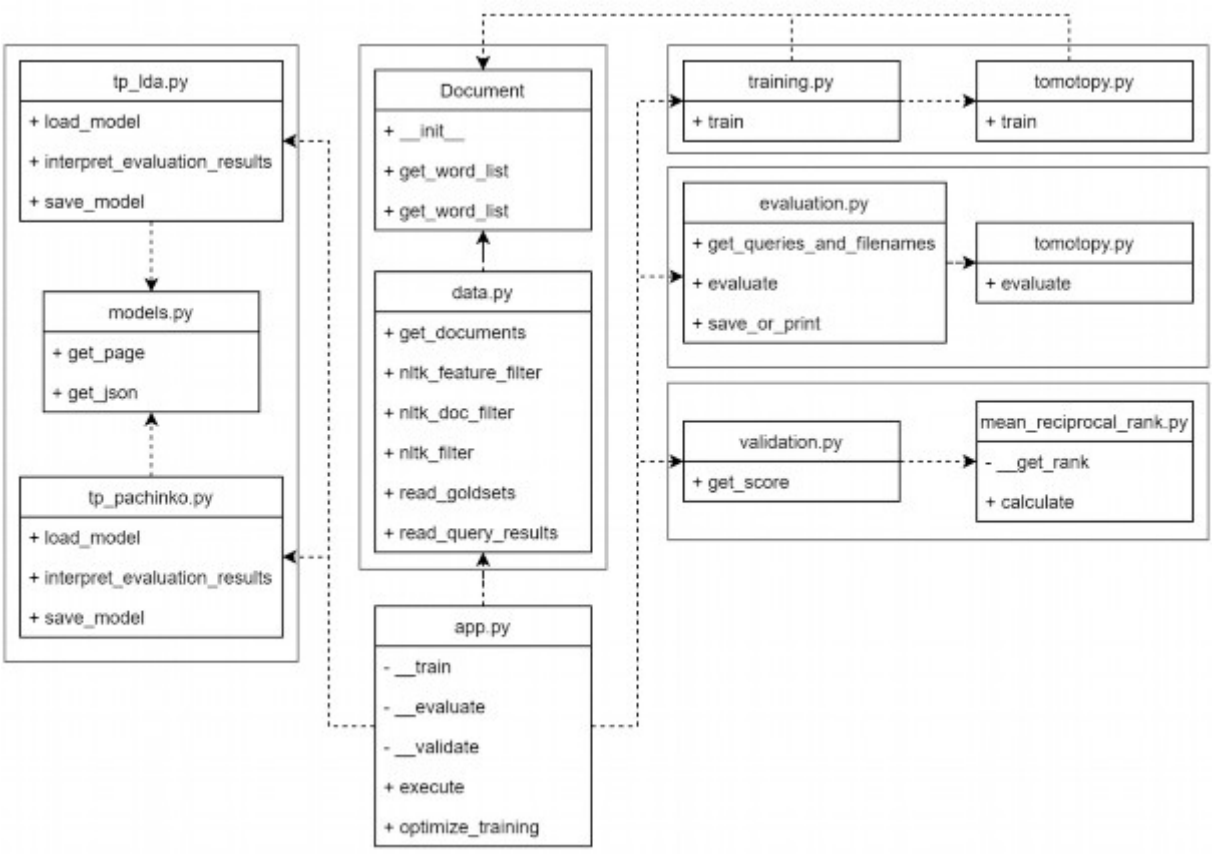


Рис. 3.7. UML-діаграма будівельних блоків у програмі визначення місця розташування функцій

Як показано на діаграмі на рисунку 3.7, реалізація поділена на 6 компонентів високого рівня, які вже були представлені в попередньому розділі: app.py, моделі, дані, навчання, оцінка та перевірка.

Файл app.py надає інтерфейс командного рядка для доступу до найпоширеніших функцій програми. Під час виконання скрипту доступні вхідні параметри, перелічені в таблиці 3.7.

Таблиця 3.7.

Опції командного рядка файлу app.py

Parameter	Description	Options
`-t, --train`	Choosing the models that should be trained.	`lda, pa`
`--lda_k1`	Number of clusters if LDA is being trained.	`Integer`
`--pa_k1`	Number of first level clusters for training Pachinko.	`Integer`
`--pa_k2`	Number of second-level clusters for training Pachinko.	`Integer`
`-b, --base`	Choosing either file or class-based training.	`class, file`
`-e, --eval`	Choosing the models that should be evaluated.	`lda, pa`
`-q, --query`	Search query for which a topic must be found.	`String`
`-i, --input`	Query and goldset directory as an alternative to a custom query.	`Path`
`-p, --pages`	Number of pages to display in the evaluation result.	`Integer`
`-d, --determ`	Evaluation method used to determine which documents match the search query.	`ml, dist`
`-v, --validate`	Choosing the models that should be validated.	`lda, pa`

У наступних розділах особливості реалізації цих блоків описані на високому рівні, розділені на три завдання, визначені в стратегії рішення

попереднього розділу: навчання (train), оцінка (evaluate) та валідація (validate).

3.3.3. Процес навчання

Процес навчання починається із завантаження документів з бази даних MongoDB та їх форматування відповідно до тематичного моделювання на основі класів або файлів. Аналогічно, описи проблем завантажуються з бази даних та зберігаються в контексті виконання програми. Це обробляється файлом app.py.

Фактичний процес навчання моделей LDA та розподілу Pachinko за допомогою бібліотеки tomotopy є загальним та реалізований у модулі навчання файлу tomotopy.py. Він отримує модель для LDA або розподілу Pachinko, згенеровану з модуля models файлом app.py через скрипт training.py, який діє як адаптер. Завдяки реалізації training.py як адаптера, код, специфічний для tomotopy, можна легко замінити, якщо відповідні моделі були попередньо створені.

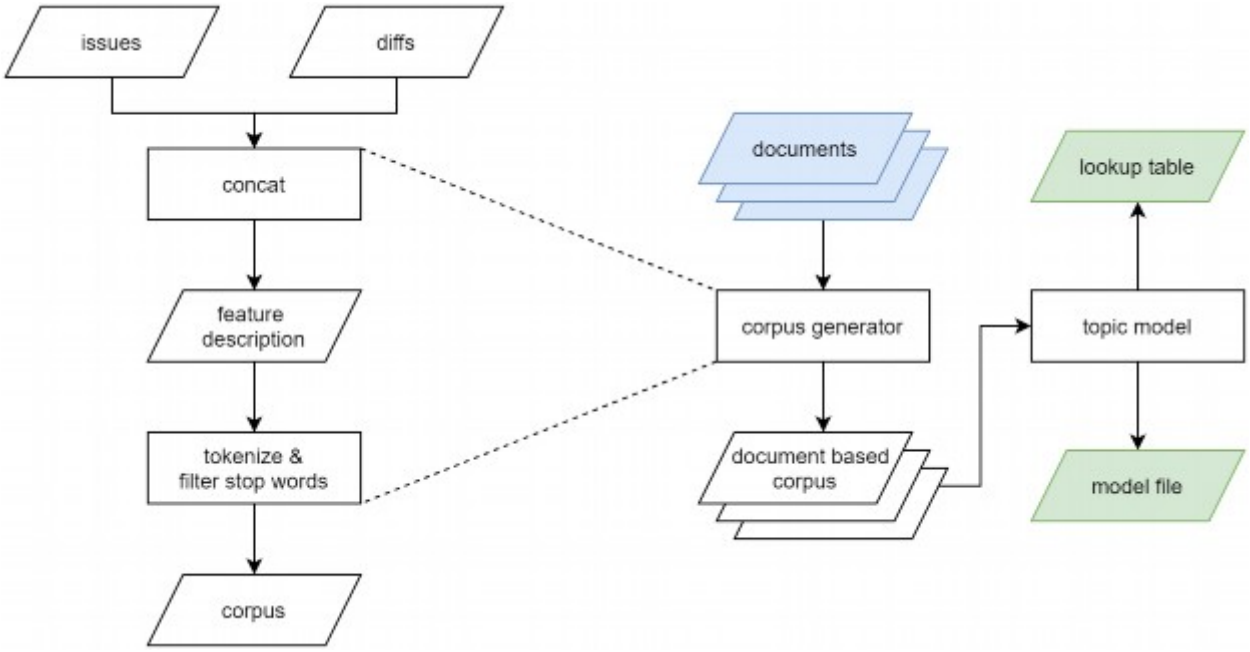


Рис. 3.8. Процес навчання

На рисунку 3.8 показано процес з високим рівнем деталізації. Починаючи з формування корпусу на основі документів, описи функцій генеруються шляхом об'єднання описів проблем, а також заголовків та диференціалів файлів, включаючи додані рядки та ті, що залишилися незмінними. Ці описи функцій потім токенізуються на окремі слова та фільтруються, щоб більше не містити стоп-слів. Цей корпус потім використовується для навчання моделей, що призводить до виведення файлу моделі та таблиці пошуку у вигляді файлу CSV. У цій реалізації таблиця пошуку містить принаймні всі документи з їх шляхом та іменем, а також розподіл ймовірностей кожного документа для кожної теми.

3.3.4. Оцінка

На основі файлу моделі, згенерованого раніше в завданні навчання, та запиту можна визначити розподіл ймовірностей тем для цього запиту. Використовуючи таблицю пошуку, яка також була згенерована на попередньому кроці, створюються результати пошуку. Це робиться шляхом зіставлення документів у таблиці пошуку з розподілом тем. Графічний огляд наведено на рисунку 3.9.

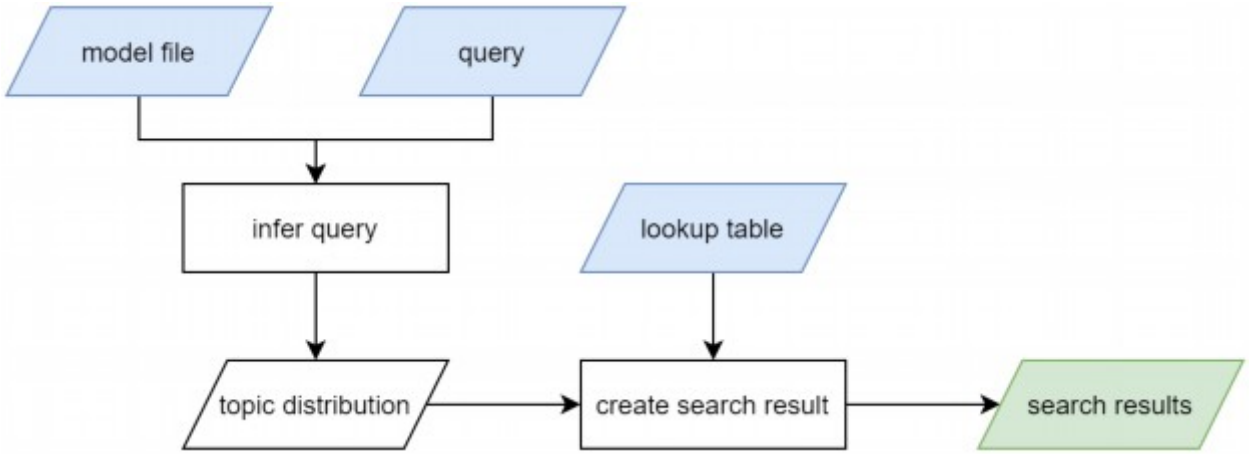


Рис. 3.9. Блок-схема завдання оцінки

Як вимагає стратегія рішення, існує два варіанти для створення цих збігів. Перший - це просте сортування документів у таблиці пошуку за спаданням за найбільш ймовірною темою запиту. Другий - це порівняння відхилення між усіма темами документа та запитом, як описано в попередньому розділі. Обидва реалізовані та доступні через інтерфейс командного рядка, причому останній є опцією за замовчуванням.

Якщо замість власного запиту було обрано вхідний каталог, процес запускається для вмісту всіх текстових файлів у цьому каталозі. Крім того, вивід зберігається у вигляді текстових файлів, у цьому випадку - у підкаталозі вхідних даних з назвою "lda" або "pa" залежно від того, який тип моделі було оцінено. Приклад такого файлу результатів пошуку можна побачити в лістингу 3.2 нижче.

Лістинг 3.2. Початок результату пошуку на основі класів для репозиторію zookeeper

```
{
    "log_ll": -158.93582153320312,
    "res": [
        {
            "_id": "6062e8fea09c5e09e16fe3e5",
            "path": ".../Login.java -> Login",
            "name": "Login"
        },
        {
            "_id": "6062e8fea09c5e09e16fe3e6",
            "path": ".../client/ZooKeeperSaslClient.java -> ZooKeeperSaslClient",
            "name": "ZooKeeperSaslClient"
        },
        ...
}
```

3.3.5. Валідація

Після проведення оцінки запитів, що містяться в золотому наборі, валідація повинна згенерувати оцінку, яку можна використовувати для визначення ефективності методу тематичного моделювання у визначенні місця розташування функцій. Вона вимагає визначення вхідного каталогу через інтерфейс командного рядка та використовує дані, що зберігаються

там, для порівняння раніше згенерованих результатів пошуку для запитів золотого набору з очікуваними результатами цих золотих наборів. Потік показано на рисунку 3.10

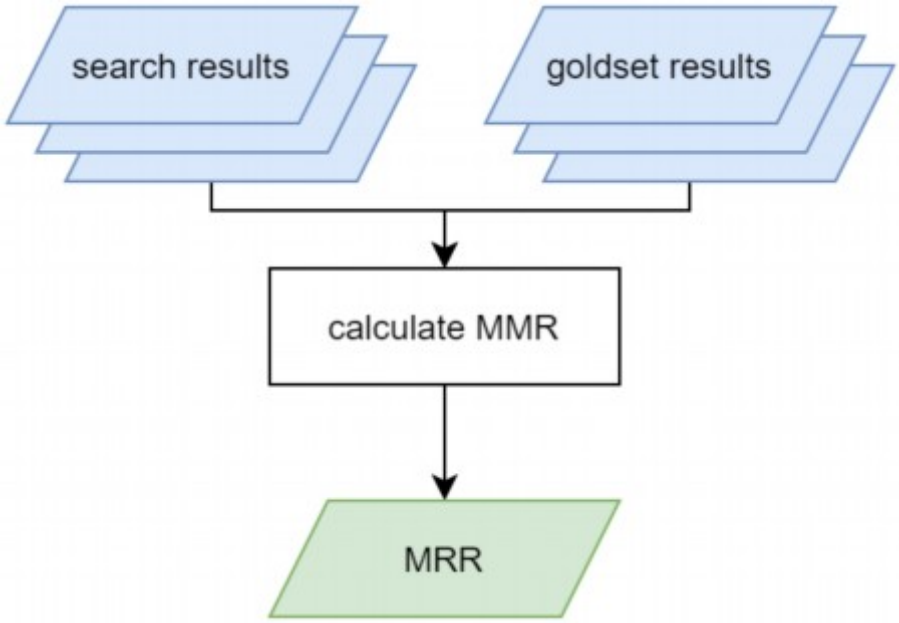


Рис. 3.10. Блок-схема завдання валідації

Метод, реалізований для цього - це метрика середнього зворотного рангу. Його формула вже була представлена в другому розділі. Його результат описує середній ранг, на якому можна знайти перший правильний результат у всій колекції результатів пошуку.

3.4. Перешкоди в процесі визначення місця розташування функцій

Під час етапу оцінки цієї роботи інструмент для виконання визначення місця розташування функцій на основі тематичного моделювання був протестований шляхом запуску з різними конфігураціями. Протягом часу, проведеного з оцінкою, стало очевидно, що в основній бібліотеці інструменту tomotopy є проблема, яка впливає на детермінізм процесу навчання. Це призводить до обмеженої відтворюваності представлених результатів.

Незважаючи на те, що tomotopy пропонує можливість встановити значення початкового числа (seed value) для відтворення результатів під час навчання моделей для обох методів, моделі, створені з однаковою конфігурацією параметрів, в кінцевому підсумку працювали по-різному. Вони демонструють незначну варіацію в оцінці логарифмічної правдоподібності, і їх результати можуть значно відрізнятися при оцінці та валідації відносно золотого набору. Приклад у вигляді виводу під час навчання, оцінки та валідації для двох, здавалося б, ідентичних процесів навчання LDA та їх результатів показано в лістингу 3.3.

Лістинг 3.3. Навчання, оцінка та валідація моделі LDA, яка працює по-різному, незважаючи на те, що були використані однакові параметри та значення початкового числа

```
> python app.py -t lda --lda_k1 100 -e lda -v lda -i .\tmp\
--- train ------
Iteration: 10                                Log-likelihood: -6.812546821579827
Iteration: 20                                Log-likelihood: -6.552810377639443
Iteration: 30                                Log-likelihood: -6.466817423457372
Iteration: 40                                Log-likelihood: -6.423062776760207
Iteration: 50                                Log-likelihood: -6.398546375060162
Iteration: 60                                Log-likelihood: -6.387198130807414
Iteration: 70                                Log-likelihood: -6.376373306876543
Iteration: 80                                Log-likelihood: -6.372046251722799
Iteration: 90                                Log-likelihood: -6.366875461321159
Iteration: 100                               Log-likelihood: -6.364717015332481
LDA ll per word          -6.364717015332481
--- evaluate ---
| |                    #                                 | 159 Elapsed Time: 0:01:16
--- validate ---
LDA MRR:        0.18178203688584976
Time Lapsed = 0:1:59.99697017669678

> python app.py -t lda --lda_k1 100 -e lda -v lda -i .\tmp\
--- train ------
Iteration: 10                                Log-likelihood: -6.722635893883447
Iteration: 20                                Log-likelihood: -6.5114688372249985
Iteration: 30                                Log-likelihood: -6.43296017583458
Iteration: 40                                Log-likelihood: -6.398586748371089
Iteration: 50                                Log-likelihood: -6.387941462968303
Iteration: 60                                Log-likelihood: -6.379946728737874
Iteration: 70                                Log-likelihood: -6.378963118261652
Iteration: 80                                Log-likelihood: -6.377343502242806
Iteration: 90                                Log-likelihood: -6.3754312534110555
Iteration: 100                               Log-likelihood: -6.374021941608899
LDA ll per word          -6.374021941608899
--- evaluate ---
| |   #                                                 | 159 Elapsed Time: 0:01:14
--- validate ---
LDA MRR:        0.1561320680286651
Time Lapsed = 0:1:54.462520122528076
```

У відповідь на це значення початкового числа не використовувалося в експериментах, щоб запобігти плутанині щодо передбачуваного, але неіснуючого детермінізму.

3.4.1. Ефективність результатів пошуку

На основі розуміння процесу визначення місця розташування функцій, отриманого з попереднього розділу, в цьому розділі оцінюється ефективність двох хороших моделей, однієї з латентного розміщення Діріхле (LDA) та однієї з розподілу Pachinko.

Для порівняння ефективності було обрано дві добре працюючі моделі, знайдені під час оцінки процесу визначення місця розташування функцій. Їх конфігурація наведена в таблиці 3.8.

Під час навчання вони працювали з логарифмічною правдоподібністю -6.3451 для моделі LDA та -10.2163 для моделі розподілу Pachinko.

Щоб дати деяке уявлення про те, як працювали тематичні моделі під абстракцією інструменту, в таблиці 3.9 показано 5 найкращих слів з однієї з тем кожної моделі. Вони поєднуються з їх відповідною ймовірністю з розподілу Діріхле, який описує їх зв'язок з темою.

З точки зору оцінки, ця таблиця показує, що більш комплексна попередня фільтрація слів може бути корисною для подальшого покращення результатів, які представлені в наступному розділі.

Таблиця 3.8.

Конфігурації LDA та PA, використані для оцінки ефективності результатів пошуку

Technique	Topics		Iterations	Burn-in	Alpha		Eta
LDA	350		100	10	0.01		0.01
PA	250	50	100	100	0.01	0.01	0.01

Таблиця 3.9.

Розподіл слів, пов'язаних з однією з тем/підтем моделей LDA та розподілу Pachinko, використаних для оцінки ефективності

LDA		PA	
Word	**Probability in topic**	**Word**	**Probability in subtopic**
code	0.0939	proposal	0.0183
deprecated	0.0641	quorumpacket	0.0158
keeperexception	0.0286	type	0.0138
link	0.0218	follower	0.0133
3.1.0	0.0203	qp	0.0094

3.4.2. Визначення остаточної ефективності

Для остаточної оцінки ефективності визначення місця розташування функцій, застосованого за допомогою методів, представлених у цій дисертації, раніше представлені моделі були перевірені на золотому наборі запитів та очікуваних результатів пошуку Корлі.

Слідуючи роботі попередніх досліджень, основна увага приділяється метриці середнього зворотного рангу (MRR), оскільки вона виражає ефективність пошукової системи. Додаткові метрики, розраховані з виводу валідації, - це середній та медіанний ранг, а також середнє абсолютне відхилення від середнього.

Хоча MRR LDA вищий (кращий) за відповідне значення для розподілу Pachinko, інші метрики, які базуються на ранзі, здається, сприяють новішому методу. Z-оцінка та p-значення, розраховані за допомогою знаково-рангового тесту Вілкоксона для рангу, додатково підтверджують результати статистично значущою різницею, припускаючи загальноприйнятий рівень значущості $\alpha = 0.05$.

Таблиця 3.10.

Метрики, розраховані під час оцінки ефективності результатів пошуку

Metric	LDA	PA
MRR	0.2068	0.1662
Mean rank	111	51
Avg. abs. deviation	102.6	59.1
Median rank	67	18
Z-score of the rank	5.78	
p-value of the rank	< 0.01	
Z-score of the reciprocal rank	1.35	
p-value of the reciprocal rank	0.18	

Як описано в другому розділі, p-значення має бути вищим за вибране значення α, щоб різниця була значущою. З іншого боку, p-значення для зворотного рангу не підтверджує таке припущення за того ж рівня значущості. Результати розрахунків наведено в таблиці 3.10, а проміжні значення для знаково-рангового тесту Вілкоксона наведено в таблиці 3.11.

Таблиця 3.11.

Проміжні значення знаково-рангового тесту Вілкоксона

Variable	*Rank*	*Reciprocal Rank*
T (smaller sum of signed ranks)	2715.5	5151.5
N (sample size)	153	153
μ	5890.5	5890.5
σ	548.9965847	548.9965847
Z	5.782367484	1.34518141
$\phi(\lvert Z \rvert)$	1.00E+00	0.91
p	7.36566E-09	0.178566658

На додаток до метрик, розподіл рангів результатів пошуку, що описує позицію першого правильного результату, було вилучено з процесу валідації та візуалізовано для подальшого обговорення. Результати показано на рисунку 3.11 та відображено за допомогою гістограми з 5 інтервалами та інтервалом переповнення для рангів вище 20.

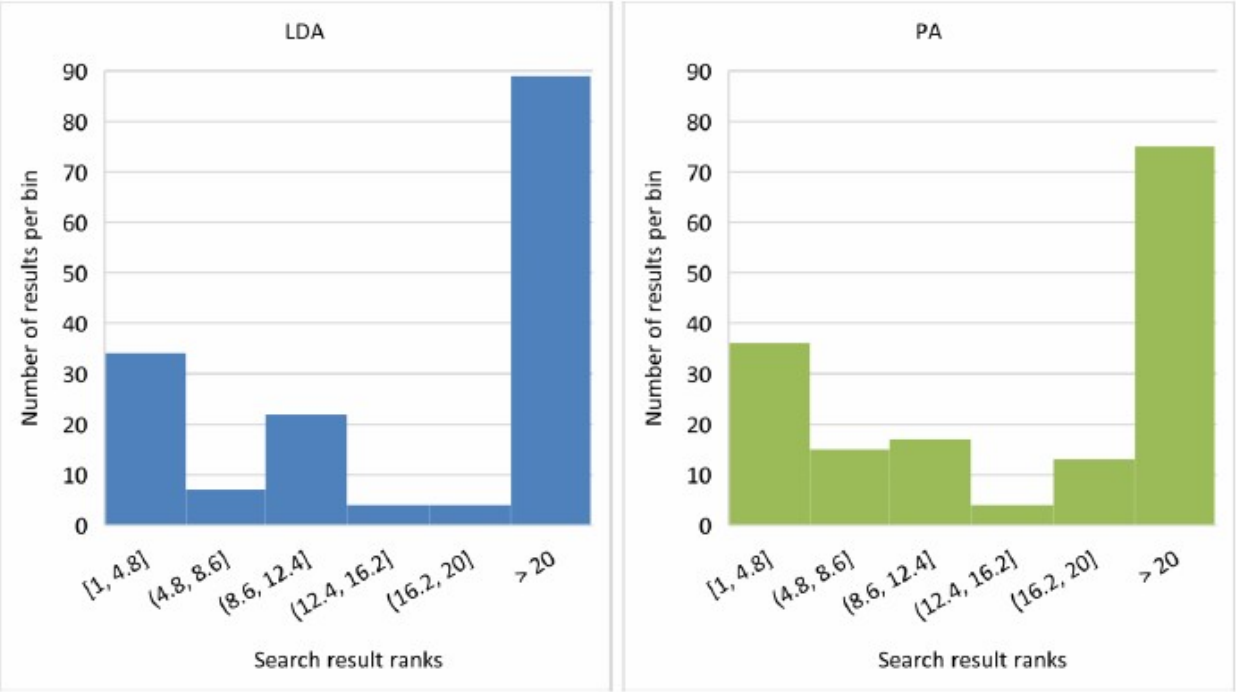


Рис. 3.11. Гістограма розподілу рангів результатів пошуку, вилучених з процесу валідації результатів визначення місця розташування функцій на основі моделей LDA та розподілу Pachinkod

Видно, що результати LDA включають більше рангів, які перевищують значення переповнення, і тому сортуються в останній інтервал. Також очевидно, що розподіл Pachinko згенерував більше результатів у двох інтервалах, що описують нижчі ранги результатів пошуку.

Висновки до розділу

Отже, в цьому розділі детально розглянуто процес імплементації методів та засобів тематичного моделювання функцій локації, а також

проаналізовано ключові аспекти їх реалізації. У рамках розробки загальної стратегії рішення визначено підхід до тематичного моделювання функцій локації, який поєднує методи структурного моделювання, машинного навчання та валідації результатів. Це дозволило створити інтегровану архітектуру для розв'язання поставлених завдань.

Було впроваджено модуль імпорту, який включає кілька структурних блоків: контекст і область застосування, модуль задач Jira, фіналізація даних. Такий підхід забезпечує гнучкість та універсальність у роботі з різними джерелами даних. Використання модулю Jira дозволило інтегрувати додаткові функції для управління задачами та зберігання результатів.

Розроблено та впроваджено алгоритм для визначення місця розташування функцій, що включає кілька ключових етапів. Запропонований підхід базується на комбінуванні методів кластеризації та тематичного аналізу, що дозволило досягти значної точності у визначенні місця розташування функцій.

Таким чином, у цьому розділі розроблено та впроваджено комплексне рішення для тематичного моделювання функцій локації, яке може бути адаптоване до різних областей застосування.

ВИСНОВКИ

Магістерська робота присвячена дослідженню та впровадженню тематичного моделювання як методу обробки природної мови (NLP) для визначення локаційного розташування функцій у документах.

У першому розділі проведено огляд предметної області тематичного моделювання та аналіз основних понять. Визначено проблему дослідження, яка полягає у складності визначення локаційного місця функцій у текстових документах. Запропоновано таксономію понять, що стосуються тематичного моделювання, та інструменти, які забезпечують його реалізацію. Описано процес видобування тексту (Text Mining) як ключового етапу у підготовці даних до тематичного моделювання. Узагальнено концепцію тематичного моделювання як потужного інструменту для пошуку інформації та автоматизованого аналізу тексту.

У другому розділі здійснено аналіз математичних моделей та методологій. Розглянуто Латентний розподіл Діріхле (LDA) та його варіації, як-от розподіл Пачінко, що дозволяють ефективно моделювати багатовимірні дані. Описано процес підготовки тексту: видобуток, очищення та попередня обробка даних. Вивчено параметри тематичних моделей та гіперпараметри, що впливають на якість кластеризації текстів. Розроблено підхід для визначення локаційного розташування функцій із використанням пошукових запитів, метрик продуктивності та валідації на основі золотого набору.

У третьому розділі представлено практичну реалізацію розробленої методології. Визначено загальну стратегію вирішення завдання, побудовану на інтеграції тематичного моделювання з сучасними інструментами управління даними, зокрема, модулем задач Jira. Реалізовано програму імпорту та обробки даних, включаючи фіналізацію результатів для точного визначення місця розташування функцій. Проведено навчання та валідацію моделей із використанням оптимальних налаштувань гіперпараметрів, що забезпечило високу точність аналізу. Описано перешкоди, які виникли під

час реалізації, зокрема, труднощі з оптимізацією пошукових запитів і підвищенням ефективності результатів.

Отже, в роботі вдалося сформулювати теоретичну основу для тематичного моделювання функцій локації, запропонувати нові підходи до обробки текстових даних та підвищення продуктивності моделей NLP та реалізувати прототип системи, яка може бути використана для вирішення практичних завдань у різних доменах.

ПЕРЕЛІК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. Buntine, W. L.; Mishra, S.: Experiments with non-parametric topic models. In (Macskassy, S. et al. Eds.): Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining. ACM, New York, NY, 2014; pp. 881–890.
2. Corley, C. S.; Damevski, K.; Kraft, N. A.: Changeset-Based Topic Modeling of Software Repositories. In IEEE Transactions on Software Engineering, 2020, 46; pp. 1068–1080.
3. Chochlov, M.; English, M.; Buckley, J.: Using changeset descriptions as a data source to assist feature location: 2015 IEEE 15th International Working Conference on Source Code Analysis and Manipulation (SCAM), 2015; pp. 51–60.
4. Chochlov, M.; English, M.; Buckley, J.: A historical, textual analysis approach to feature location. In Information and Software Technology, 2017, 88; pp. 110–126.
5. Chang, S. K.: Handbook of software engineering & knowledge engineering. World Scientific, River Edge, N.J., London, 2001.
6. Blei, D. M., Ng, A. Y., & Jordan, M. I. (2003). Latent Dirichlet Allocation. Journal of Machine Learning Research.
7. Hofmann, T. (1999). Probabilistic Latent Semantic Analysis. In Proceedings of UAI.
8. Yan, X., Guo, J., Lan, Y., & Cheng, X. (2013). A biterm topic model for short texts. WWW Conference Proceedings.
9. Alghamdi, R.; Alfalqi, K.: A Survey of Topic Modeling in Text Mining. In International Journal of Advanced Computer Science and Applications, 2015
10. Banerjee, A.; Basu, S.: Topic Models over Text Streams: A Study of Batch and Online Unsupervised Learning. In (Apté, C. V. Ed.): Proceedings of the

Seventh SIAM International Conference on Data Mining. Society for Industrial and Applied Mathematics, Philadelphia, 2007; pp. 431–436.

11. Blei, D. M.: Probabilistic topic models. In Communications of the ACM, 2012, 55; pp. 77–84.
12. Mikolov, T., Sutskever, I., Chen, K., Corrado, G. S., & Dean, J. (2013). Distributed Representations of Words and Phrases and Their Compositionality. Advances in Neural Information Processing Systems.
13. Wang, C., & Blei, D. M. (2011). Collaborative Topic Modeling for Recommending Scientific Articles. KDD Proceedings.
14. Sievert, C., & Shirley, K. (2014). LDAvis: A method for visualizing and interpreting topics. ACL Proceedings.
15. Dieng, A. B., Wang, C., Gao, J., & Paisley, J. (2020). Topic modeling in embedding spaces. Transactions of the Association for Computational Linguistics.
16. Gao, Y., Zhang, Y., & Zhao, B. (2019). Topic Modeling with Word Embedding Regularization. IEEE Transactions.
17. Boyd-Graber, J., Hu, Y., & Mimno, D. (2014). Applications of Topic Models. Foundations and Trends® in Information Retrieval.
18. Mohr, J. W., & Bogdanov, P. (2013). Introduction—Topic Models: What They Are and Why They Matter. Poetics.
19. Cheng, X., Yan, X., Lan, Y., & Guo, J. (2014). BTM: Biterm Topic Model for Short Texts. International Conference on Knowledge Discovery and Data Mining.
20. Kim, D., Lee, H., & Oh, A. (2015). Modeling Topic Flows in Microblog Posts. CIKM Proceedings.
21. Griffiths, T. L., & Steyvers, M. (2004). Finding scientific topics. Proceedings of the National Academy of Sciences.
22. Lau, J. H., & Baldwin, T. (2016). An Empirical Evaluation of doc2vec with Practical Insights into Document Embedding Generation. ACL Proceedings.

23. Mehrotra, R., et al. (2013). Improving LDA Topic Models for Microblogs via Tweet Pooling and Automatic Labeling. SIGIR Conference.
24. Alghamdi, R., & Alfalqi, K. (2015). A survey of topic modeling in text mining. International Journal of Advanced Computer Science and Applications.
25. Mimno, D., Wallach, H., Talley, E., Leenders, M., & McCallum, A. (2011). Optimizing Semantic Coherence in Topic Models. EMNLP Proceedings.
26. Tang, J., Meng, Z., Nguyen, X., Mei, Q., & Zhang, M. (2014). Understanding the Limiting Factors of Topic Modeling via Posterior Contraction Analysis. ICML Proceedings.
27. Zhao, W. X., et al. (2011). Comparing Twitter and Traditional Media Using Topic Models. ECIR Proceedings.
28. Hu, Y., Boyd-Graber, J., Satinoff, B., & Smith, A. (2014). Interactive Topic Modeling. Machine Learning Journal.
29. McAuliffe, J. D., & Blei, D. M. (2008). Supervised topic models. Advances in Neural Information Processing Systems.
30. Blei, D. M.; Ng, A. Y.; Jordan, M. I.: Latent Dirichlet Allocation. In J. Mach. Learn. Res., 2003, 3; pp. 993–1022.
31. Bourque, P. Ed.: Guide to the software engineering body of knowledge. Version 3.0 ; SWEBOK ; a project of the IEEE Computer Society. IEEE Computer Soc, Los Alamitos, Calif. [u.a.], 2014.
32. Chang, J., Gerrish, S., Wang, C., Boyd-Graber, J., & Blei, D. M. (2009). Reading tea leaves: How humans interpret topic models. NIPS Proceedings.
33. Ramage, D., Rosen, E., Chuang, J., Manning, C. D., & McFarland, D. A. (2009). Topic modeling for the social sciences. NIPS Proceedings.
34. Lin, C., & He, Y. (2009). Joint sentiment/topic model for sentiment analysis. CIKM Proceedings.
35. Blei, D. M. (2012). Probabilistic topic models. Communications of the ACM.

36. Chih-wei, H.; Chih-chung, C.; Chih-jen, L.: A practical guide to support vector classification, 2010.
37. Qiang, J., Qian, Z., Li, Y., Yuan, Y., & Wu, X. (2020). Short text topic modeling techniques, applications, and performance. ACM Transactions on Intelligent Systems and Technology.
38. Griffiths, T. L., & Steyvers, M. (2004). Finding scientific topics. PNAS.
39. Maier, D., et al. (2018). Applying LDA topic modeling in communication research: Toward a valid and reliable methodology. Communication Methods and Measures.
40. Röder, M., Both, A., & Hinneburg, A. (2015). Exploring the space of topic coherence measures. WSDM Proceedings.
41. Rosen-Zvi, M., Griffiths, T., Steyvers, M., & Smyth, P. (2004). The author-topic model for authors and documents. UAI Proceedings.
42. Yin, J., & Wang, J. (2014). A Dirichlet multinomial mixture model-based approach for short text clustering. KDD Proceedings.
43. Li, J., Sun, A., Han, J., & Li, C. (2020). A survey on deep learning for named entity recognition. IEEE Transactions.
44. Nguyen, V.-A., Boyd-Graber, J., Resnik, P., & Miler, J. (2014). Tea party in the house: A hierarchical ideal point topic model and its application to Republican legislators in the 112th Congress. ACL Proceedings.
45. Kim, Y., & Oh, A. (2011). Topic chains for understanding a news corpus. ACL Proceedings.
46. Gerlach, M., Peixoto, T. P., & Altmann, E. G. (2018). A network approach to topic models. Science Advances.
47. Kingma, D. P., & Welling, M. (2013). Auto-Encoding Variational Bayes. ICLR Proceedings.
48. Roberts, M. E., et al. (2014). Structural topic models for open-ended survey responses. American Journal of Political Science.
49. Zhang, Y., Jin, R., & Zhou, Z. (2010). Understanding bag-of-words model: A statistical framework. IJCAI Proceedings.

50. Zhao, W., et al. (2015). A neural topic model for document representation. Proceedings of the International Conference on Learning Representations.
51. Zhong, Y., & Ghosh, J. (2014). A scalable hierarchical topic model for large document collections. ACM Transactions on Knowledge Discovery from Data.
52. Egyed, A.; Binder, G.; Grunbacher, P.: STRADA: A Tool for Scenario-Based Feature-to-Code Trace Detection and Analysis: 29th International Conference on Software Engineering. ICSE 2007 companion volume proceedings 20-26 May 2007, Minneapolis, Minnesota. IEEE Computer Society, Los Alamitos CA, 2007; pp. 41–42.