НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ»

ННК "Інститут прикладного системного аналізу"
(повна назва інституту/факультету)

Кафедра Системного проектування
(повна назва кафедри)

«До захисту допущено»

Завідувач кафедри
_____ А.І.Петренко
(підпис) (ініціали, прізвище)

"___"_____2016 р.

Дипломна робота

першого (бакалаврського) **рівня вищої освіти**
(першого (бакалаврського), другого (магістерського))

зі спеціальності7.05010102, 8.05010102 Інформаційні технології проектування
7.05010103, 8.05010103 Системне проектування
(код та назва спеціальності)

на тему: Структурування знань з використанням технологій інтелектуального аналізу тексту

Виконав (-ла): студент (-ка) _4__ курсу, групи __ДА-21_____
(шифр групи)

Калюжний Максим Сергійович
(прізвище, ім'я, по батькові) (підпис)

Керівник _____Кисельова Анна Генадіївна, к.т.н., доцент _____
(посада, науковий ступінь, вчене звання, прізвище та ініціали) (підпис)

Консультант _____ _____ _____
(назва розділу) (посада, вчене звання, науковий ступінь, прізвище, ініціали) (підпис)

Рецензент _____ _____
(посада, науковий ступінь, вчене звання, науковий ступінь, прізвище та ініціали) (підпис)

Нормоконтроль _____ст. викладач Британ О.А._____

Засвідчую, що у цій дипломній роботі немає запозичень з праць інших авторів без відповідних посилань.
Студент _____
(підпис)

Київ – 2016 року

Національний технічний університет України
«Київський політехнічний інститут»

Факультет (інститут) ННК "Інститут прикладного системного аналізу"
(повна назва)

Кафедра Системного проектування
(повна назва)

Рівень вищої освіти Перший(Бакалаврський)
(перший (бакалаврський), другий (магістерський) або спеціаліста)

Спеціальність 7.05010102, 8.05010102 Інформаційні технології проектування 7.05010103, 8.05010103 Системне проектування
(код і назва)

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ А.І.Петренко
(підпис) (ініціали, прізвище)
«___»_____2016 р.

ЗАВДАННЯ
на дипломний проект (роботу) студенту
Калюжному Максиму Сергійовичу
(прізвище, ім'я, по батькові)

1. Тема проекту (роботи)) Структурування знань з використанням технологій інтелектуального аналізу тексту
керівник проекту (роботи)) Кисельова Анна Генадіївна, к.т.н., доцент ,
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом по університету від 12 травня 2016 р. № 50-ст
2. Строк подання студентом проекту (роботи) 12.06.2016
3. Вихідні дані до проекту (роботи) _____

Набори текстів у форматі txt
Мова текстів - англійська
4. Зміст розрахунково-пояснювальної записки (перелік завдань, які потрібно розробити)

1. Розглянути можливі етапи інтелектуального аналізу тексту.
2. Розглянути теоретично алгоритми для кожного з етапів.
3. Обрати алгоритми для реалізації.

4. Розробити алгоритми для кожного з етапів.
5. Скласти набори текстів для тестування системи та збору даних для подальшого аналізу ефективності алгоритмів.
6. Провести аналіз результатів.

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслеників, плакатів тощо)

1. Блок-схема алгоритму роботи системи – плакат.
2. Таблиця результатів оцінки ефективності пошуку ключових слів – плакат.
3. Архітектура системи - плакат.
4. Приклади роботи програми – плакат.

6. Консультанти розділів проекту (роботи)*

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Економічний розділ			
Основна частина			

7. Дата видачі завдання 01.02.2015

Календарний план

з/п	Назва етапів виконання дипломного проекту (роботи)	Строк виконання етапів проекту (роботи)	Примітка
	Отримання завдання	01.02.2016	
	Збір інформації	15.02.2016	
	Вивчення варіантів реалізації та вибір варіанту для розробки	28.02.2016	
	Розробка алгоритмів та структури системи	10.03.2016	
	Розробка плану тестування	15.03.2016	
	Розробка программи	25.03.2016	
	Розробка опису системи	25.04.2016	
	Тестування системи та аналіз результатів	30.04.2016	
	Оформлення дипломної роботи	05.06.2016	
	Отримання допуску до захисту та подача роботи в ДЕК	12.06.2016	

Студент _____ М.С. Калюжний
(підпис) (ініціали, прізвище)

Керівник проекту (роботи) _____ А.Г. Кисельова
(підпис) (ініціали, прізвище)

* Консультантом не може бути зазначено керівника дипломного проекту (роботи).

АНОТАЦІЯ

до бакалаврської дипломної роботи Калюжного Максима Сергійовича

на тему «Структурування знань з використанням технологій інтелектуального аналізу тексту»

Дипломна робота присвячена дослідженню методів та засобів інтелектуального аналізу тексту, що застосовуються для структурування знань. Розглянуті та програмно реалізовані методи попередньої обробки тексту, виділення ключових слів та класифікації документів. Було проведено дослідження ефективності методів на основі зібраної статистики з точки зору повноти та точності.

Загальний об’єм роботи 99 сторінок, 10 рисунків, 20 таблиць, 19 посилань та 2 додатки на 19 сторінок.

Ключові слова: інтелектуальний аналіз тексту, text mining, стемінг, стоп-слова, ключові слова, TF-IDF, лексико-статистичні шаблони, LSPL, метод Роше, метод Наївного Байеса, Y-інтерпретація закону Бредфорда, класифікація тестів, кластеризація текстів.

АННОТАЦИЯ

к бакалаврской дипломной работе Калюжного Максима Сергеевича

на тему: «Структурирование знаний с использованием технологий интеллектуального анализа текста»

Дипломная работа посвящена исследованию методов и средств интеллектуального анализа текста, применяемые для структурирования знаний. Рассмотрены и реализованы программно методы предварительной обработки текста, выделение ключевых слов и классификации документов. Было проведено исследование эффективности методов на основе собранной статистики с точки зрения полноты и точности.

Общий объем работы 99 страниц, 10 рисунков, 20 таблиц, 19 ссылок и 2 дополнения на 19 страниц.

Ключевые слова: интеллектуальный анализ текста, text mining, стемминг, стоп-слова, ключевые слова, TF-IDF, лексико-статистические шаблоны, LSPL, метод Роше, метод наивного Байеса, Y-интерпретация закона Брэдфорда, классификация тестов, кластеризация текстов.

ABSTRACT

of a bachelor’s degree work, which made by Kaliuzhnyi Maxym Serhiyovych

on theme: «Structuring knowledge using text mining»

Thesis is devoted to research of methods of text mining, which are developoed to structure knowledge. Considered and software implemented methods pretreatment text, keyword selection and classification of documents. There have been studies of the effectiveness of methods based on statistics collected in terms of completeness and accuracy.

The total volume of work is 99 pages, 10 illustrations, 20 tables, 19 references and 2 additions on 19 pages.

Keywords: text mining, stemming, stop words, keywords, TF-IDF, lexical and statistical templates, LSPL, method Roche, Naive Bayes method, Y-law interpretation of Bradford, classification tests, clustering texts.

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ

ПП – програмний продукт

ПМ – природна мова

ФВА – функціонально-вартісний аналіз

NN – singular common noun

NNP – proper noun

AJ – general adjective

DT – general determiner

NLP – natural language processing

TF-IDT – term frequency – inverse document frequency

ВСТУП

На даному етапі розвитку технологій була спродукована велика кількість інформації. Цей обсяг зростає експоненціально. За даними міжнародної дослідницької і консалтингової компанії International Data Corporation ця кількість у 2007 році становила приблизно 161 ексабайт (161 * 10^18 байт або 161 * 10^10 Гігабайт). Станом на 2010 рік це число зросло в 6.2 рази, тобто досягло відмітки у 998 ексабайт.

Головною проблемою у максимально повному використанні таких обсягів даних є те, що лише 10% даних є структурованими, а решта, головним чином текст – є неструктурованими. Споживачеві інформаційного продукту доводиться довго шукати корисну і значущу інформацію, методом ручного перебору. Раніше цю проблему вирішували ручним способом структуризації та класифікації, як, наприклад, Карл Лінней у 18 столітті, але, враховуючи сьогоденний обсяг інформації та динаміку його росту, постала необхідність у розробці технологій, які виконували б вищезазначені процеси автоматично.

Актуальність теми даної роботи обумовлена наявністю нагальної потреби дослідження засобів та методів інтелектуального аналізу текстів, що було проілюстровано вище, а також необхідністю проаналізувати, структурувати, порівняти підходи та алгоритми на кожному з етапів Text mining.

Метою даної роботи є аналіз методів Text mining, дослідження існуючих реалізацій як окремих алгоритмів, так і програмних систем, які були створені для структурування інформації, а також практичне порівняння роботи алгоритмів інтелектуального аналізу тексту (Text Mining).

Інформаційна система, яка досліджується та розробляється в даній роботі має автоматично виділяти і структурувати знання і поняття з сукупності текстів, які без згенерованих цією системою метаданих являють собою лише послідовність закодованих за одним із текстових стандартів символів. Знання у

даному контексті являють собою структуровану систему з понять та зв’язків між ними. А кожне поняття з сукупності текстів, кожен з яких розкриває його з тієї чи іншої сторони.

За загальним алгоритмом, система, яка має проводити семантичний аналіз тексту, включає в себе наступні етапи:

- пошук інформації
- попередня обробка тексту;
- пошук ключових слів (колокацій, сталих конструкцій);
- класифікації, кластеризації;
- інтерпретація результатів;

У першому розділі означена категорія знань, предметна область та дані основні визначення. А також проаналізовані базові підходи, розглянуті основні алгоритми, розроблені для виконання вищезазначених задач. Був проведений аналіз наукових джерел.

У другому розділі висвітлено готові бібліотеки з API та програмні продукти, що реалізують ті чи інші підходи до Text Mining.

У третьому розділі представлено результати роботи методів структурування даних, порівняльна статистика цих методів на наборах текстів.

У четвертому розділі проведено функціонально-вартісний аналіз програмного продукту.

1. АНАЛІТИЧНИЙ ОГЛЯД ТЕХНОЛОГІЙ: ЕТАПИ, АЛГОРИТМИ

1.1 Основні задачі

Спостерігається так званий «інформаційний вибух». Тобто проблема пошуку джерел інформації, яка стояла перед людиною раніше перетворилася на проблему вибору або фільтрації необхідної інформації серед інформаційного шуму. Основну частину всієї корисної інформації на Землі складають тексти.

Постала проблема створення систем з пошуку знань та структуруванні знань в текстах. Для вирішення даної задачі виник напрямок інтелектуального аналізу текстів (англомовна назва Text Mining). Його відносять до Data Mining, він також включає в себе методи статистики, лінгвістики та штучного інтелекту. Також існує термін «Комп'ютерна лінгвістика».

Виявлення знань в тексті - це нетривіальний процес виявлення дійсно нових, потенційно корисних і зрозумілих шаблонів в неструктурованих текстових даних.

Як видно, з визначення Data Mining воно відрізняється тільки новим поняттям "неструктуровані текстові дані". Під такими знаннями розуміється набір документів, що представляють собою логічно об'єднаний текст без будь-яких обмежень на його структуру. Прикладами таких документів є: Web-сторінки, електронна пошта, нормативні документи і т. п. У загальному випадку такі документи можуть бути складними і великими та включати в себе не тільки текст, а й графічну інформацію. документи, які використовують мову розширюваної розмітки XML (eXtensible MarkupLanguage), стандартна мова узагальненої розмітки SGML (Standard Generalised Markup Language) і інші подібні структурою тексти, прийнято називати напівструктурованими документами. Вони також можуть бути оброблені методами Text Mining.

В даний час в літературі описано багато прикладних задач, що вирішуюся шляхом застосування методів аналізу текстових документів. Це і класичні задачі Data Mining: класифікація, кластеризація, і характерні тільки для текстових документів завдання: автоматичне анотування, витяг ключових понять і ін.

Класифікація (classification) - стандартна задача з області Data Mining. Її метою є визначення для кожного документа однієї або декількох заздалегідь заданих категорій, до яких цей документ відноситься. особливістю задачі класифікації є припущення, що сукупність документів, що класифікуються, не містить сторонніх тем, тобто кожен з документів відповідає який-небудь заданої категорії.

Окремим випадком задачі класифікації є завдання визначення тематики документа.

Метою кластеризації (clustering) документів це автоматичне виявлення груп семантично схожих документів серед заданої фіксованої множини. Відзначимо, що групи формуються тільки на основі попарної схожості описів документів, і ніякі характеристики цих груп не задаються заздалегідь.

Автоматичне анотування (summarization) дозволяє скоротити текст, зберігаючи його зміст. Вирішення цього завдання зазвичай регулюється користувачем за допомогою визначення кількості речень або відсотком тексту, що виділяється, по відношенню до всього тексту. Результат включає в себе найбільш значимі речення в тексті.

Первинною метою **виділення ключових понять (feature extraction)** є ідентифікація фактів і відношень в тексті. У більшості випадків такими поняттями є іменники і загальні назви: імена і прізвища людей, назви організацій і ін. Алгоритми вилучення понять можуть використовувати словники, щоб ідентифікувати деякі терміни і лінгвістичні шаблони для визначення інших.

Навігація по тексту (text-base navigation) дозволяє користувачам переміщатися по документам відносно тем і значущих термінів. це виконується за рахунок ідентифікації ключових понять і деяких відношень між ними.

Виявлення (detection) дозволяє виявиити в потоці даних цікавих нових знань, таких як моделі, конструкції, асоціації, зміни, аномалії і структурні новоутворення.

Системи автоматичних відповідей (automatical answers systems) дає відповіді на запитання користувачів, що задаються на природній мові, замислювалися ще на зорі кібернетики.

За базу знань в цих системах передбачається використовувати ресурси Internet, оброблені сучасними засобами глибинного аналізу текстів.

Сьогодні вже створені та вдосконалюються реально працюючі систему, здатні відповідати на запитання користувачів. Це такі системи, як Siri, Viv, Google Speech search та інші. Роботи в цьому напрямку ведуться в дослідницьких центрах корпорацій.

Відповідно до розроблених алгоритмів, спочатку аналізується структура питання, визначається підмет (об'єкт пошуку), перетворюється питання в пошуковий запит, останній вирушає на звичайний пошуковик, отримують результати, а потім шукають необхідні слова серед знайдених сторінок і видають відповідь.

Розробляються ще більш складні системи з використанням штучного інтелекту, які могли б давати не односкладові, а розгорнуті варіанти відповідей до десятка слів.

1.2 Вимоги до системи

1. **Точність (accuracy)** - алгоритми обробки природної мови НЕ гарантують отримання лише коректних результатів, так що дизайн системи повинен

враховувати це і надавати можливості для підвищення точності за рахунок, наприклад, відкату до використання інших алгоритмів;

2. **Ефективність (efficiency)** - в більшості досліджень питання ефективності практично розглядаються як деталі реалізації, однак при інтерактивній роботі затримки у відповіді системи більш ніж на кілька секунд можуть бути неприйнятними для користувачів;

3. **Продуктивність (productivity)** - зусилля, витрачені на розробку ПМ-додатків часто вище, ніж для багатьох інших областей розробки ПЗ в зв'язку з відсутністю знань про існуючі ресурси, неможливості інтегрувати сторонні компоненти;

4. **Гнучкість (flexibility)** - як і інше програмне забезпечення ПМ-системи повинні бути гнучкими, вони повинні підтримувати різні формати, джерела даних та можливість застосування для різних завдань;

5. **Стійкість (robustness)** - системи повинні зберігати працездатність в різних умовах, містити обробку різних виняткових ситуацій і помилок алгоритмів;

6. **Масштабованість (scalability)** - для можливості обробки великих обсягів даних системи повинні мати можливість збільшення продуктивності наприклад, за рахунок використання розподіленої схеми;

7. **Мультимодальність (multimodality)** - різні лінгвістичні компоненти використовують при аналізі різні аспекти інформації, відповідно система повинна підтримувати можливість доступу до них;

8. **Розрідженість даних (data sparseness)** - оскільки багато алгоритми обробки ПМ-текстів вимагають наявності підготовлених даних для навчання, їх побудова (особливо для багатомовних випадків) може представляти проблему;

9. **Багатомовність (multilinguality)** - користувачі говорять на різних мовах і, відповідно, як в ПО необхідна інтернаціоналізація, підтримка відповідних

кодувань, проте в для ПМ-систем ситуація значно ускладнюється тим, що для кожної мови необхідні свої набори тренувальних даних, правил обробки, а крім того, можуть відрізнятися використовувані алгоритми.

1.3 Загальні етапи та підходи

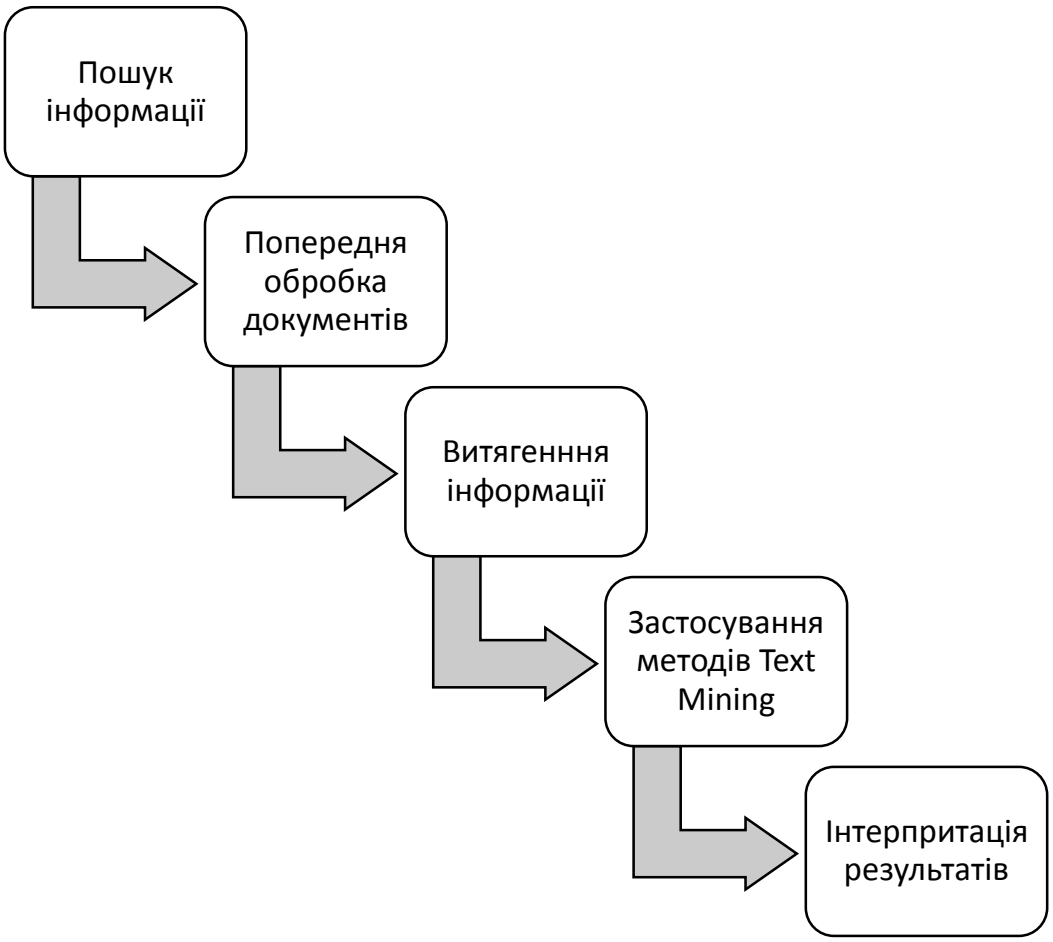

Рисунок 1.1 – Етапи Text Mining

Процес аналізу текстових документів можна уявити як послідовність декількох кроків (рис. 1.1).

1. Пошук інформації. На першому кроці необхідно ідентифікувати, які документи повинні бути проаналізовані і забезпечити їх доступність. Як правило, користувачі можуть визначити набір документів, що аналізуються

самостійно - вручну, але при великій кількості документів необхідно використовувати варіанти автоматизованого відбору за заданими критеріями.

2. Попередня обробка документів. На цьому кроці виконуються найпростіші, але необхідні перетворення з документами для ставлення їх у вигляді, з яким працюють методи Text Mining. метою таких перетворень є видалення зайвих слів і надання тексту більш строгої форми. Детальніше методи попередньої обробки будуть описані далі.

3. Витягнення інформації. Витяг інформації з обраних документів передбачає виділення в них ключових понять, над якими надалі буде виконуватися аналіз. Даний етап є дуже важливим і буде докладно описаний далі.

4. Застосування методів Text Mining. На даному етапі витягуються шаблони і відношення, наявні в текстах. Даний крок є основним в процесі аналізу текстів, і практичні завдання, які вирішуються на цьому шаге, описуються в далі.

5. Інтерпретація результатів. Останній крок у процесі виявлення знань передбачає інтерпретацію отриманих результатів. як прави ло, інтерпретація полягає або в поданні результатів на природньою мовою, або в їх візуалізації в графічному вигляді.

Візуалізація також може бути використана як засіб аналізу тексту. Для цього беруться ключові поняття, які і подаються в графічному вигляді. Такий підхід допомагає користувачеві швидко ідентифікувати головні теми і поняття, а також визначити їх важливість.

1.4 Попередня обробка

Однією з основних проблем аналізу текстів є велика кількість допоміжних (незначущих) слів, а також різні форми слів в одному документі. Якщо

аналізувати кожне таке слово, то час пошуку нових знань різко зросте і навряд чи задовольнятиме вимоги користувачів. У той же час очевидно, що не всі слова в тексті несуть корисну інформацію. Крім того, в силу гнучкості природних мов формально різні слова (синоніми і т. п.) насправді означають однакові поняття. Таким чином, видалення неінформативних слів, а також приведення близьких за змістом слів до єдиної формі значно скорочують час аналізу текстів. Усунення описаних проблем виконується на етапі попередньої обробки тексту.

Зазвичай використовують такі прийоми видалення неінформативних слів і підвищення строгості текстів:

- **видалення стоп-слів**. Стоп-словами називаються слова, які є допоміжними і несуть мало інформації про зміст документа. Зазвичай заздалегідь складаються списки таких слів, і в процесі попередньо котельної обробки вони видаляються з тексту. Типовим прикладом таких слів є допоміжні слова і артиклі, наприклад: "так як", "Крім того" і т. п.;
- **стеммінг** - морфологічний пошук. Він полягає в перетворенні кожного слова до його нормальної формі. Нормальна форма виключає схиляння слова, множинну форму, особливості усного мовлення і т. п. Наприклад, слова "стиснення" і "стислий" повинні бути перетворені в нормальну форму слова "стискати". Алгоритми морфологічного розбору враховують мовні особливості і внаслідок цього є мовно-залежними алгоритмами;
- **N-грами** - це альтернатива морфологічному розбору і видалення стоп-слів. N-грами - це частина рядка, що складається з N символів. наприклад, слово "дата" може бути представлено 3-граммой "_Да", "дат", "Ата", "та_" або 4-граммой "_дат", "дата", "ата_", де символ підкреслення замінює попередній або замикає слово пробіл. Порівняно зі стеммінгом або видаленням стоп-слів, N-грами менш чутливі до граматичних та орфографічних помилок. Крім того, N-грами не вимагають лінгвістичного подання слів, що робить даний прийом більш

незалежним від мови. Однак N-грами, дозволяючи зробити текст більш суворим, не вирішують проблему зменшення кількості неінформативних слів;

- **приведення регістра**. Цей прийом полягає в перетворенні всіх символів до верхнього або нижнього регістру. Наприклад, все слова "текст", "Текст", "ТЕКСТ" наводяться до нижнього регістру "текст". Проте це не завжди так. Наприклад, коли слово пишеться з великої літери, система може розпізнати його як власну назву чи ім’я.

Найбільш ефективно спільне застосування цих методів.

1.5 Методи видалення стоп слів

У даній роботі буде реалізовано та проаналізовано 3 методи видалення стоп-слів: зі словником, статистичний на основі об’єднання слів із різних текстів та на основі Y-інтерпретації закону Бредфорда.

1.5.1 Словниковий

Даний метод полягає у використанні готового переліку стоп-слів. Список стоп-слів складається вручну, що є першим недоліком даного методу, адже цей процес є трудоємким. Другим недоліком можна виділити невелику універсальність словника, що веде за собою недостатньо повне видалення стоп-слів, так як цей перелік є дуже великим і для різних типів текстів.

Перевагою даного методу є точність, тобто використання даного методу не видалить слова, які несуть смислове навантаження в тексті.

1.5.2 Статистичний на основі об’єднання

Метод полягає в аналізі великої кількості текстів з предметної області, до якої належить досліджуваний текст, і вибірка з цих текстів списку найбільш слів, що зустрічаються найчастіше. Перевага такого методу полягає в автоматизації і в тому, що словник стоп-слів необхідно створити один раз і надалі не потрібно створювати його для кожного тексту і витрачати на це обчислювальний час. Але при цьому не враховуються особливості тексту. Наприклад, може виникнути ситуація, при якій з тексту будуть вилучені слова, які згідно зі статистикою опинилися в словнику стоп-слів, але для даного тексту ці слова несуть смислове навантаження і їх видалення приведе до зміни змісту тексту і, як наслідок, неправильних результатів.

1.5.3 За Y-інтерпретацією закону Бредфорда

Закон Бредфорда - емпіричне правило розподілу публікацій з виданням, згідно з якою в списку наукових журналів, розташованих в порядку убування числа статей по заданому питанню, можна виділити три зони, що містять рівне число статей по заданому питанню.

Ці три зони розрізняються кількістю і якістю складових їх журналів:

- В першу зону (зону ядра) входять профільні журнали, безпосередньо присвячені заданому питанню;

- В другу зону входять журнали, частково присвячені заданому питанню;

- В найчисленнішу третю зону входять журнали, тематика яких далека від заданого питання.

Згідно із законом Бредфорда для кожної тематичної області існує коефіцієнт кратного збільшення кількості журналів в кожної наступної зони.

Математичне формулювання даного закону виглядає наступним чином:

$$S_1 + qS_1 + q^2S_1 = S \qquad (1)$$

Де: S_1 – перша зона (зона «ядра»);

qS_1 – друга зона

q^2S_1 – третя зона

S – загальна кількість публыкацій

Y-інтерпретація закону Бредфорда - інтерпретація, запропонована В.А. Яцко, що дозволяє обчислювати порогові рівні при виділенні підмножин зі складу безлічі на основі методики зонального аналізу [3, с. 30].

Що стосується аналізу тексту суть методу полягає в поділі тексту на 3 зони [3, с. 30]:

1. Зона J0 - зона службових слів або стоп-слова.

2. Зона J1 - слова, що представляють основний зміст тексту.

3. Зона J2 - слова, що рідко зустрічаються в тексті.

1.6 Ключові слова

Ключові слова - це одно- і багатокомпонентні лексичні групи, що відображають зміст документа. Автоматичне виділення ключових слів є необхідним етапом обробки тексту в таких важливих додатках, як системи автоматичного інформаційного пошуку, анотування, реферування і т. д. Однак, незважаючи на досить велику кількість досліджень, щоб автоматично завантажувати ключових слів являє собою проблему, яка до сих пір не вирішена. Проблематичним є автоматичне вилучення багатокомпонентних ключових слів, особливо, якщо робиться спроба автоматично отримати певні типи лексичних груп, наприклад, іменні групи. При всіх методиках алгоритм верхнього рівня виділення ключових слів універсальний і включає етапи: а) формування

«кандидатів» в ключові слова і б) фільтрації цієї множини для отримання результуючого списку ключових слів.

Виділяють 3 групи методів виділення ключових слів: статистичні, лексичні та гібридні.

1.6.1 Статистичні методи виділення ключових слів

Статистичні методи ґрунтуються на численних даних про частоту зустрічання слова в тексті.

В літературі відзначається, що перевагами статистичних методів є універсальність алгоритмів вилучення ключових слів, відсутність необхідності в трудомістких процедурах побудови лінгвістичних баз знань, простота реалізації. Але статистичні методи часто не забезпечують задовільної якості результатів. При цьому область ефективного застосування статистичних моделей обмежена мовами з бідної морфологією; як правило, є проблеми для природних мов з багатою морфологією, зокрема, для російської мови.

Класичними підходами в галузі статистичної обробки природної мови можна вважати використання метрики TF-IDF і її модифікацій (для виділення ключових слів), і аналіз колокацій (для виділення словосполучень).

TF-IDF - статистична міра, яка використовується для оцінки важливості слова, як в контексті документа (TF), так і в контексті корпусу документів (IDF). Метрика використовується з різними варіантами обчислення TF і IDF. Відомі різні розширення моделі TF-IDF, наприклад Okapi BM25. Існує величезна кількість досліджень і розробок в цій галузі. Відзначимо важливу особливість використання TF-IDF - набір даних не повинен змінюватися під час розрахунку. Це ускладнює обчислення, якщо потрібно провести обрахування даних в реальному часі.

Оцінка важливості слова за даним методом визначається за наступними формулами:

TF (term frequency — частота слова):

$$tf(t,d) = \frac{n_i}{\sum_k n_k} \qquad (2)$$

Де: n_i – число входжень слова в документ

$\sum_k n_k$ – загальна кількість слів у документі

IDF (inverse document frequency —обернена частота документа)

$$idf(t,D) = \log \frac{|D|}{|d_i \ni t_i|} \qquad (3)$$

Де: $|D|$ – кількість документів в корпусі

$|d_i \ni t_i|$ – кількість документів, у яких зустрічається термін t_i,

тобто $n_i \neq 0$

Таким чином міра TF-IDF є добутком двох попередніх оцінок:

$$tf - idf(t,d,D) = tf(t,d) \times idf(t,D) \qquad (4)$$

Є й інші статистичні підходи для виділення термінологічних сполучень. Наприклад, один з варіантів полягає в знаходження n-слівних поєднань по заданих частотних характеристиках. Це можуть бути значення абсолютних або відносних частот для даних словосполучень або значення деякої статистичної міри, згідно з якою дана конструкція була знайдена і видана серед результатів. Далі може бути використаний поріг відсікання по заданому значенню.

1.6.2 Лексичні методи виділення ключових слів

Спроби створення універсального лінгвістичного методу виділення ключових слів не мали успіху. Різні методи і прийоми можна класифікувати за певними лінгвістичним напрямками.

Лінгвістичні методи ґрунтуються на значеннях слів, використовують онтології і семантичні дані про слово. На жаль, ці методи занадто трудомісткі на ранніх етапах: розробка онтологій є дуже затратним процесом. До того ж, операції лінгвістичного аналізу текстів, що виконуються вручну, є джерелом значної кількості помилок і неточностей і роблять сам процес аналізу документів складним і тривалим. Тому необхідною умовою є наявність доступних програмних засобів, що дозволяють автоматизувати процес аналізу текстів документів.

Прикладом лінгвістичних методів вилучення ключових слів є методика аналізу текстів документів, що є основою автоматизованої інформаційної системи аудиту нормативних документів організації, що включає два етапи: дослідний і аналітичний. На дослідному етапі відбувається виділення ключових слів з тексту з використанням словників (словник неінформативних лексичних одиниць, словник сталих словосполучень та ін.) І операцій над множинами. Отримані ключові слова є вихідними даними для другого (аналітичного) етапу, метою якого є формулювання рекомендацій щодо оптимізації бізнес-процесів і електронних документопотоків.

Існує метод автоматичного вилучення ключових термінів з текстових документів, заснований на міру семантичної близькості термінів, обчисленої з використанням бази даних Вікіпедії, побудові семантичного графа, виборі тематичних термінів за допомогою алгоритму Гірвана-Ньюмана. Одним з переваг цього методу, на думку авторів, є відсутність необхідності в попередньому навчанні, так як працює безпосередньо з базою даних. Експериментальні оцінки ефективності методу показують високу точність і повноту вилучення з тексту ключових термінів. Лінгвістичні методи, засновані на застосуванні використання значень слів, словників, онтологій, енциклопедій, в тому числі, Вікіпедії, автори даної статті пропонують виділити в окремий метод на основі баз даних і значень слів.

Графові лінгвістичні методи представляють великий інтерес в області обробки природної мови, завдяки своїй універсальності і ефективності заснованих на них алгоритмів. У цих методах основною процедурою є побудова семантичного графа. Це зважений граф, вершинами якого є терміни документа, наявність ребра між двома вершинами свідчить про те, що терміни семантично пов'язані між собою, вага ребра є чисельним значенням семантичної близькості двох термінів. Ключові слова відбираються алгоритмами обробки графа. Графові методи розрізняються між собою способами відбору безлічі термінів і визначення близькості окремих термінів, які засновані на статистичних параметрах, а також на морфологічному, синтаксичному або семантичному аналізі. Взагалі кажучи, графові методи вбирають в себе безліч підходів, деякі автори вважають їх гібридними.

Використовуються лінгвістичні методи, засновані на маркемному аналізі. Такий аналіз визначає, в яких відношеннях перебувають та інтуїтивно виділяються ключові слова і маркеми. Це дозволяє досліджувати можливість автоматичного виділення ключових слів. Пропонована методика використовує доступне програмне забезпечення. Робляться висновки, що запропонований алгоритм маркемного аналізу наукових текстів цілком здатний давати достовірну інформацію про семантику цих текстів.

В описуються результати експериментального дослідження процедур автоматичного виявлення термінів в текстах, засновані на лексико-синтаксичних шаблонах мови LSPL. Такий шаблон задає послідовність елементів-слів, з яких повинна складатися мовна конструкція, і вказує умови синтаксичного узгодження цих елементів. Розроблено процедури виявлення вживання термінів на основі набору лексико-синтаксичних шаблонів. Запропонована стратегія спільного застосування цих процедур, дозволить підвищити F-міру повноти і точності розпізнавання.

У праці [2] виділені такі шаблони, наведемо приклади цих шаблонів:

Таблиця 1.1 – Приклади лексико-синтаксичних шаблонів

№	Групи шаблонів	Приклади шаблонів	Приклади термінів за шаблонами
1	Морфо-синтаксичні зразки термінів	N1 (N1)	тригер
		A1 N1 <A1=N1> (N1)	Незміщена оцінка
		N1 N2<c=gen> (N1)	Період напіврозпаду
2	Контексти визначення авторських термінів	Defin <c=acc> "будемо" "називати" Term<c=ins> #Term<c=nom>	Такі вектори будемо називати власними векторами
		"под" Term<c=ins> "розуміється" Defi n<c=nom> #Term<c=nom>	Під інкапсуляцією розуміється
3	Контексти введення синонімів термінів	Term1 "("Term2")" <Term1.c=Term2.c> #Term1<c=nom>, Term2<c=nom>	Перезапис памя'ті (тригерів)
		Term1 "," "чи" Term2 <Term1. c=Term2.c> #Term1<c=nom>, Term2<c=nom>	Розряднісю, чи довжиною слова
4	Словникові терміни	N1<вектор> [N2<намагниченности,c=gen>| N2<стану,c=gen>|"Умова"]	Вектор, вектор намагніченості, векотор Умова
		A1<бітовый > {N2<массив>| N2<образ>}<1,1> <A1=N2>	Бітовий вектор, бітова карта
5	Лексико-синтаксичні варіанти	N1 N2<c=gen> #N1, N1 N4<c=gen> <Syn(N2,N4)>, N3 N2<c=gen> <Syn(N1,N3)>, A1 N1 <A1.st=N2.st>	Перевизначення методів – перевизначення (N1)

Таблиця 1.1 (продовження)

№	Групи шаблонів	Приклади шаблонів	Приклади термінів за шаблонами
6	Об’єднання термінів	A1 A2 N1 <A1=A2=N1> #A1 N1, A2 N1	Розрядніть внутрішніх регістрів – розрядність регістра, внутрішній регістр

1.6.3 Гібридні методи виділення ключових слів

У попередніх підрозділах розглянуто виділення ключових слів і словосполучень, засноване на лінгвістичних або статистичних методах. Було зазначено, що кожен з цих методів має свої недоліки, і лише при їх комбінації досягається найбільша точність вилучення.

Проте великим потенціалом володіють гібридні методики, в яких статистичні методи обробки документів доповнюються однієї або декількома лінгвістичними процедурами і лінгвістичними базами знань різної глибини. Розроблені методи автоматичного вилучення двуслівних термінів з окремого тексту або корпусу текстів на основі статистики виявлення й морфологічних шаблонів. Показано, що використання сукупності ознак словосполучень значно покращує витяг термінів.

На основі порівняння існуючих систем автори статті [13] роблять висновок, що для поліпшення вилучення термінів і отримання більш релевантних результатів (зменшення інформаційного шуму), повинні бути виконані наступні умови: проведені лінгвістично орієнтовані дослідження семантичних зв'язків

термінів і умов обмеження термінологічних одиниць в межах даної спеціальної області і в даному текстовому типі; програмні системи повинні навчитися поєднувати статистичні та лінгвістичні методи і підтримувати більше однієї стратегії. Також повинна бути корисною розробка загальної шкали тестування і оцінки порівняння якості видобутих термінів.

Стаття [8] представляє результати дослідження по виділенню термінологічних словосполучень на основі статистичних заходів та граматики синтаксичних конструкцій за допомогою спеціалізованої системи обробки корпусних даних. Описуються експерименти з автоматичного виділення термінів і термінологічних сполучень. Оцінюється ефективність різних підходів. Фактично цей підхід є комбінованим, так як об'єднує лінгвістичний і статистичний методи.

1.7 Лінгвістична розмітка

Одним з методів представлення лінгвістичних даних є лінгвістична розмітка.

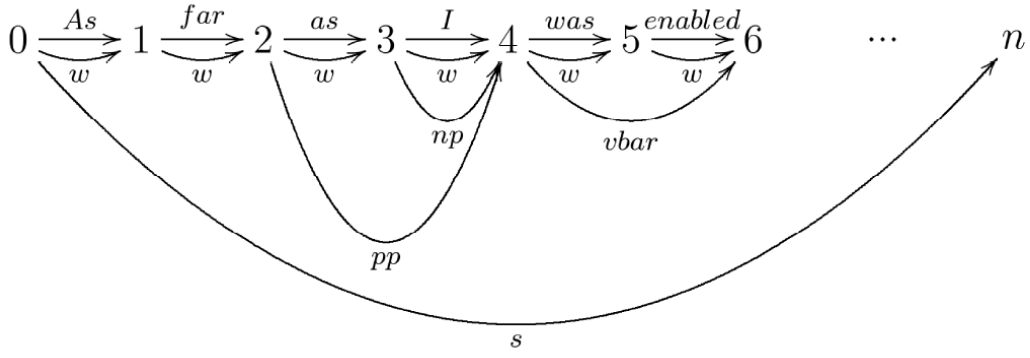

Рисунок 1.2

```
<s id="6293">
  <w c="w" pos="IN" id="624">As</w>
  <w c="w" pos="RB" id="625">far</w>
```

```
<pp id="21236">
<w c="w" pos="IN" id="626" head="yes">as</w>
<np number="singular" person="1" id="627">
<w c="w" pos="PRP" head="yes" id="628">I</w>
</np>
</pp>
<vbar voice="passive" time="past" id="629" args="+6302">
<w c="w" pos="VBD" stem="be" head="yes" id="630">was</w>
<w c="w" pos="VBN" stem="enable" id="631">enabled</w>
</vbar>
```

Використання спеціальної розмітки для представлення лінгвістичної інформації в документі виглядають досить зручними для задач обробки природної мови. Вони дозволяють легко аналізувати результати обробки користувачем або розробником, ігнорувати не відноситься до задачі розмітку і використовувати стандартні інструменти для обробки. Але основна проблема цього підходу - уявлення складних і пересічних структур. Наприклад, такі структури можуть виникнути внаслідок неоднозначності аналізу тексту на одному з етапів обробки. Подання таких структур значно ускладнює використовувану схему розмітки, що зводить нанівець переваги від використання стандартних програм і аналізований людиною.

1.8 Класифікація текстових документів

Існує два протилежні підходи до формування словника (множини F(C)) і побудови правил:

- машинне навчання - передбачається наявність навчальної вибірки документів, за яким будується множина F(C);

- експертний метод - передбачає, що виділення ознак - множини F(C) - і складання правил проводиться експертами.

У разі машинного навчання аналізується статистика лінгвістичних шаблонів (таких як лексична близькість, повторюваність слів і т. п.) з документів навчальної вибірки. У неї повинні входити документи, що відносяться до кожної рубрики, щоб створити набір ознак (статистичну сигнатуру) для кожної рубрики, який згодом буде використовуватися для класифікації нових документів. Перевагою даного підходу є відсутність необхідності в словниках, які складно побудувати для великих предметних областей знань. Однак щоб уникнути неправильної класифікації, потрібно забезпечити гарне представлення документів для кожної рубрики.

У другому випадку формування словника (безлічі F (C)) може бути виконано на основі набору термінів предметної області і відношень між ними (основні терміни, синоніми і родинні терміни). класифікації може потім визначити рубрику документа відповідно до частоти, з якої з'являються виділені в тексті терміни (ключові поняття).

Можлива й комбінація двох описаних підходів, коли виділення ознак і складання правил виконуються автоматично на основі навчальної вибірки, і в той же час правила будуються в такому вигляді, щоб експерту була зрозуміла логіка автоматичної рубрикації, і у нього була можливість вручну коригувати ці правила.

Для класифікації текстових документів успішно використовуються багато методів і алгоритмів класифікації Data Mining. Наприклад: Naive Bayes, метод найменших квадратів, C4.5, SVM і ін. Очевидно, що потрібна модифікація цих методів для роботи з текстовою інформацією. Як правило, адаптація алгоритмів пов'язана з тим, що поняття незалежної змінної пов'язано не з атрибутами об'єкта, а з наявністю в текстовому документі тієї чи іншої ознаки f. Розглянемо

модифікацію таких алгоритмів на прикладі методів Naive Bayes та Роше, описаних далі.

1.8.1 Метод Naive Bayes

Метод Байєса це простий класифікатор, заснований на ймовірнісної моделі, що має сильне припущення незалежності компонент вектора ознак. Зазвичай це припущення не відповідає дійсності і тому одна з назв методу - Naıve Bayes (Наївний Байес).

Імовірнісна модель методу заснована на відомій формулі Байеса по обчисленню апостеріорної ймовірності гіпотез. Застосовуючи цю формулу для завдання класифікації текстів, отримаємо ймовірність того, що документ належить категорії:

$$P(y = c_r|E) = P\left(x_1 = c_p^1 \middle| y = c_r\right) \times P(x_2 = c_d^{21}|y = c_r) \times \ldots \times$$
$$\times P(x_m = c_b^m|y = c_r) = P(y = c_r)/P(E) \qquad (5)$$

Де: y - залежна змінна, що вказує на належність документа до категорії c_r;

E – подія, що полягає в наявності в текстовому документі ознак, що характеризують категорію c_r;

Як і всі ймовірні класифікатори, класифікатор, заснований на методі Байєса, правильно класифікує документи, якщо відповідний документу клас більш імовірний, ніж будь-який інший. У цьому випадку формула для визначення найбільш вірогідною категорії прийме наступний вигляд:

Припустимо, що класифікатор складається з рубрик та може бути виражена через параметрів. Тоді відповідний алгоритм Байеса для класифікації тексту матиме параметрів. Але на практиці, найчастіше за все (випадок бінарної класифікації) і. Тому, число параметрів для методу Байеса зазвичай одно, де - розмірність вектора ознак.

Наївний класифікатор Байеса має кілька властивостей, які роблять його надзвичайно корисним практично, не дивлячись на те, що сильні припущення незалежності часто порушуються. Цей метод показує високу швидкість роботи і досить високу якість класифікації. Його можна рекомендувати для побудови класифікатора, коли існую жорсткі обмеження на час рахунку і скористатися більш точними методами, не представляється можливі.

1.8.2 Метод Роше

Одним з найбільш простих класифікаторів, заснованих на векторної моделі, є так званий класифікатор Роше. Основна особливість цього методу полягає в тому, що для кожної рубрики обчислюється зважений центр ваги. Він виходить вирахуванням ваги кожного терма векторів ознак не відповідають рубриці документів, з вагів термів векторів ознак відповідних рубриці документів.

Нехай кожен документ рубрики буде представлений у вигляді вектора ознак наступним чином. Тоді рубрика буде представлена у вигляді вектора ознак. Для кожної рубрики обчислюється зважений центр ваги.

Таким чином, вийшов зважений центр ваги представляє рубрику в просторі ознак. Належність рубриках невідомого документа, визначається шляхом обчислення відстані між центроїдом кожної з рубрик та вектором документа, що класифыкується. Якщо відстань не перевищує деякий, наперед заданий поріг, документ вважається належить даній рубриці.

Практичне дослідження методу Роше показали, що даний метод має високу ефективність у вирішенні задачі класифікації текстів. Однією з головних його особливостей є можливість змінювати вектор зваженого центроїда рубрики, без перенавчання класифікатора. Ця властивість може бути корисно, наприклад, у випадках, коли навчальна колекція часто поповнюється новими документами, а перенавчання займає надто багато часу. Завдяки своїй результативності та

простоті метод Роше став одним з найпопулярніших в розглянутій нами області і часто використовується як базовий, для порівняння ефективності різних класифікаторів.

1.9 Кластеризація текстових документів

Всі алгоритми кластеризації базуються на вимірах схожості з різними критеріями. Деякі використовують слова, часто з'являються разом (лексичну близькість), інші використовують видобувні особливості (Такі як імена людей і т. П.). Різниця полягає також і в створюваних кластерах. Виділяють три основні типи методів кластеризації документів:

- **ієрархічний** - створює дерево з усіма документами в кореневому вузлі і одним документом в вузлі-листі. Проміжні вузли містять різні документи, які стають більш і більш спеціалізованими в міру наближення до листя дерева. Цей метод корисний, коли досліджують нову колекцію документів і хочуть отримати загальне представлення про неї;
- **бінарний** - забезпечує угруповання і перегляд документальних кластер по посиланнях подібності. В один кластер поміщаються найближчі за своїми властивостями документи. В процесі кластеризації будується базис посилань від документа до документа, заснований на вагах і спільному вживанні визначаються ключових слів;
- **нечіткий** - включає кожен документ в усі кластери, але при цьому зв’язує з ним вагову функцію, що визначає ступінь приналежності даного документа певного кластеру.

1.9.1 Ієрархічні методи кластеризації текстів

Методи ієрархічної кластеризації бувають:

- агломеративні - кластеризація виконується, починаючи з індивідуальних елементів, групуючи їх в кластери (від низу до верху);
- дивізимні - кластеризація виконується, починаючи з єдиного кластера ірозбиваючи його на кілька (зверху вниз).

Ієрархічна агломеративного кластеризація (HAC - Hierarchical Agglomerative Clustering) спочатку представляє кожен з N документів окремим кластером. В процесі кластеризації ці кластери об'єднуються, і кількість кластерів зменшується до тих пір, поки один кластер НЕ буде містити всі N документів. Такий підхід різниться методами групування окремих кластерів:

- однозв'язний метод групує найближчих членів;
- повнозв'язну - далеких членів;
- средньозв’язний - найближчих до середини членів.

Результатами такої кластеризації є дендрограмма. Приклад даного виду діаграм наведено нижче:

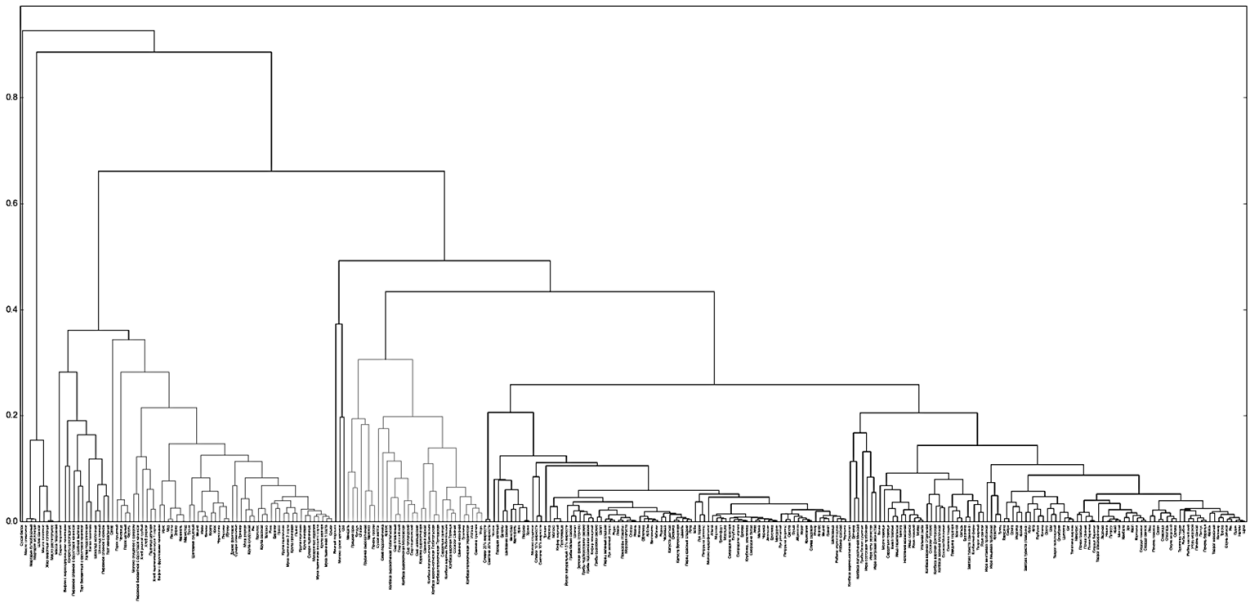

Рисунок 1.3 – Приклад дендрограми

Представником дивізимної ієрархічної кластеризації текстових документів є алгоритм дивізімного поділу по головному напрямку (PDDP - Principal Direction

Divisive Partitioning). Він будує бінарне дерево, в якому кожен вузол містить документи. PDDP починає будувати дерево з кореневого кластера, який містить всі документи. Далі він рекурсивно ділить кожен лист дерева на два дочірніх вузла, поки зберігається критерій ділення. Для збереження балансування бінарного дерева PDDP використовує функцію розкиду для визначення необхідності поділу вузла. Ця функція обчислює, наскільки близькі елементи в кластері. Наприклад, якщо середньоквадратична відстань кластера більше заданого порогового значення, то кластер (вузол дерева) має бути поділений. Матриця слів і документів використовується для визначення головного напрямку і поділу гіперпростору.

1.10 Висновки до розділу 1

У даному розділі були теоретично розглянуті методи, етапи та алгоритми інтелектуального аналізу текстової інформації. Сформульовано такі етапи: попередня обробка, виділення ключових слів та класифікація результатів. Розглянуто наступні методи попередної обробки: словниковий, статистичний та побудований на основі Y-інтерпретації закону Бредфорда. Для виділення ключових слів проаналізовано основні статистичні, лексичні та гібридні методи. Для етапу об’єднання методів виділено метод Naïve Bayes та метод Роше.

Для подальшого аналізу обрано наступні методи: словниковий, статистичний та побудований на основі Y-інтерпретації закону Бредфорда, TF-IDF міра, F-міра та метод лакричных шаблонів.

2. РЕАЛІЗАЦІЇ

2.1 Бібліотеки

2.1.1 Apache OpenNLP

Apache software foundation розробив та виклав у відкритий доступ бібліотеку OpenNLP. Бібліотека Apache OpenNLP являє собою інструментарій на основі машинного навчання для обробки текстів на природній мові. Вона підтримує найбільш поширені завдання обробки ПМ, такі як токенізація, сегментація речень, розмітка частин мови, виділення семантично значущих об'єктів і синтаксичного аналізу. Ці завдання, як правило, потрібно для створення більш складних систем з обробки тексту. OpenNLP також включає в себе такі методи навчання: максимуму ентропії і навчання на основі перцептронів.

Бібліотека Apache OpenNLP містить компоненти, що дозволяють побудувати один повний програмний продукт з обробки природної мови. Ці компоненти включають в себе: детектор пропозиції, токенізатор, пошук власних назв, засоби з категорізації документів, класифікатор частин мови, синтаксичний аналізатор. Компоненти містять частини, які дозволяють виконати відповідне завдання обробки природної мови, для навчання моделі і також для оцінки загальної моделі. Кожен з цих об'єктів доступний через інтерфейс прикладного програмування (API). Крім того, інтерфейс командного рядка (CLI) надається для зручності експериментів і навчання.

2.1.2 Snowball Stemmer for Java

Другою бібліотекою, що застосовується в проекті є Snowball Stemmer, побудований на алгоритмі Портера, що був розроблений у 1980 році.

Основна ідея Стеммер Портера полягає в тому, що існує обмежена кількість словотвірних суфіксів, і стемінг слова відбувається без використання будь-яких баз основ: тільки безліч існуючих суфіксів і вручну задані правила.

Алгоритм складається з п'яти кроків. На кожному кроці відсікається словотвірний суфікс і решта перевіряється на відповідність правилам (наприклад, для російських слів основа повинна містити не менше однієї голосної). Якщо отримане слово задовольняє правилам, відбувається перехід на наступний крок. Якщо немає - алгоритм вибирає інший суфікс для відсікання. На першому кроці відсікається максимальний формоутворювальний суфікс, на другому - буква «і», на третьому – словотвірний суфікс, на четвертому - суфікси чудових форм, "ь" і одна з двох «н».

Даний алгоритм часто обрізає слово більше необхідного, що ускладнює отримання правильної основи слова, наприклад inform { informal, informality, informally, informals, informant, informant's, informants, information, information's, informational, informations, informative, informatively, informer, informer's, informers, informs } (при цьому реально незмінна частина - inform, але Стеммер вибирає для видалення найбільш довгу морфему). Також Стеммер Портера не справляється зі всілякими змінами кореня слова (наприклад, випадають і швидкі голосні).

2.2 Готові реалізації

2.2.1 KH Coder

KH Coder є додатком для кількісного аналізу тексту у прикладній лінгвістиці. Він може обробляти тексти японською, англійською, французькою, німецькою, італійською, португальською та іспанською мовами. Шляхом введення вихідних текстів можна виконувати лінгвістичний і статистичний

аналіз. Продукт має такі функціональні можливості як KWIC, статистика колокацій, суміжності мережі, карти, що самоорганізуються, кластерний аналіз та аналізу відповідності.

Даний програмний продукт є безкоштовним і його використовували у більш ніж 120 дослідженнях в усьому світі.

2.2.2 STATISTICA Text Miner

STATISTICA Text Miner - це додаток до STATISTICA Data Miner, який ідеально підходить для того щоб переводити неструктурований текст в цінну інформацію, придатну для прийняття "золотих" рішень. Більшість користувачів, знайомих з системами Text Mining, добре знають про те, що, як правило, реальні "необроблені" дані є не завжди придатними для сприйняття і подальшого аналізу.

STATISTICA Text Miner дозволяє вибрати з потоку інформації необхідні дані і структурувати їх. STATISTICA Text Miner інтегрована в додаток STATISTICA Data Miner і в інші продукти компанії StatSoft, відмінною рисою яких є те, що вони є найбільш повними і потужними інструментами на ринку і виконані з бездоганним увагою як до питань підвищення ефективності, так і до питань розширення можливостей.

Цей додаток використовує багато-потокові комп'ютерні технології для досягнення максимальної продуктивності передових багатопроцесорних серверних систем.

Також як і всі компоненти STATISTICA Data Miner, STATISTICA Text Miner спеціально розроблений як загальний засіб з відкритою архітектурою, призначене для видобутку даних з потоку неструктурованою інформації. Особливістю засобів Text Mining, а також інших аналітичних інструментів, доступних в STATISTICA Text Miner, є те, що в якості вхідних даних можна

використовувати не тільки текстові документи або веб-сторінки, але також посилання, списки або кластери.

Неструктурована інформація, яку ви аналізуєте навіть може включати в себе неперетворені бітові зображення, звукові файли і т. д.

2.3 Висновки до розділу 2

У даному розділі було розглянуто бібліотеки та програмні засобі, що будуть використані для подальшої реалізації та аналізу алгоритмів інтелектуального аналізу текстів.

Це Java бібліотеки для стемінгу (Snowball Stemmer for Java), для виділення частин мови (Apache OpenNPL), а також KH coder, який являє собою повноцінний продукт з Text Mining.

3. ВИБІР, РОЗОРБКА І ТЕСТУВАННЯ ЕФЕКТИВНОСТЫ МЕТОДІВ. АНАЛІЗ ОТРИМАНИХ РЕЗУЛЬТАТІВ

Для виконання порівняльного аналізу алгоритмів була розроблена програма, у якій реалізовані деякі методи, описані у першому розділі. Мовою програмування було обрано Java, оскільки саме для неї розроблено багато бібліотек, що полегшують створення даної системи. Більш детально вибір технологій описаний у розділі 4: Функціонально- вартісний аналіз програмного продукту. Код наведений у Додатку 1.

3.1 Видалення стоп слів

Першим набором алгоритмів, які було вирішено порівняти є алгоритми пошуку стоп-слів, що є невід'ємною частиною попередньої обробки текстів. Для порівняння обрано алгоритм, побудований на Y-інтерпретації закону Бредфорда, причому в двох варіаціях: з попереднім стемінгом та виконанням стемінгу після видалення стоп-слів. Другий алгоритм – словниковий. Словник представлений на сайті http://www.ranks.nl/stopwords. Третій – на основі об'єднання слів, що зустрічаються найчастіше у текстах набору. Усі алгоритми описані у розділі 1.5 «Методи видалення стоп слів».

3.1.1 За законом Бредфорда без стемінгу

Слід зазначити, що у даній роботі використовується стемер Портера, що працює на принципі N-грам, тому результати можуть сильно різнитись від порядку застосування алгоритмів, що і буде продемонстровано нижче.

Для демонстрації результатів було проведено аналіз на прикладах з 4 тестів наукової тематики. Як показало дослідження для отримання достовірних даних достатньо 4 текстів. Результати наведено в таблицях 3.1 та 3.2.

Біля назви тексту в дужках наведено загальну кількість слів у тексті. Для кожного тексту розраховано кількість та частоту входжень слова у текст. Після цього вручну було перевірено точність методу. В таблиці виділено слова, які алгоритм відмітив, як стоп- слова, проте вони несуть інформативне навантаження у тексті.

Таблиця 3.1 – Результати виявлення стоп-слів за законом Бредфорда без стемінгу, частина 1

Назва тексту (загальна кількість слів)	Machine Learning (1784)			Sentiment Analysys (1725)		
	Word	Amount	Percentage	Word	Amount	Percentage
	the	92	5.156951	the	99	5.73913
	learning	60	3.363229	of	72	4.173913
	of	60	3.363229	a	59	3.42029
	a	55	3.08296	sentiment	47	2.724638
	and	53	2.970852	to	45	2.608696
	to	51	2.858744	and	44	2.550725
	in	42	2.35426	is	33	1.913043
	is	35	1.961883	analysi	29	1.681159
	machine	29	1.625561	in	27	1.565217
	data	22	1.233184	or	25	1.449275
	from	20	1.121076	that	21	1.217391
	as	20	1.121076	on	20	1.15942
	with	18	1.008969	as	19	1.101449
	on	17	0.952915	text	17	0.985507
	are	17	0.952915	it	17	0.985507
	that	16	0.896861	this	16	0.927536
	by	11	0.616592	can	15	0.869565
	it	11	0.616592	be	14	0.811594
	be	11	0.616592	for	13	0.753623
	training	11	0.616592	featur	12	0.695652
	theory	10	0.560538	classifi	11	0.637681
	Field	10	0.560538	are	11	0.637681
	can	10	0.560538			
Всього		677	37.94843		666	38.6087
Всього після перевірки		557	31.22222		557	32.2889

Таблиця 3.2 – Результати виявлення стоп-слів за законом Бредфорда без стемінгу, частина 2

Назва тексту (загальна кількість слів)	Text Mining (1291)			Big Data (1721)		
	Word	Amount	Percentage	Word	Amount	Percentage
	data	82	6.351665	data	88	5.113306
	the	74	5.731991	and	82	4.764672
	and	46	3.563129	the	74	4.299826
	of	37	2.865995	of	62	3.602557
	mining	32	2.478699	to	43	2.498547
	in	32	2.478699	big	38	2.208019
	to	29	2.246321	a	37	2.149913
	is	22	1.704105	in	31	1.801278
	a	19	1.471727	is	21	1.220221
	as	17	1.316809	that	16	0.929692
	for	16	1.239349	for	14	0.813481
	be	14	1.084431	are	14	0.813481
	or	13	1.006971	as	14	0.813481
	learning	12	0.929512	with	13	0.755375
	analysis	12	0.929512	information	12	0.697269
	process	11	0.852053	can	12	0.697269
	patterns	11	0.852053	or	11	0.639163
	used	11	0.852053	on	11	0.639163
	knowledge	10	0.774593	it	11	0.639163
	discovery	10	0.774593	storage	10	0.581058
				analytics	10	0.581058
				parallel	10	0.581058
				processing	9	0.522952
				from	9	0.522952
Всього		510	39.50426		652	37.88495
Всього після перевірки		319	24.70952		513	29.80825

3.1.2 За законом Бредфорда зі стемінгом

Аналогічно попередньому розділу проведемо аналіз для тих самих текстів, але з попереднім стемінгом.

Таблиця 3.3 – Результати виявлення стоп-слів за законом Бредфорда зі стемінгом, частина 1

Назва тексту (загальна кількість слів)	Machine Learning (1784)			Sentiment Analysys (1725)		
	Word	Amount	Percentage	Word	Amount	Percentage
	the	92	5.156951	the	99	5.73913
	learning	60	3.363229	of	72	4.173913
	of	60	3.363229	a	59	3.42029
	a	55	3.08296	sentiment	47	2.724638
	and	53	2.970852	to	45	2.608696
	to	51	2.858744	and	44	2.550725
	in	42	2.35426	is	33	1.913043
	is	35	1.961883	analysi	29	1.681159
	machine	29	1.625561	in	27	1.565217
	data	22	1.233184	or	25	1.449275
	from	20	1.121076	that	21	1.217391
	as	20	1.121076	on	20	1.15942
	with	18	1.008969	as	19	1.101449
	on	17	0.952915	text	17	0.985507
	are	17	0.952915	it	17	0.985507
	that	16	0.896861	this	16	0.927536
	by	11	0.616592	can	15	0.869565
	it	11	0.616592	be	14	0.811594
	be	11	0.616592	for	13	0.753623
	training	11	0.616592	featur	12	0.695652
	theory	10	0.560538	classifi	11	0.637681
	Field	10	0.560538	are	11	0.637681
	can	10	0.560538			
Всього		677	37.94843		666	38.6087
Всього після перевірки		557	31.22222		557	32.2889

Таблиця 3.4 – Результати виявлення стоп-слів за законом Бредфорда зі стемінгом, частина 2

Назва тексту (загальна кількість слів)	Text mining (1291)			Big Data (1721)		
	Word	Amount	Percentage	Word	Amount	Percentage
	data	82	6.351665	data	88	5.113306
	the	74	5.731991	and	82	4.764672
	and	46	3.563129	the	74	4.299826
	of	37	2.865995	of	62	3.602557
	mine	33	2.556158	to	43	2.498547
	in	32	2.478699	big	38	2.208019
	to	29	2.246321	a	37	2.149913
	is	22	1.704105	in	31	1.801278
	a	19	1.471727	is	21	1.220221
	use	18	1.394268	process	18	1.045904
	as	17	1.316809	that	16	0.929692
	for	16	1.239349	it	16	0.929692
	be	14	1.084431	for	14	0.813481
	process	13	1.006971	are	14	0.813481
	or	13	1.006971	use	14	0.813481
	set	12	0.929512	as	14	0.813481
	learn	12	0.929512	with	13	0.755375
	databas	12	0.929512	can	12	0.697269
				set	11	0.639163
				or	11	0.639163
				inform	11	0.639163
Всього		501	38.80713		651	37.82684
Всього після перевірки		349	27.03333		507	29.45969

3.1.3 Результати

Зведемо результати двох попередніх підрозділів у таблиці та порівняємо з результатами комерційного продукту advego.ru/

Таблиця 3.5 – Результати для Y-інтерпретації закону Бредфорда без стемінгу

Назва тексту	Всього слів	advego		Бредфорд		Перевірка		Похибка відносно перевірки		Похибка відносно advego	
		Всього	Процент	Всього	Процент	Всього	Процент	Всього	Процент	Всього	Процент
Text mining	1291	377	29,20	510	39.50	319	24.70	191	37.45	133	35.27
Big Data	1721	608	35,32	652	37.88	513	29.80	139	21.31	44	7.23
Machine learning	1784	644	36,09	677	37.94	557	31.22	120	17.72	33	5.12
Sentiment Analysys	1725	608	35,24	666	38.60	557	32.28	109	16.36	58	9.54

Таблиця 3.6 – Результати для Y-інтерпретації закону Бредфорда зі стемінгом

Назва тексту	Всього слів	advego		Бредфорд		Перевірка		Похибка відносно перевірки		Похибка відносно advego	
		Всього	Процент	Всього	Процент	Всього	Процент	Всього	Процент	Всього	Процент
Text mining	1291	377	29,20	501	38.80	349	27.03	152	30.33	124	32.8912
Big Data	1721	608	35,32	651	37.82	507	29.45	144	22.11	43	7.07
Machine learning	1784	644	36,09	677	37.94	557	31.22	120	17.72	33	5.12
Sentiment Analysys	1725	608	35,24	666	38.60	557	32.28	109	16.36	58	9.53

Складемо порівняльну характеристику з урахуванням не лише точності, а й повноти виділення стоп-слів.

Таблиця 3.7 – Зведені результати для Y-інтерпретації закону Бредфорда

Назва тексту	Всього слів	Без стемінгу				Зі стемінгом			
		Всього	Процент	Точність	Повнота	Всього	Процент	Точність	Повнота
Text mining	1291	510	39.50	64.73	81.82%	501	38.80	67.1088	91.98%
Big Data	1721	652	37.88	92.77	81.48%	651	37.82	92.93	80.08%
Machine learning	1784	677	37.94	94.88	84.38%	677	37.94	94.88	84.38%
Sentiment Analysys	1725	666	38.60	90.46	90.84%	666	38.60	90.47	90.84%

Як видно з таблиці, методи після стемінгу дає дещо кращі результати. Також можна помітити залежність між кількістю слів та ефективністю роботи алгоритму. Проілюструємо це на графіках.

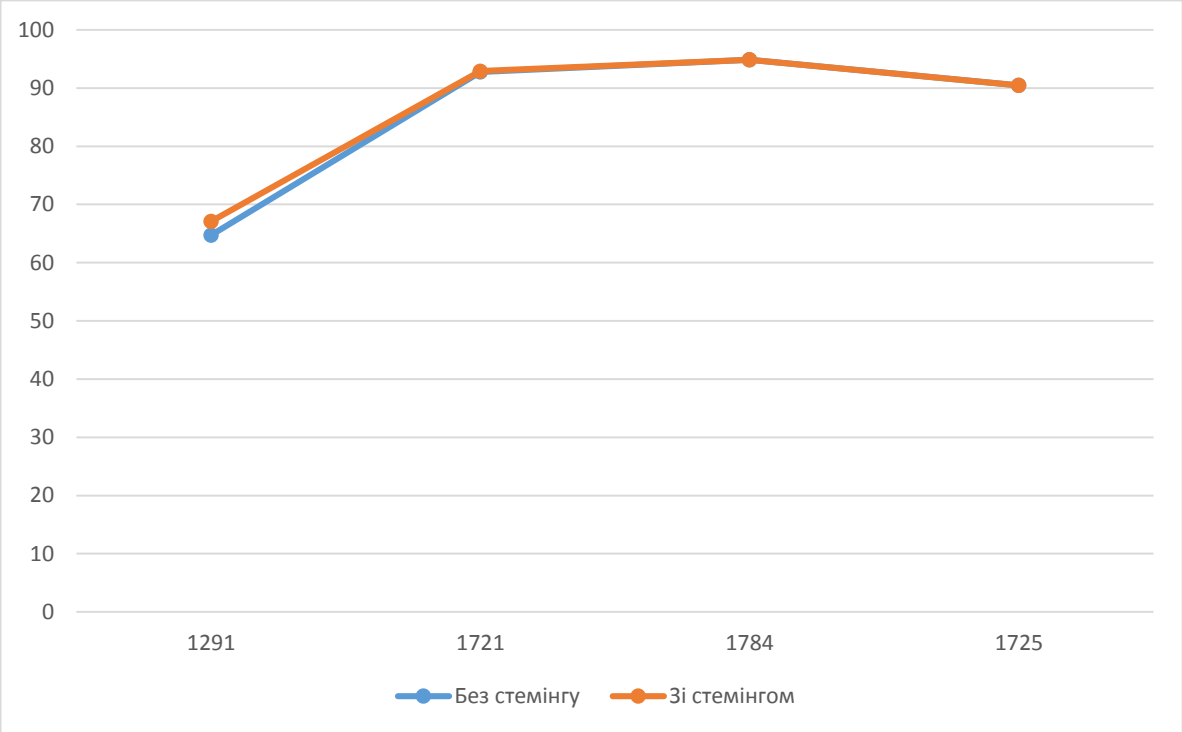

Рисунок 3.1 – залежність між точністю і кількістю слів

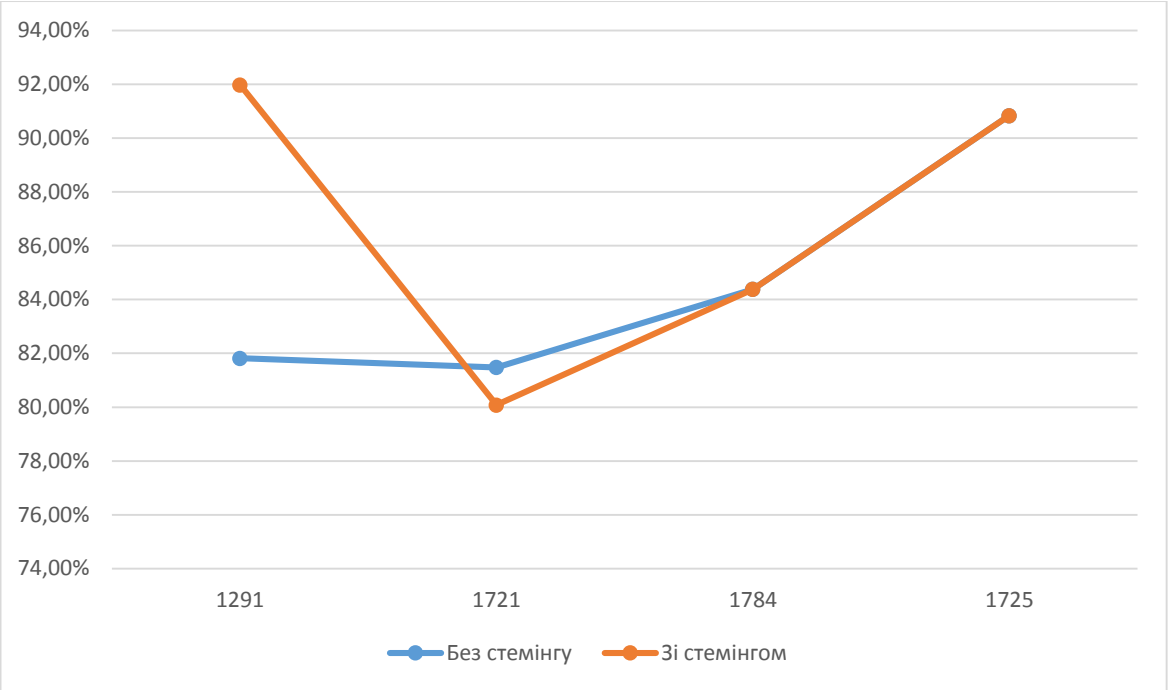

Рисунок 3.2 – залежність між повнотою і кількістю слів

3.1.4 Словниковий метод

Як вже біло зазначено раніше, стоп-слова задані заздалегідь вручну. В даному випадку їх список взятий із сайту http://www.ranks.nl/stopwords, цей перелік налічує 665 слів.

Було проаналізовано 15 текстів наукової тематики. Результати наведено у таблиці нижче.

Таблиця 3.8 – Результати для словникового методу

Назва тексту	Загальна кількість слів	Кількість видалених стоп-слів		Advego	
		Кількість	Відсоток	Кількість	Відсоток
Getting Started in Text Mining Part Two	1505	826	54,88%	501	33,29%
Text Mining Introductory Overview	1709	964	56,41%	576	33,70%
Machine Learning 2	1784	961	53,87%	494	27,69%
Big Data	1721	776	45,09%	608	35,33%
Machine Learning	1784	961	53,87%	644	36,10%
How Companies Can Use Sentiment Analysis to Improve their business	1693	970	57,29%	622	36,74%
Getting Started in Text Mining	1891	1074	56,80%	691	36,54%
Sentiment Analysys	1725	906	52,52%	608	35,25%
Sentiment analysis using product review data	3868	1942	50,21%	1409	36,43%
Where to start with text mining	3102	1740	56,09%	1203	38,78%
Text mining is a burgeoning new field that attempt	1677	909	54,20%	574	34,23%
Text Mining	1291	605	46,86%	377	29,20%
Data Mining What is data mining	1636	739	45,17%	457	27,93%
Text categorization	1632	908	55,64%	560	34,31%
Sentiment Analysis of news comments	917	503	54,85%	314	34,24%

З таблиці видно, що словниковий метод видаляє набагато більше слів, це зумовлено словником, що підходить для багатьох типів і тематик текстів, а також для SEO.

3.1.5 Метод об’єднання

Як вже було описано вище, даний метод полягає у знаходженні та об’єднанні слів, що зустрічаються найчастіше у текстах із набору.

Розглянемо підхід до вибору порогу входження слова до переліку стоп-слів з інтерпретацією закону Бредфорда.

Із набору текстів була виділена зона J0 (зона стоп-слів). Ці слова наведені у таблиці нижче. Сірим виділено помилки у побудові словнику.

Таблиця 3.9 – Результати для словникового методу

Word	Amount	Percentage
the	1331	4.775402
of	953	3.419202
to	746	2.676521
and	719	2.57965
a	681	2.443312
in	504	1.808266
is	496	1.779564
data	393	1.410017
that	347	1.244977
for	336	1.205511
are	259	0.929248
as	254	0.911309
be	245	0.879018
it	233	0.835964
use	226	0.81085
or	224	0.803674
on	201	0.721154
Sentiment	192	0.688863
text	189	0.6781
with	186	0.667336
mine	186	0.667336
Learn	185	0.663749
can	175	0.62787
this	174	0.624282
word	138	0.495121
from	135	0.484357
by	132	0.473594
analysi	122	0.437715
machin	112	0.401837
document	109	0.391073
process	106	0.38031
an	101	0.362371
have	100	0.358783

Всього помилкових слів - 1594 із 10589 слів у словнику всього, тобто похибка 15.053%, а точність 84.947%. З одного боку, за числами це дуже непогана точність, а з іншого на цьому прикладі видно, що цей метод видалить основні ключові слова, які характеризують цей набір тестів.

Цей метод може показати кращі результати, якщо взяти більше текстів абсолютно різної направленості, але така обробка за часом підготовки словника програє ручній підготовці словника.

3.1.6 Приклад тексту із проведеної попередньою обробкою

Маємо фрагмент тексту про Data Mining:

Data mining is an interdisciplinary subfield of computer science. It is the computational process of discovering patterns in large data sets involving methods at the intersection of artificial intelligence, machine learning, statistics, and database systems. The overall goal of the data mining process is to extract information from a data set and transform it into an understandable structure for further use. Aside from the raw analysis step, it involves database and data management aspects, data pre-processing, model and inference considerations, interestingness metrics, complexity considerations, post-processing of discovered structures, visualization, and online updating. Data mining is the analysis step of the "knowledge discovery in databases" process, or KDD.

The term is a misnomer, because the goal is the extraction of patterns and knowledge from large amounts of data, not the extraction (mining) of data itself. It also is a buzzword and is frequently applied to any form of large-scale data or information processing (collection, extraction, warehousing, analysis, and statistics) as well as any application of computer decision support system, including artificial intelligence, machine learning, and business intelligence. The book Data mining: Practical machine learning tools and techniques with Java (which covers mostly machine learning material) was originally to be named just Practical machine learning, and the term data mining was only added for marketing reasons. Often the more general terms (large scale) data analysis and analytics – or, when referring to actual methods, artificial intelligence and machine learning – are more appropriate.

Після стемінгу та видалення стоп-слів зі словнику:

data mine interdisciplinari subfield comput scienc . comput process discov pattern data involv method intersect artifici intellig machin learn statist databas system . goal data mine process extract data set transform understand structur . raw analysi step involv databas data manag aspect data pre-process model infer consider interesting metric complex consider post-process discov structur visual onlin updat . data mine analysi knowledg discoveri databas process KDD .

term misnom goal extract pattern knowledg larg amount data extract mine data . buzzword frequent appli form large-scal data process collect extract wareh analysi statist well applic comput decis system includ artifici intellig machin learn busi intellig . book data mine practic machin learn tool techniqu Java cover machin learn materi origin practic machin learn term data mine market reason . term larg scale data analysi analyt – refer actual method artifici intellig machin learn – appropri .

Після стемінгу та видалення стоп-слів за Бредфордом:

interdisciplinari subfield comput scienc . comput discov pattern involv method intersect artifici intellig machin learn statist system . goal mine process extract data set transform understand structur . raw analysi step involv manag aspect data pre- model infer consider interesting metric complex consider post- discov structur visual onlin updat . analysi knowledg discoveri KDD .

term misnom goal extract pattern knowledg larg amount data extract. buzzword frequent appli form large-scal data collect extract wareh analysi statist well applic comput decis system includ artifici intellig machin busi intellig . book practic machin learn tool techniqu Java cover machin learn materi origin practic machin term market reason . term larg scale analysi analyt – refer actual method artifici intellig machin– appropri .

Після стемінгу та видалення стоп-слів за методом об’єднання:

interdisciplinari subfield comput scienc . comput discov pattern involv method intersect artifici intellig machin statist databas system . goal extract transform understand structur . raw step involv databas manag aspect pre- model infer consider interesting metric complex consider post- discov structur visual onlin updat . analysi knowledg discoveri KDD .

term misnom goal extract pattern knowledg larg amount extract. buzzword frequent appli form large-scal collect extract wareh statist well applic comput decis

system includ artifici intellig machin busi intellig . book mine practic machin tool techniqu Java cover machin learn materi origin practic machin term data mine market reason . term larg scale– refer actual method artifici intellig machin– appropri .

Розглянувши результати, очевидно, що найкраще себе показує саме словниковий метод. Так як обидва інші видаляють основні ключові слова. В даному прикладі такі як: data, mining, process, learning, set, database.

Ці слова якнайкраще характеризують текст, є ключовими, а за даним методом, вони мають бути виключеними з аналізу на етапі попередньої обробки. Тому можна зробити висновок, що дані методи не варто використовувати у системі.

3.2 Виділення ключових слів

Продемонструємо роботу алгоритмів виділення ключових слів. Для аналізу взято 3 алгоритми: 2 статистичних та один гібридний.

3.2.1 Міра TF-IDF

Наведемо приклади виділення ключових слів за мірою TF-IDF у таблиці нижче.

Таблиця 3.10 – Результати для міри TF-IDF

Big Data		Text Mining		Text categorization		Text Mining inductor overview	
Word	TF-IDF	Word	TF-IDF	Word	TF-IDF	Word	TF-IDF
datum	61	datum	15	text	12	word	12
information	10	Data	12	document	11	text	11
set	8	mining	11	categorization	10	document	10
analytic	7	process	7	category	6	example	10
processing	7	pattern	6	word	6	mining	10
storage	7	analysis	5	language	5	term	8
architecture	6	learning	5	method	5	application	6
system	6	pre-processing	5	technique	5	input	6

Таблиця 3.10 (продовження)

Big Data		Text Mining		Text categorization		Text Mining inductor overview	
Word	TF-IDF	Word	TF-IDF	Word	TF-IDF	Word	TF-IDF
technology	6	database	4	authorship	4	number	6
application	5	machine	4	classification	4	algorithm	4
framework	5	method	4	example	4	datum	4
process	5	set	4	machine	4	domain	4
size	5	task	4	model	4	indexing	4
velocity	5	analytic	3	retrieval	4	information	4
volume	5	business	3	scheme	4	type	4
characteristic	4			set	4	approach	3
database	4			test	4	cluster	3
definition	4			topic	4	form	3
insight	4			training	4	language	3
management	4			application	3	list	3
model	4			approach	3	method	3
organization	4			author	3	project	3
server	4			datum	3	service	3
time	4			learning	3	time	3
type	4			metada	3	variable	3
value	4			mining	3		
3v	3			n-gram	3		
ability	3			news	3		
algorithm	3						
analysis	3						
business	3						
component	3						
computing	3						
concept	3						
connection	3						
cost	3						
difficulty	3						
factory	3						

З таблиці видно, що ці слова є ключовими, бо вони відповідають ознакам ключових слів для даний текстів і характеризують їх.

3.2.2 F-міра

Продемонструємо результати роботи іншого алгоритму, навівши приклади з використанням тих самих текстів, але з використанням F-міри.

Таблиця 3.11 – Результати для F-міри

Big Data		Text Mining		Text categorization		Text Mining inductor overview	
Word	Weight	Word	Weight	Word	Weight	Word	Weight
analysis	0.04842	Discovering	0.09087	Word	0.08044	Mining	0.0649
varies	0.0443	Learning	0.0587	Sample	0.06145	Numbers	0.0342
Traditional	0.03554	Analysis	0.04284	Learning	0.0602	Variables	0.0341
capture	0.031887	Community	0.2112	Constrain	0.0531	appropriate	0.0319
Information	0.02235	Inferences	0.02004	Particulary	0.04192	Summarise	0.031
Greater	0.021312	Machine	0.0198	Retrieval	0.03791	Automatic	0.0271
Exabytes	0.0193	Databases	0.0161	Traditional	0.0364	algorythms	0.0269
Other	0.01835	statistic	0.01608	Counts	0.0264	Terms	0.0236
Store	0.01805	Phrase	0.01607	Identify	0.02173	department	0.0234
Technological	0.017093	Intersection	0.01445	Training	0.02145	Process	0.0204
Predictive	0.0167			Estimated	0.02054	Design	0.0196
Result	0.01547			Predict	0.01864		
Systems	0.0142						
Requires	0.1405						

Із даних прикладів видно, що цей метод менш ефективний, ніж попередній.

3.2.3 Лінгво-статистичні шаблони

Метод лінгво-статистичних шаблонів значно важчий та набагато більш трудоємкий для реалізації, тому було прийнято рішення використати готовий продукт KH Coder, де вже реалізований цей метод. Результати роботи наведені нижче.

Список словосполучень, виділених програмою, був наведений не повністю, але цього достатньо, щоб зробити висновки щодо роботи алгоритму.

Таблиця 3.12 – Результати для лінгво-статистичних шаблонів частина 1

Big Data		Text Mining	
Word	Weight	Word	Weight
big data	439.161	data mining	428.794
data sets	40.818	of data	40.15
big data analytics	36.523	data analysis	29.659
data lake	17.907	knowledge discovery	24.973
stored data	17.907	data mining step	23.421
data storage	16.181	term data mining	21.51
2.5 exabytes of data	10.647	machine learning	21.22
data size	10.647	data set	19.834
petabytes of data	10.647	data sets	19.834
data set	9.621	data fishing	16.678
process data	8.953	data preparation	16.678
analysis of data sets	8.22	data mining process	13.903
business intelligence	8.207	large data sets	12.444
amount of data	8.09	target data set	11.631

Таблиця 3.13 – Результати для лінгво-статистичних шаблонів частина 2

Text categorization		Text Mining inductor overview	
Word	Weight	Word	Weight
text categorization	66.73	text mining	121.413
document text	21.393	input documents	14.651
document categorization	18.248	text mining applications	14.28
new document	14.455	data mining project	10.593
text classification	13.441	large numbers of documents	7.74
language identification	9.391	large numbers	7.737
document profile	8.859	results of text mining	7.14
document retrieval	8.379	text mining algorithms	7.14
new words	7.045	clusters of words	6.387
document's language	6.62	purpose of text mining	6.361
1990s text categorization	5.593	text mining program	6.361
automatic text categorization	5.593	meaning of text	6.16
text categorization methods	5.593	various data mining	5.557
text categorization problem	5.593	text mining introductory overview	5.464

Таким чином можемо зробити висновок, що гібридні методи виділення ключових слів є найеффективнішими.

3.3 Висновки до розділу 3

У даному розділі було проведено безпосередній аналіз методів, описаних у першому розділі. При аналізі методів виділення стоп-слів було виявлено, що метод, який працює на основі Y-інтерпретації закону Бредфорда чисельно продемонстрував доволі високу точність (близько 85%) проте при більш детальному аналізі було виявлено, що цей метод видаляє найбільш значущі ключові із текстів, що були проаналізовані, що робить даний метод непридатним для використання у такого роду системах у даному вигляді. Словниковий метод не є універсальним, оскільки універсальний словник робить цей метод менш точним (погіршує результат подальшого виявлення колокацій) та менш повним. Тому для використання даного методу варто використовувати словник стоп-слів, розроблений саме для даної предметної області. Було запропоновано скомбінувати ці методи при аналізі великої кількості текстів різної тематики та направленості. Аналізуються слова, що зустрічаються у цих текстах найчастіше та відбираються за Бредфордом. Чим більша варіативність тематики текстів, тим більша якість словника. При аналізі методів виявлення ключових слів, було використано 3 алгоритми: 2 статистичні: міра TF-IDF та F-міра та гібридний – метод лінгво-статистичних шаблонів. При ручному аналізі результатів було виявлено, що F-міра показала значно гірші результати за TF-IFD. Значно складнішим у реалізації та більш ресурсозатратним в плані виконання є алгоритм лінгво-статистичних шаблонів, проте саме цей алгоритм показав найкращий результат, що було показано у даному розділі.

4. ФУНКЦІОНАЛЬНО-ВАРТІСНИЙ АНАЛІЗ ПРОГРАМНОГО ПРОДУКТУ

У даному розділі проводиться оцінка основних характеристик програмного продукту, призначеного для аналізу методів інтелектуального аналізу текстів, що використовуються для структурування знань. Інтерфейс користувача був розроблений за допомогою мови програмування Java у середовищі розробки NetBeans 8.1.

Програмний продукт призначено для використання на персональних комп'ютерах під управлінням операційної системи Windows, Unix, Mac.

Нижче наведено аналіз різних варіантів реалізації модулю з метою вибору оптимальної, з огляду при цьому як на економічні фактори, так і на характеристики продукту, що впливають на продуктивність роботи і на його сумісність з апаратним забезпеченням. Для цього було використано апарат функціонально-вартісного аналізу.

Функціонально-вартісний аналіз (ФВА) – це технологія, яка дозволяє оцінити реальну вартість продукту або послуги незалежно від організаційної структури компанії. Як прямі, так і побічні витрати розподіляються по продуктам та послугам у залежності від потрібних на кожному етапі виробництва обсягів ресурсів. Виконані на цих етапах дії у контексті метода ФВА називаються функціями.

Мета ФВА полягає у забезпеченні правильного розподілу ресурсів, виділених на виробництво продукції або надання послуг, на прямі та непрямі витрати. У даному випадку – аналізу функцій програмного продукту й виявлення усіх витрат на реалізацію цих функцій.

Фактично цей метод працює за таким алгоритмом:

– визначається послідовність функцій, необхідних для виробництва продукту. Спочатку – всі можливі, потім вони розподіляються по двом групам:

ті, що впливають на вартість продукту і ті, що не впливають. На цьому ж етапі оптимізується сама послідовність скороченням кроків, що не впливають на цінність і відповідно витрат.

– для кожної функції визначаються повні річні витрати й кількість робочих годин.

– для кожної функції на основі оцінок попереднього пункту визначається кількісна характеристика джерел витрат.

– після того, як для кожної функції будуть визначені їх джерела витрат, проводиться кінцевий розрахунок витрат на виробництво продукту.

4.1 Постановка задачі

У роботі застосовується метод ФВА для проведення техніко-економічний аналізу розробки програмного продукту, призначеного для аналізу методів інтелектуального аналізу текстів, що використовуються для структурування знань. Оскільки основні проектні рішення стосуються всієї системи, кожна окрема підсистема має їм задовольняти. Тому фактичний аналіз представляє собою аналіз функцій програмного продукту, призначеного для збору, обробки та проведення аналізу гетероскедастичних процесів в економіці та фінансах.

Відповідно цьому варто обирати і систему показників якості програмного продукту.

Технічні вимоги до продукту наступні:

– програмний продукт повинен функціонувати на персональних комп’ютерах із стандартним набором компонент та встановленою JRE;

– забезпечувати високу швидкість обробки великих об’ємів даних;

– забезпечувати зручність і простоту взаємодії з користувачем або з розробником програмного забезпечення у випадку використовування його як модуля;

– передбачати мінімальні витрати на впровадження програмного продукту.

4.1.1 Обґрунтування функцій програмного продукту

Головна функція F_0– розробка програмного продукту, який проводить аналіз текстів та виводить результати, а також необхідні статистичні дані. Виходячи з конкретної мети, можна виділити наступні основні функції ПП:

F_1 – вибір мови програмування;

F_2 – збереження вихідних даних;

F_3 – інтерфейс користувача.

Кожна з основних функцій може мати декілька варіантів реалізації.

Функція F_1:

а) мова програмування Java;

б) мова програмування PHP;

Функція F_2:

а) система управління базами даних;

б) виведення у файли Excel.

Функція F_3:

а) веб-інтерфейс користувача;

б) інтерфейс користувача, створений за технологією JavaFX;

в) консольний інтерфейс.

4.1.2 Варіанти реалізації основних функцій

Варіанти реалізації основних функцій наведені у морфологічній карті системи (рис. 4.1). На основі цієї карти побудовано позитивно-негативну матрицю варіантів основних функцій (таблиця 4.1).

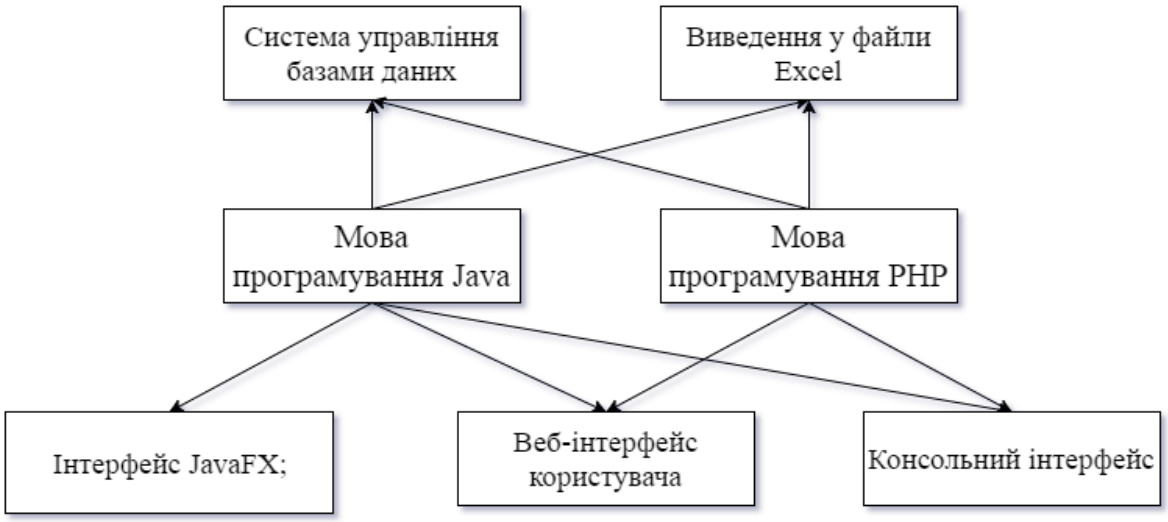

Рисунок 4.1 – Морфологічна карта

Морфологічна карта відображує всі можливі комбінації варіантів реалізації функцій, які складають повну множину варіантів ПП.

Таблиця 4.1 – Позитивно-негативна матриця

Основні функції	**Варіанти реалізації**	**Переваги**	**Недоліки**
F1	*А*	Велика кількість бібліотек з Text Mining у вільному доступі	Займає більше часу при написанні коду
	Б	Займає менше часу при написанні коду	Немає бібліотек з Text Mining у вільному доступі
F2	А	Швидкодія, гнучкість	Необхідно багато додаткового програмного коду, необхідна обробка великої кількості виключень
	Б	Менше програмного коду, зручне подальше використання даних у звітах	Нижча швидкодія, менша розширюваність продукту
F3	*А*	Можливе розгортання на сервері для подальшого загального користування, зручність користування	Час розробки, велика кількість додаткового коду, що не стосується Text Mining Велика кількість додаткових технологій, що необхідно використати
	Б	Зручність користування	Більше коду
	В	Найшвидший метод розробки, що не вимагає додаткового коду	Незручний для користувача

На основі аналізу позитивно-негативної матриці робимо висновок, що при розробці програмного продукту деякі варіанти реалізації функцій варто відкинути, тому, що вони не відповідають поставленим перед програмним продуктом задачам. Ці варіанти відзначені у морфологічній карті.

Функція *F1:*

Оскільки для покращення якості вихідного продукту та зменшення часу розробки деякі функції варто використати з готових бібліотек, що можливо у разі використання Java, варіант б) має бути відкинутий.

Функція *F2:*

Оскільки в рамках даного дослідження швидкість розробки та презентація даних є пріоритетною, то варіант б) має бути відкинутий

Функція *F3:*

Інтерфейс користувача не відіграє велику роль у даному програмному продукту, а варіант а) вимагає набагато більшого часу розробки, тому варіант а) відкидаємо, а вважаємо варіанти б) та в) гідними розгляду.

Таким чином, будемо розглядати такі варіанти реалізації ПП:

1. F1а – F2а – F3б
2. F1а – F2а – F3в

Для оцінювання якості розглянутих функцій обрана система параметрів, описана нижче.

4.2 Обґрунтування системи параметрів ПП

4.2.1 Опис параметрів

На підставі даних про основні функції, що повинен реалізувати програмний продукт, вимог до нього, визначаються основні параметри виробу, що будуть використані для розрахунку коефіцієнта технічного рівня.

Для того, щоб охарактеризувати програмний продукт, будемо використовувати наступні параметри:

– *X1* – Об'єм оперативної пам’яті, що використовується;

– *X2* – об’єм пам’яті для збереження даних;

– *X3* – час обробки даних;

– *X4* – потенційний об’єм програмного коду.

X1: Відображає об'єм оперативної пам’яті.

X2: Відображає об’єм пам’яті в постійній пам’яті персонального комп’ютера, необхідний для збереження та обробки даних під час виконання програми.

X3: Відображає час, який витрачається на обробку вхідних даних.

X4: Показує розмір програмного коду який необхідно створити безпосередньо розробнику.

4.2.2 Кількісна оцінка параметрів

Гірші, середні і кращі значення параметрів вибираються на основі вимог замовника й умов, що характеризують експлуатацію ПП як показано у табл. 4.2.

Таблиця 4.2 – Основні параметри ПП

Назва Параметра	Умовні позначення	Одиниці виміру	Значення параметра		
			гірші	середні	кращі
Об'єм оперативної пам’яті	X1	МБ	2000	800	200
Об’єм пам’яті для збереження даних	X2	Мб	128	64	8
Час обробки даних алгоритмом	X3	мс	5500	4000	600
Потенційний об’єм програмного коду	X4	кількість строк коду	4500	3000	2000

За даними таблиці 4.2 будуються графічні характеристики параметрів – рис. 4.2 – рис. 4.5.

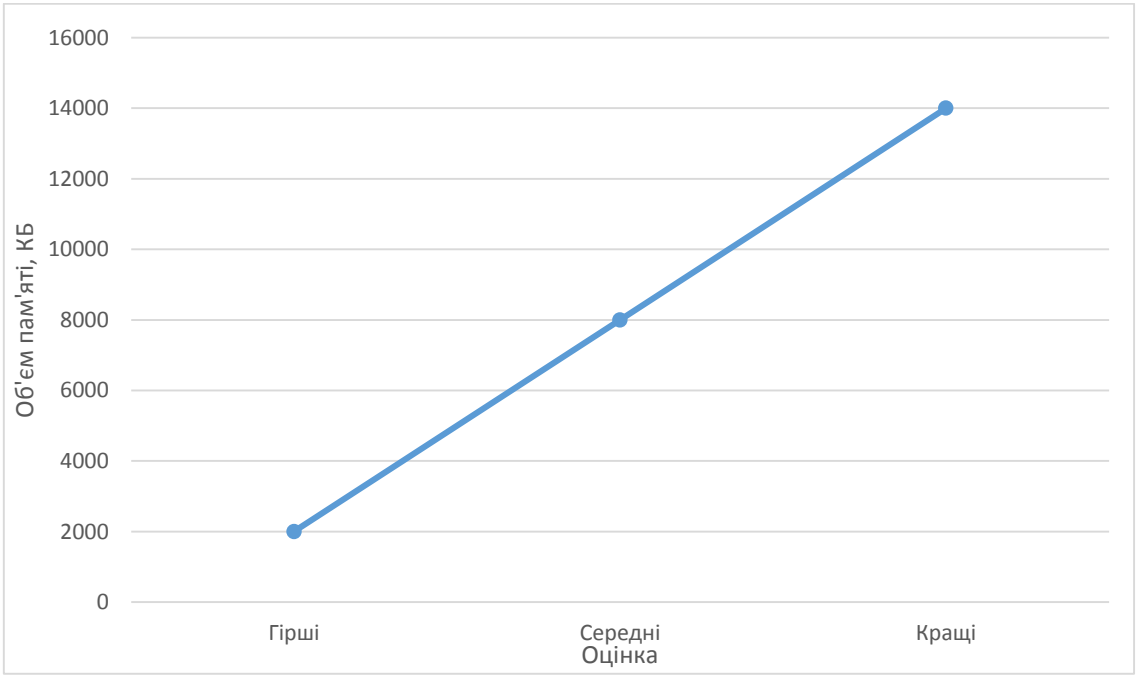

Рисунок 4.2 – X1, Об'єм оперативної пам’яті

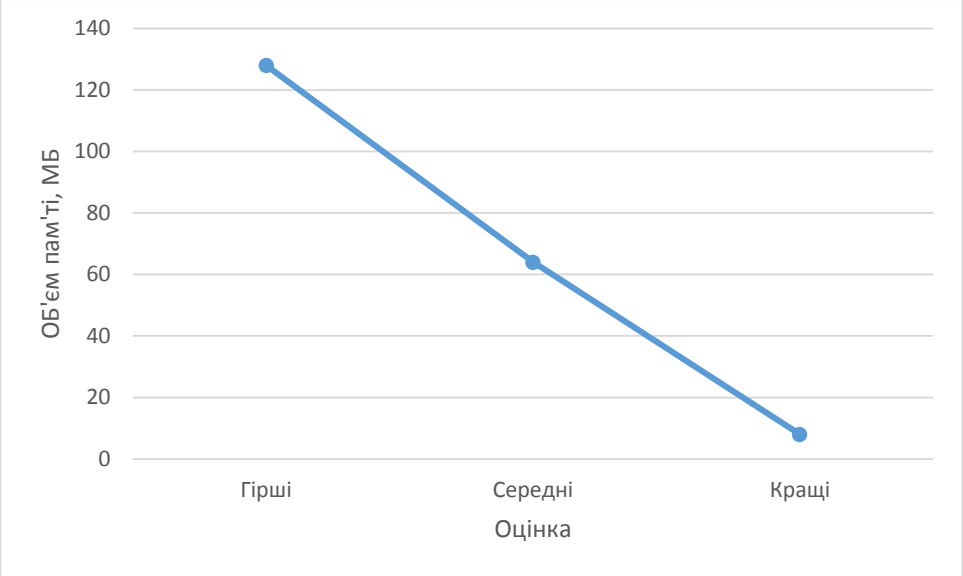

Рисунок 4.3 – X2, об’єм пам’яті для збереження даних

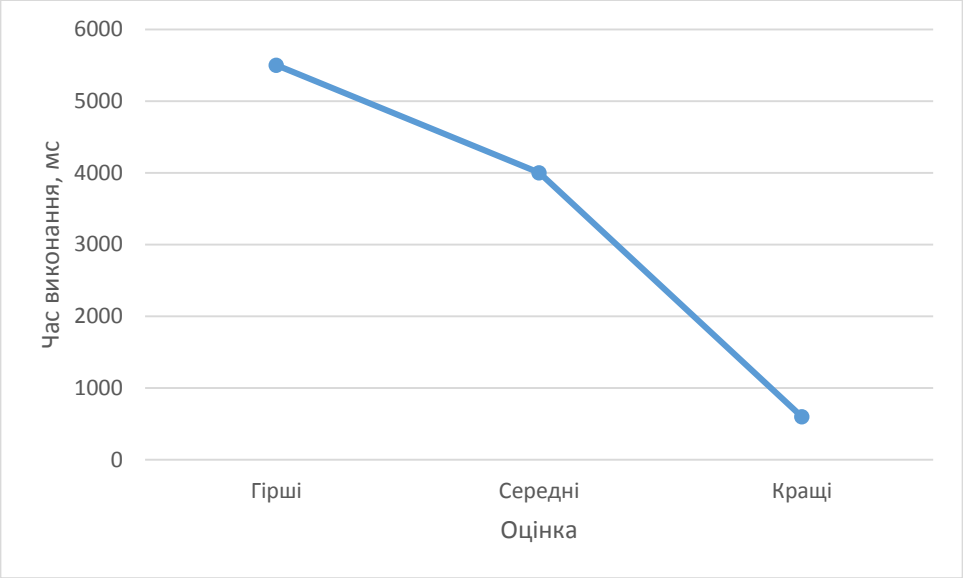

Рисунок 4.4 – X3, час обробки даних алгоритмом

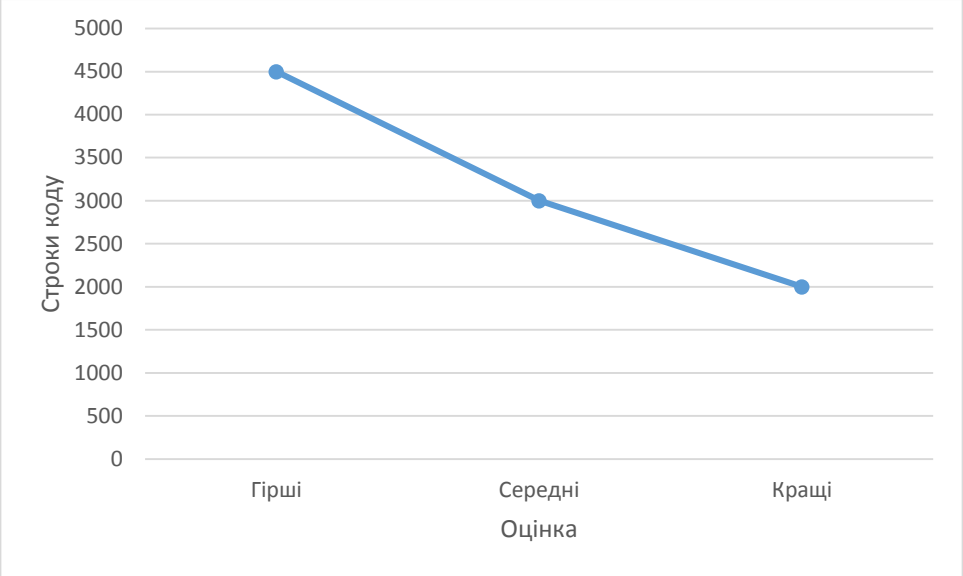

Рисунок 4.5 – X4, потенційний об’єм програмного коду

4.2.3 Аналіз експертного оцінювання параметрів

Після детального обговорення й аналізу кожний експерт оцінює ступінь важливості кожного параметру для конкретно поставленої цілі – розробка програмного продукту, який дає найбільш точні результати при знаходженні параметрів моделей адаптивного прогнозування і обчислення прогнозних значень.

Значимість кожного параметра визначається методом попарного порівняння. Оцінку проводить експертна комісія із 7 людей. Визначення коефіцієнтів значимості передбачає:

– визначення рівня значимості параметра шляхом присвоєння різних рангів;

– перевірку придатності експертних оцінок для подальшого використання;

– визначення оцінки попарного пріоритету параметрів;

– обробку результатів та визначення коефіцієнту значимості.

Результати експертного ранжування наведені у таблиці 4.3.

Таблиця 4.3 – Результати ранжування параметрів

Позначення параметра	Назва параметра	Одиниці вимірy	Ранг параметра за оцінкою експерта							Сума рангів Ri	Відхилення Δi	Δi2
			1	2	3	4	5	6	7			
X1	Об'єм оперативної пам’яті	МБ	3	1	2	2	2	1	2	13	-4.5	20.25
X2	Об’єм пам’яті для збереження даних	МБ	4	4	4	4	3	4	4	27	9.5	90.25
X3	Час обробки даних алгоритмом	Мс	1	3	3	3	4	3	3	20	2.5	6.25
X4	Потенційний об’єм програмного коду	кількість строк коду	2	2	1	1	1	2	1	10	-7.5	56.25
	Разом		10	10	10	10	10	10	10	70	0	173

Для перевірки степені достовірності експертних оцінок, визначимо наступні параметри:

а) сума рангів кожного з параметрів і загальна сума рангів:

$$R_i = \sum_{j=1}^{N} r_{ij} R_{ij} = \frac{Nn(n+1)}{2} = 70, \qquad (1)$$

де N – число експертів, n – кількість параметрів;

б) середня сума рангів:

$$T = \frac{1}{n} R_{ij} = 17.5.$$

в) відхилення суми рангів кожного параметра від середньої суми рангів:

$$\Delta_i = R_i - T$$

Сума відхилень по всім параметрам повинна дорівнювати 0;

г)загальна сума квадратів відхилення:

$$S = \sum_{i=1}^{N} \Delta_i^2 = 173.$$

Порахуємо коефіцієнт узгодженості:

$$W = \frac{12S}{N^2(n^3 - n)} = \frac{12 \cdot 173}{7^2(4^3 - 4)} = 0{,}71 > W_k = 0{,}67$$

Ранжування можна вважати достовірним, тому що знайдений коефіцієнт узгодженості перевищує нормативний, котрий дорівнює 0,67.

Скориставшись результатами ранжирування, проведемо попарне порівняння всіх параметрів і результати занесемо у таблицю4.4.

Таблиця4.4 – Попарне порівняння параметрів

Параметри	Експерти							Кінцева оцінка	Числове значення
	1	2	3	4	5	6	7		
X1 і X2	>	>	>	>	>	>	>	>	1,5
X1 і X3	<	>	>	>	>	>	>	>	1,5
X1 і X4	<	>	<	<	<	>	<	<	0,5
X2 і X3	<	<	<	<	>	<	<	<	0,5
X2 і X4	<	<	<	<	<	<	<	<	0,5
X3 і X4	>	<	<	<	<	<	<	<	0,5

З отриманих числових оцінок переваги складемо матрицю А=$\| a_{ij} \|$.

Для кожного параметра зробимо розрахунок вагомості $K_{вi}$ за наступними формулами:

$$K_{\text{вi}} = \frac{b_i}{\sum_{i=1}^{n} b_i}, \text{де } b_i = \sum_{i=1}^{N} a_{ij}. \qquad (2)$$

Відносні оцінки розраховуються декілька разів доти, поки наступні значення не будуть незначно відрізнятися від попередніх (менше 2%).На другому і наступних кроках відносні оцінки розраховуються за наступними формулами:

$$K_{\text{вi}} = \frac{b_i'}{\sum_{i=1}^{n} b_i'}, \text{де } b_i' = \sum_{i=1}^{N} a_{ij} b_j. \qquad (3)$$

Як видно з таблиці 4.5, різниця значень коефіцієнтів вагомості не перевищує 2%, тому більшої кількості ітерацій не потрібно.

Таблиця 4.5 – Розрахунок вагомості параметрів

Параметриx_i	Параметриx_j				Перша ітер.		Друга ітер.		Третя ітер	
	X1	X2	X3	X4	b_i	$K_{\text{вi}}$	b_i^1	$K_{\text{вi}}^1$	b_i^2	$K_{\text{вi}}^2$
X1	1.0	1.5	1.5	0.5	4.5	0.281	20.3	0.29	91.1	0.29
X2	00.5	1.0	0.5	0.5	2.5	0.156	6.25	0.09	15.6	0.05
X3	0.5	1.5	1.0	1.5	3.5	0.219	12.3	0.18	42.9	0.14
X4	1.5	1.5	1.5	1.0	5.5	0.344	30.3	0.44	166	0.53
Всього:					16	1	69	1	316	1

4.3 Аналіз рівня якості варіантів реалізації функцій

Визначаємо рівень якості кожного варіанту виконання основних функцій окремо.

Абсолютні значення параметрів *X2*(об’єм пам’яті для збереження даних) та *X1* (об’єм оперативної пам’яті) відповідають технічним вимогам умов функціонування даного ПП.

Абсолютне значення параметра *X4 (кількість строк коду*) обрано не найгіршим (не максимальним), тобто це значення відповідає або варіанту а) 4000 або варіанту б) 2000 мс.

Коефіцієнт технічного рівня для кожного варіанта реалізації ПП розраховується так (таблиця 4.6):

$$K_K(j) = \sum_{i=1}^{n} K_{вi,j} B_{i,j}\,, \qquad (4)$$

де n – кількість параметрів;

$K_{вi}$– коефіцієнт вагомості i–го параметра;

B_i – оцінка i–го параметра в балах.

Таблиця 4.6 – Розрахунок показників рівня якості варіантів реалізації основних функцій ПП

Основні функції	Варіант реалізації функції	Абсолютне значення параметра	Бальна оцінка параметра	Коефіцієнт вагомості параметра	Коефіцієнт рівня якості
F1(X1)	А	800	3.25	0.2884	0.9373
F2(X2)	А	64	6.4	0.0494	0.9373
F3(X3,X4)	А	4000	5	0.1357	0.6785
	Б	2000	2.5	0.5265	1.31625

За даними з таблиці 4.6 за формулою

$$K_K = K_{\text{ТУ}}[F_{1k}] + K_{\text{ТУ}}[F_{2k}] + \ldots + K_{\text{ТУ}}[F_{zk}],$$

визначаємо рівень якості кожного з варіантів:

КК1 = 0.9373 + 0.9373 + 0.6785 = 1.932

КК2 = 0.9373 + 0.9373 + 1.31625= 2.5697

Як видно з розрахунків, кращим є другий варіант, для якого коефіцієнт технічного рівня має найбільше значення.

4.4 Економічний аналіз варіантів розробки ПП

Для визначення вартості розробки ПП спочатку проведемо розрахунок трудомісткості.

Всі варіанти включають в себе два окремих завдання:

1. Розробка проекту програмного продукту;

2. Розробка програмної оболонки;

Завдання 1 за ступенем новизни відноситься до групи А, завдання 2 – до групи Б. За складністю алгоритми, які використовуються в завданні 1 належать до групи 1; а в завданні 2 – до групи 3.

Для реалізації завдання 1 використовується довідкова інформація, а завдання 2 використовує інформацію у вигляді даних.

Проведемо розрахунок норм часу на розробку та програмування для кожного з завдань.

Проведемо розрахунок норм часу на розробку та програмування для кожного з завдань. Загальна трудомісткість обчислюється як

$$T_O = T_P \cdot K_П \cdot K_{СК} \cdot K_M \cdot K_{СТ} \cdot K_{СТ.М}, \quad (5.1)$$

Де:

T_P – трудомісткість розробки ПП;

$K_П$ – поправочний коефіцієнт;

$K_{СК}$ – коефіцієнт на складність вхідної інформації;

K_M – коефіцієнт рівня мови програмування;

$K_{СТ}$ – коефіцієнт використання стандартних модулів і прикладних програм;

$K_{СТ.М}$ – коефіцієнт стандартного математичного забезпечення

Для першого завдання, виходячи із норм часу для завдань розрахункового характеру степеню новизни А та групи складності алгоритму 1, трудомісткість дорівнює: T_P =90 людино-днів. Поправочний коефіцієнт, який враховує вид перемінної інформації для першого завдання: $K_П$ = 2,84. Поправочний коефіцієнт, який враховує складність контролю вхідної та вихідної інформації для всіх семи завдань рівний 1: $K_{СК}$ = 1. Оскільки при розробці першого завдання

використовуються стандартні модулі, врахуємо це за допомогою коефіцієнта $К_{СТ}$ = 0.8. Тоді, за формулою 5.1, загальна трудомісткість програмування першого завдання дорівнює:

$Т_1$ = 90·2.84·0.8 = 204,48 людино-днів.

Проведемо аналогічні розрахунки для подальших завдань.

Для другого завдання (використовується алгоритм третьої групи складності, степінь новизни Б), тобто $Т_Р$ =27 людино-днів, $К_П$ =0.9,$К_{СК}$ = 1,$К_{СТ}$ =0.8:

$Т_2$ = 27 · 0.9 · 0.8 = 19.44 людино-днів.

Складаємо трудомісткість відповідних завдань для кожного з обраних варіантів реалізації програми, щоб отримати їх трудомісткість:

$Т_I$ = (204,48 + 19.44) · 8 = 1791,36 людино-годин;

В розробці беруть участь два програмісти з окладом 6000 грн. Визначимо зарплату за годину за формулою:

$$С_Ч = \frac{М}{T_m \cdot t} \text{ грн.}, \qquad (5.2)$$

де М – місячний оклад працівників;T_m – кількість робочих днів тиждень;t – кількість робочих годин в день.

$$С_Ч = \frac{6000 + 6000}{2 \cdot 21 \cdot 8} = 35{,}714 \text{ грн.}$$

Тоді, розрахуємо заробітну плату за формулою

$$С_{ЗП} = С_Ч \cdot Т_i \cdot К_Д, \qquad (5.3)$$

де $С_Ч$– величина погодинної оплати праці програміста;$Т_i$ – трудомісткість відповідного завдання; $К_Д$ – норматив, який враховує додаткову заробітну плату.

Зарплата розробників становить:

$С_{ЗП}$ = 35,714· 1791,36· 1.2 = 76771.95725 грн.

Відрахування на єдиний соціальний внесок становить 22%:

$С_{ВІД}$ = СЗП · 0.22 =76771.95725 · 0.22 = 16889.83 грн.

Тепер визначимо витрати на оплату однієї машино-години. (СМ)

Так як одна ЕОМ обслуговує одного програміста з окладом 6000 грн., з коефіцієнтом зайнятості 0,2 то для однієї машини отримаємо:

$С_Г$ = 12·М·$К_З$ = 12 · 6000· 0,2 = 14400 грн.

З урахуванням додаткової заробітної плати:

$С_{ЗП} = С_Г \cdot (1 + К_З) = 14400 \cdot (1 + 0.2) = 17280$ грн.

Відрахування на єдиний соціальний внесок:

$С_{ВІД} = С_{ЗП} \cdot 0,22 = 17280 \cdot 0,22 = 3806,6$ грн.

Амортизаційні відрахування розраховуємо при амортизації 25% та вартості ЕОМ – 8000 грн.

$С_А = К_{ТМ} \cdot К_А \cdot Ц_{ПР} = 1.15 \cdot 0.25 \cdot 8000 = 2300$ грн.,

де $К_{ТМ}$– коефіцієнт, який враховує витрати на транспортування та монтаж приладу у користувача; $К_А$– річна норма амортизації; $Ц_{ПР}$– договірна ціна приладу.

Витрати на ремонт та профілактику розраховуємо як:

$С_Р = К_{ТМ} \cdot Ц_{ПР} \cdot К_Р = 1.15 \cdot 8000 \cdot 0.05 = 460$ грн.,

де $К_Р$– відсоток витрат на поточні ремонти.

Ефективний годинний фонд часу ПК за рік розраховуємо за формулою:

$Т_{ЕФ} = (Д_К – Д_В – Д_С – Д_Р) \cdot t_З \cdot К_В = (365 – 104 – 8 – 16) \cdot 8 \cdot 0.9 = 1706.4$ годин,

де $Д_К$ – календарна кількість днів у році;

$Д_В$, $Д_С$ – відповідно кількість вихідних та святкових днів;

$Д_Р$ – кількість днів планових ремонтів устаткування;

t –кількість робочих годин в день;

$К_В$– коефіцієнт використання приладу у часі протягом зміни.

Витрати на оплату електроенергії розраховуємо за формулою:

$С_{ЕЛ} = Т_{ЕФ} \cdot N_С \cdot К_З \cdot Ц_{ЕН} = 1706,4 \cdot 0,156 \cdot 0,2 \cdot 1,506 = 80,17$ грн.,

де $N_С$ – середньо-споживча потужність приладу;

$К_З$– коефіцієнтом зайнятості приладу;

$Ц_{ЕН}$ – тариф за 1 КВт-годин електроенергії.

Накладні витрати розраховуємо за формулою:

$С_Н = Ц_{ПР} \cdot 0.67 = 8000 \cdot 0,67 = 5360$ грн.

Тоді, річні експлуатаційні витрати будуть:

$$С_{ЕКС} = С_{ЗП} + С_{ВІД} + С_А + С_Р + С_{ЕЛ} + С_Н \qquad (5.4)$$

$C_{ЕКС}$ = 17280 + 3806,6 + 2300 + 460 + 80,17 + 5360 = 29286,77 грн.

Собівартість однієї машино-години ЕОМ дорівнюватиме:

$C_{М\text{-}Г} = C_{ЕКС} / T_{ЕФ}$ = 29286,77 /1706,4 = 17,162 грн/год.

Оскільки в даному випадку всі роботи, які пов‘язані з розробкою програмного продукту ведуться на ЕОМ, витрати на оплату машинного часу, в залежності від обраного варіанта реалізації, складає:

$$C_М = C_{М\text{-}Г} \cdot T \qquad (5.5)$$

I. $C_М$ = 17,162 · 1791,36= 30743.32 грн.;

Накладні витрати складають 67% від заробітної плати:

$$C_Н = C_{ЗП} \cdot 0{,}67$$

I. $C_Н$ = 66437,31· 0,67 = 44513 грн.;

Отже, вартість розробки ПП становить:

$$C_{ПП} = C_{ЗП} + C_{ВІД} + C_М + C_Н \qquad (5.6)$$

I. $C_{ПП}$ = 66437,31 + 3806,6 + 30743,32 + 44513 = 145500.23 грн.;

4.5 Вибір кращого варіанта ПП техніко-економічного рівня

Розрахуємо коефіцієнт техніко-економічного рівня за формулою:

$$К_{ТЕРj} = К_{Кj} / С_{Фj}, \qquad (6)$$

$К_{ТЕР1}$ = 1.932/145500,23 = $1.326559 \cdot 10^{-5}$;

$К_{ТЕР2}$ = 2.5697/145500,23 = $1.766195616 \cdot 10^{-5}$;

Як бачимо, найбільш ефективним є перший варіант реалізації програми з коефіцієнтом техніко-економічного рівня $К_{ТЕР1}$= $1.766195616 \cdot 10^{-5}$.

4.6 Висновки до розділу 4

В даному розділі проведено повний функціонально-вартісний аналіз ПП, який було розроблено для аналізу ефективності методів інтелектуального аналізу тексту. Процес аналізу складається з двох частини.

В першій з них проведено дослідження ПП з технічної точки зору: було визначено основні вимоги до ПП та сформовано множину варіантів їх реалізації; на основі обчислених значень параметрів, а також експертних оцінок їх важливості було обчислено коефіцієнт технічного рівня, який і дав змогу визначити оптимальну з технічної точки зору альтернативу реалізації функцій ПП.

У другій частину ФВА проводиться вибір із альтернативних варіантів реалізації найбільш економічно обґрунтованого. Порівняння робились з урахуванням витрат на заробітні плати, електроенергії, накладні витрати.

Після виконання функціонально-вартісного аналізу програмного комплексу що розроблюється, для аналізу ефективності методів інтелектуального аналізу тексту можна зробити висновок, що з альтернатив, що залишились після першого відбору двох варіантів виконання програмного комплексу оптимальним є перший варіант реалізації програмного продукту. У нього виявився найкращий показник техніко-економічного рівня якості $К_{ТЕР} = 1.766195616 \cdot 10^{-5}$.

Цей варіант реалізації програмного продукту має такі параметри:

– мова програмування – Java;

– виведення даних в таблиці Excel;

– консольний інтерфейс користувача.

Даний варіант виконання програмного комплексу дає користувачу зручний інтерфейс, непоганий функціонал і швидкодію.

ВИСНОВКИ

В результаті виконання даної роботи була здійснена розробка системи інтелектуального аналізу текстової інформації. Дана система є фундаментом для подальшого дослідження даної предметної області.

Тема структурування знань з використанням інтелектуального аналізу текстової інформації є досить цікавою і представляє собою широке поле для подальших досліджень в галузі Text Mining. Тому в роботі були висвітлені основні вимоги до такого роду систем.

У першому розділі було висвітлено такі етапи процесу аналізу як: попередня обробка, виділення ключових слів та класифікації та кластеризації текстових документів. На першому етапі розглянуто алгоритми стемінгу, виділення N-грам, а також більш детально проаналізовано алгоритми виявлення та виділення стоп-слів такі як: словниковий, статистичний на основі об'єднання та побудований на основі Y-інтерпретації закону Бредфорда. Було виявлено, що останні можуть мати невелику точність, а перші – повноту видалення. На другому етапі класифіковано алгоритми пошуку стоп-слів на статистичні, лінгвістичні та гібридні. Встановлено, що найбільший потенціал та ефективність мають саме гібридні методи, зокрема метод лінгво-статистичних шаблонів. На третьому етапі висвітлено методи класифікації та кластеризації.

У другому розділі описано бібліотеки, що використовувалися про розробці програмного засобу аналізу методів інтелектуального аналізу текстів. Snowball Stemmer, побудований на основі стемера Портера використовує N—грами, що підвищує ефективність роботи даного програмного продукту. Дуже потужною є розробка Apache OpenNLP, яка допомагає виконати токенізацію та визначення частин мови, що необхідно для аналізу методу лінгво-статистичних шаблонів. Слід виділити також KH Coder, який використовується для досліджень у даній галузі по всьому світу.

У третьому розділі було проведено безпосередній аналіз методів, описаних у першому розділі. При аналізі методів виділення стоп-слів було виявлено, що метод, який працює на основі Y-інтерпретації закону Бредфорда чисельно продемонстрував доволі високу точність (близько 85%) проте при більш детальному аналізі було виявлено, що цей метод видаляє найбільш значущі ключові із текстів, що були проаналізовані, що робить даний метод непридатним для використання у такого роду системах у даному вигляді. Словниковий метод не є універсальним, оскільки універсальний словник робить цей метод менш точним (погіршує результат подальшого виявлення колокацій) та менш повним. Тому для використання даного методу варто використовувати словник стоп-слів, розроблений саме для даної предметної області. Було запропоновано скомбінувати ці методи при аналізі великої кількості текстів різної тематики та направленості. Аналізуються слова, що зустрічаються у цих текстах найчастіше та відбираються за Бредфордом. Чим більша варіативність тематики текстів, тим більша якість словника. При аналізі методів виявлення ключових слів, було використано 3 алгоритми: 2 статистичні: міра TF-IDF та F-міра та гібридний – метод лінгво-статистичних шаблонів. При ручному аналізі результатів було виявлено, що F-міра показала значно гірші результати за TF-IFD. Значно складнішим у реалізації та більш ресурсозатратним в плані виконання є алгоритм лінгво-статистичних шаблонів, проте саме цей алгоритм показав найкращий результат, що було показано у 3 розділі. Для фінального структурування знань існує 2 найпоширеніших методи – Naïve Bayes та метод Роше.

Перспективою подальших досліджень є розробка повноцінної системи структурування знань з використанням інтелектуального аналізу текстової інформації з використанням модифікованих алгоритмів та словників її розгортання на веб-сервері.

ПЕРЕЛІК ПОСИЛАНЬ

1. Яцко В.А. История вычислительной техники и информатики. — Абакан: Издательство ФГБОУ ВПО «Хакасский Государственный Университет им Н.Ф. Катанова», 2013. — 88 с.

2. Ефремова Н.Э., Большакова Е.И., Носков А.А., Антонов В.Ю. Терминологический анализ текста на основе лексико-синтаксических шаблонов / Ефремова Н.Э., Большакова Е.И., Носков А.А., Антонов В.Ю. //Компьютерная лингвистика и интеллектуальные технологии: По материалам ежегодной Международной конференции «Диалог» 26 - 30 мая 2010 р. Бекасово, Росія: матеріали – 2010. – С.124 -129

3. Абрамов, Е.Г. Подбор ключевых слов для научной статьи //Научная периодика: проблемы и решения. - 2011. - № 2. - С. 35−40.

4. Зиберт А.О., Хрусталев В.И. Разработка системы определения наличия заимствований в работах студентов высших учебных заведений. Алгоритмы поиска нечетких дубликатов // Universum: Технические науки : электрон. научн. журн. 2014. № 3 (4)/ [Электронный ресурс]. — Режим доступа: URL: http://7universum.com/ru/tech/archive/item/1139 (дата обращения: 26.05.2016).

5. Нога Р. Аналітичний огляд методів та засобів опрацювання текстової інформації / Р. Нога, Н. Б. Шаховська // Вісн. Нац. ун-ту "Львів. політехніка". - 2011. - № 715. - С. 323-332.

6. Сахарный Л. В., Штерн А. С. Набор ключевых слов как тип текста // Лексические аспекты в системе профессионально-ориентированного обучения иноязычной речевой деятельности. Пермь: Пермский политехнический ун-т, 1988. С. 34—51.

7. Liu Z., Huang W., Zheng Y., Sun M. Automatic keyphrase extraction via topic decomposition. Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing. Cambridge, Massachusetts, 2010, pp. 366–376.

8. Большакова Е.И., Клышинский Э.С., Ландэ Д.В., Носков А.А., Пескова О.В., Ягунова Е.В. Автоматическая обработка текстов на естественном языке и компьютерная лингвистика : учеб. пособие / Большакова Е.И., Клышинский Э.С., Ландэ Д.В., Носков А.А., Пескова О.В., Ягунова Е.В. — М.: МИЭМ, 2011. — 272 с.

9. Michael W. Berry and Malu Castellanos, Survey of Text Mining: Clustering, Classification, and Retrieval, Second Edition / Michael W. Berry and Malu Castellanos - Springer 2007. – 254p.

10. Yatsko, V.A. TF*IDF Revisited [Electronic resource] / V.A. Yatsko // International journal of computational linguistics and natural language engineering. – 2013. – Vol. 2, Issue 6. – P. 385-387. – URL: http://www.ijclnlp.org/ vol2issue6/paper60.pdf (дата доступу: 01.05.2016).

11. Барсегян, А. А. Анализ данных и процессов: учеб. пособие / А. А. Барсегян, М. С. Куприянов, И. И. Холод, М. Д. Тесс, С. И. Елизаров. — 3-е изд., перераб. и доп. — СПб.: БХВ-Петербург, 2009. — 512 с.

12. Захаров, В.П. Автоматическое выявление терминологических словосочетаний /В.П. Захаров, М.В. Хохлова //Структурная и прикладная лингвистика. Вып.10. - Санкт-Петербург: Изд-во С.-Петерб. ун-та, 2014. – С. 182-200.

13. Sato S., Sasaki Y. Automatic Collection of Related Terms from the Web // The Companion Volume to the Proceedings of 41st Annual Meeting of the ACL, Sapporo, Japan, 2003. – P. 121–124.

14. Drott M. C., Griffith B. C. An Empirical Examination of Bradford’s Law and the Scattering of Scientific Literature // Journal of the American Society for Information Science. 1978. Vol. 29, Iss. 5. P. 238–246.

15. Salton G. On the Specification of Term Values in Automatic Indexing. Journal of Documentation. 1973, vol. 29, no. 4, pp. 351–372.

16. Bradford, S.C. "Sources of Information on Specific Subjects". Engineering: Ant Illustrated Weekly Journal (London), 137, 1934 (26 January). – P. 85-86.

17. Яцко В.А., Стариков М.С., Ларченко Е.В. Алгоритмы предварительной обработки текста: декомпозиция, аннотирование, морфологический анализ [Текст] / В.А. Яцко, М.С. Стариков, Е.В. Ларченко [и др.] // Научно-техническая информация. Сер. 2. –2009. – № 11. – С. 8-18.

18. Яцко, В.А. Лексикографические ресурсы для автоматического анализа текста [Текст] / В.А. Яцко // Вестник Иркутского государственного лингвистического университета. – 2013. – №2. – С. 19-24.

19. Баранов, А.Н. Введение в прикладную лингвистику [Текст]: учеб. особие / А.Н. Баранов. – М.: Эдиториал УРСС, 2001. – 360 с.

Додаток А. Програмний код

```
package model.entity;

import java.io.FileOutputStream;
import java.io.IOException;
import java.util.ArrayList;
import java.util.Arrays;
import java.util.Collections;
import java.util.Comparator;
import java.util.HashSet;
import java.util.Random;
import java.util.Set;
import java.util.logging.Level;
import java.util.logging.Logger;
import org.apache.poi.hssf.usermodel.HSSFCell;
import org.apache.poi.hssf.usermodel.HSSFRow;
import org.apache.poi.hssf.usermodel.HSSFSheet;
import org.apache.poi.hssf.usermodel.HSSFWorkbook;
import service.prepare.TextPrepare;
import service.util.WordComparator;

/**
 *
 * @author PRIEST
 */
public class Text {

    private int id;
    private String name;
    private String preperedText;
    private String lingMatkUp;
    private int setOfTextsId;
    private int wordsAmount;
    private ArrayList<Paragraph> paragraphs = new ArrayList<Paragraph>();
    private ArrayList<Word> words = new ArrayList<Word>();
    private Set<String> stopWordsBradford = new HashSet<String>();
   // private Set<String> stopWordsDict = new HashSet<String>();
    private ArrayList<Word> stopWordsTotal = new ArrayList<Word>();


   public void uniteWords(Word word){
    for (Word stopWord: stopWordsTotal){
        if(stopWord.getWord().equalsIgnoreCase(word.getWord())){
            stopWord.setAmount(word.getAmount()+stopWord.getAmount());
            System.out.println("summ" + word.getWord());
        } else{
        stopWordsTotal.add(word);
         System.out.println("add" + word.getWord());
    }
    }
   }

    public void sortWords() {
        Comparator<Word> wc = new WordComparator();
        Collections.sort(words, wc);

    }
     public void getStopWordByDictToExcel(){
         int totalAmount = 0;
```

```
        for(Word word: words){
            for(String dict: StopWordsList.getList())
            {
                if(word.getWord().equalsIgnoreCase(dict)){
                  // System.out.println(dict + " " + word.getAmount());
                   totalAmount+=word.getAmount();
                   break;
                }
            }
        }
         System.out.println(this.wordsAmount + " " +  totalAmount);
    }

    public void processWordsStemming() {
        ArrayList<Paragraph> paragraphs = this.getParagraphs();
        for (Paragraph paragraph : paragraphs) {
            paragraph.setParagraphString(TextPrepare.deletePunctuation(paragraph.getParagraphString()));
            paragraph.setWords(TextPrepare.splitWords(paragraph.getParagraphString()));

        }
        for (Paragraph paragraph : paragraphs) {
            ArrayList<String> strTest = new ArrayList<String>();

            strTest.addAll(paragraph.getWords());
            for (String str : strTest) {
                String stemmedWord = TextPrepare.stemm(str);
//String stemmedWord = str;
paragraph.getStemmedText().add(stemmedWord);
                Word word = this.findWord(stemmedWord);
                if (word.hasForm(str)) {
                    word.incAmount();
                } else {
                    word.addForm(str);
                    word.incAmount();
                }
                this.incWordsAmount();
            }

        }
    }


    public void deteleStopWords(){

    }
    public void toExcel() {
        try {
            FileOutputStream fileOut = null;
            if (this.name != "") {
                fileOut = new FileOutputStream(name + ".xls");
            } else {
                Random rand = new Random();
                fileOut = new FileOutputStream(Integer.toString(rand.nextInt()) + ".xls");
            }

            HSSFWorkbook workbook = new HSSFWorkbook();
            HSSFSheet worksheet = workbook.createSheet("stat");
            HSSFRow row1 = worksheet.createRow(0);
            HSSFCell cell01 = row1.createCell(0);
            cell01.setCellValue("Word");
            HSSFCell cell02 = row1.createCell(1);
```

```
        cell02.setCellValue("Amount");
        HSSFCell cell03 = row1.createCell(2);
        cell03.setCellValue("Percentage");

        int i = 1;
        for (Word word : this.getWords()) {

            HSSFRow row = worksheet.createRow(i++);
            HSSFCell cell0 = row.createCell(0);
            cell0.setCellValue(word.getWord());
            HSSFCell cell1 = row.createCell(1);
            cell1.setCellValue(word.getAmount());
            HSSFCell cell2 = row.createCell(2);

            cell2.setCellValue(100 * (double) word.getAmount() / (double) this.getWordsAmount());

        }

        workbook.write(fileOut);
        fileOut.flush();
        fileOut.close();
    } catch (IOException ex) {
        Logger.getLogger(Text.class.getName()).log(Level.SEVERE, null, ex);
    }
    stopWordsBradfordtoExcel();
}

public void stopWordsBradfordtoExcel() {
    try {
        FileOutputStream fileOut = null;
        if (this.name != "") {
            fileOut = new FileOutputStream(name + "Brad.xls");
        } else {
            Random rand = new Random();
            fileOut = new FileOutputStream(Integer.toString(rand.nextInt()) + "Brad.xls");
        }

        HSSFWorkbook workbook = new HSSFWorkbook();
        HSSFSheet worksheet2 = workbook.createSheet("Bradford");
        HSSFRow row11 = worksheet2.createRow(0);
        HSSFCell cell11 = row11.createCell(0);
        cell11.setCellValue("Word");
        HSSFCell cell12 = row11.createCell(1);
        cell12.setCellValue("Amount");
        HSSFCell cell13 = row11.createCell(2);
        cell13.setCellValue("Percentage");
        int i=0;
        for (int j = 0; j < wordsAmount/3; j+=words.get(i).getAmount()) {

            HSSFRow row2 = worksheet2.createRow(i+1);
            HSSFCell cell20 = row2.createCell(0);
            cell20.setCellValue(words.get(i).getWord());
            HSSFCell cell21 = row2.createCell(1);
            cell21.setCellValue(words.get(i).getAmount());
            HSSFCell cell22 = row2.createCell(2);
          //    System.out.println(words.get(i).getAmount() + "   " + words.get(i).getWord());
            cell22.setCellValue(100 * (double) words.get(i).getAmount() / (double) this.getWordsAmount());
         i++;
        }

         for (int j = wordsAmount/3; j < 2*wordsAmount/3; j+=words.get(i).getAmount()) {
```

```
            HSSFRow row2 = worksheet2.createRow(i+1);
            HSSFCell cell20 = row2.createCell(4);
            cell20.setCellValue(words.get(i).getWord());
            HSSFCell cell21 = row2.createCell(5);
            cell21.setCellValue(words.get(i).getAmount());
            HSSFCell cell22 = row2.createCell(6);
          //   System.out.println(words.get(i).getAmount() + "   " + words.get(i).getWord());
            cell22.setCellValue(100 * (double) words.get(i).getAmount() / (double) this.getWordsAmount());
         i++;
        }
         for (int j = 2*wordsAmount/3; j < wordsAmount; j+=words.get(i).getAmount()) {

            HSSFRow row2 = worksheet2.createRow(i+1);
            HSSFCell cell20 = row2.createCell(8);
            cell20.setCellValue(words.get(i).getWord());
            HSSFCell cell21 = row2.createCell(9);
            cell21.setCellValue(words.get(i).getAmount());
            HSSFCell cell22 = row2.createCell(10);
          //   System.out.println(words.get(i).getAmount() + "   " + words.get(i).getWord());
            cell22.setCellValue(100 * (double) words.get(i).getAmount() / (double) this.getWordsAmount());
         i++;
        }

        workbook.write(fileOut);
        fileOut.flush();
        fileOut.close();
    } catch (IOException ex) {
        Logger.getLogger(Text.class.getName()).log(Level.SEVERE, null, ex);
    }

}

public void printAllWords() {

    for (Word word : words) {
        word.printWord();
        System.out.print("   " + Double.toString(100 * (double) word.getAmount() / (double) wordsAmount) + "%");
    }
    System.out.print("/nTotal: " + wordsAmount + " words");
}

public Word findWord(String stemmed) {
    for (Word word : words) {
        if (word.getWord().equalsIgnoreCase(stemmed)) {
            return word;
        }
    }
    this.words.add(new Word(stemmed));
    return findWord(stemmed);
}

public void incWordsAmount() {
    this.wordsAmount++;
}

public void setWords(ArrayList<Word> words) {
    this.words = words;
}

public ArrayList<Word> getWords() {
```

```
        return words;
    }

    public int getId() {
        return id;
    }

    public String getName() {
        return name;
    }

    public String getPreperedText() {
        return preperedText;
    }

    public String getLingMatkUp() {
        return lingMatkUp;
    }

    public int getSetOfTextsId() {
        return setOfTextsId;
    }

    public int getWordsAmount() {
        return wordsAmount;
    }

    public ArrayList<Paragraph> getParagraphs() {
        return paragraphs;
    }

    public void setId(int id) {
        this.id = id;
    }

    public void setName(String name) {
        this.name = name;
    }

    public void setPreperedText(String preperedText) {
        this.preperedText = preperedText;
    }

    public void setLingMatkUp(String lingMatkUp) {
        this.lingMatkUp = lingMatkUp;
    }

    public void setSetOfTextsId(int setOfTextsId) {
        this.setOfTextsId = setOfTextsId;
    }

    public void setWordsAmount(int wordsAmount) {
        this.wordsAmount = wordsAmount;
    }

    public void setParagraphs(ArrayList<Paragraph> paragraphs) {
        this.paragraphs = paragraphs;
    }

}
```

```
/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package service.prepare;

import config.SplitterConfig;
import java.util.ArrayList;
import java.util.logging.Level;
import java.util.logging.Logger;
import model.entity.Paragraph;
import model.entity.Word;
import org.tartarus.snowball.SnowballStemmer;
import org.tartarus.snowball.ext.englishStemmer;

/**
 *
 * @author PRIEST
 */
public class TextPrepare {

    public static ArrayList<Paragraph> uniteParagraphs(ArrayList<Paragraph> list) {

        for (int i = 0; i < list.size(); i++) {

        }
        return list;
    }

    public static String deletePunctuation(String text) {
        text = text.replaceAll(SplitterConfig.punctuationRegExp, "");
        text = text.replaceAll("\\/", " ");
        text = text.replaceAll("\\!", " !");
        text = text.replaceAll("\\.", " .");
        text = text.replaceAll("\\?", " ?");
        return text;
    }

    public static ArrayList<String> splitWords(String text) {
        ArrayList<String> words = new ArrayList<String>();
        String[] strWords = text.split(" ");
        boolean toLowerCase = true;
        for (int i = 0; i < strWords.length; i++) {
            if (strWords[i].equals(".") || strWords[i].equals("!") || strWords[i].equals("?")) {
                toLowerCase = true;
            } else if (toLowerCase) {
                words.add(strWords[i].toLowerCase());
                toLowerCase = false;
            } else {
                words.add(strWords[i]);
            }
        }

        return words;
    }

    public static String stemm(String wordStr) {
        String word = "";

        Class stemClass;
```

```
        try {
            stemClass = Class.forName("org.tartarus.snowball.ext.englishStemmer");
            SnowballStemmer stemmer = (SnowballStemmer) stemClass.newInstance();
            stemmer.setCurrent(wordStr);
            stemmer.stem();
           word = stemmer.getCurrent();

        } catch (ClassNotFoundException ex) {
            Logger.getLogger(TextPrepare.class.getName()).log(Level.SEVERE, null, ex);
        } catch (InstantiationException ex) {
            Logger.getLogger(TextPrepare.class.getName()).log(Level.SEVERE, null, ex);
        } catch (IllegalAccessException ex) {
            Logger.getLogger(TextPrepare.class.getName()).log(Level.SEVERE, null, ex);
        }
            return word;
    }

}
```

Додаток Б. Приклади текстів, що аналізуються в роботі

Big data

Big data is a term for data sets that are so large or complex that traditional data processing applications are inadequate. Challenges include analysis, capture, data curation, search, sharing, storage, transfer, visualization, querying, updating and information privacy. The term often refers simply to the use of predictive analytics or certain other advanced methods to extract value from data, and seldom to a particular size of data set. Accuracy in big data may lead to more confident decision making, and better decisions can result in greater operational efficiency, cost reduction and reduced risk.

Analysis of data sets can find new correlations to "spot business trends, prevent diseases, combat crime and so on" Scientists, business executives, practitioners of medicine, advertising and governments alike regularly meet difficulties with large data sets in areas including Internet search, finance and business informatics. Scientists encounter limitations in e-Science work, including meteorology, genomics, connectomics, complex physics simulations, biology and environmental research.

Data sets are growing rapidly in part because they are increasingly gathered by cheap and numerous information-sensing mobile devices, aerial (remote sensing), software logs, cameras, microphones, radio-frequency identification (RFID) readers and wireless sensor networks. The world's technological per-capita capacity to store information has roughly doubled every 40 months since the 1980s; as of 2012, every day 2.5 exabytes of data are created. One question for large enterprises is determining who should own big data initiatives that affect the entire organization.

Relational database management systems and desktop statistics and visualization packages often have difficulty handling big data. The work instead requires "massively parallel software running on tens, hundreds, or even thousands of servers". What is considered "big data" varies depending on the capabilities of the users and their tools, and expanding capabilities make big data a moving target. "For some organizations, facing hundreds of gigabytes of data for the first time may trigger a need to reconsider data management options. For others, it may take tens or hundreds of terabytes before data size becomes a significant consideration."

Definition

Big data usually includes data sets with sizes beyond the ability of commonly used software tools to capture, curate, manage, and process data within a tolerable elapsed time. Big data "size" is a constantly moving target, as of 2012 ranging from a few dozen terabytes to many petabytes of data. Big data requires a set of techniques and technologies with new forms of integration to reveal insights from datasets that are diverse, complex, and of a massive scale.

In a 2001 research report and related lectures, META Group (now Gartner) analyst Doug Laney defined data growth challenges and opportunities as being three-dimensional, i.e. increasing volume (amount of data), velocity (speed of data in and out), and variety (range of data types and sources). Gartner, and now much of the industry, continue to use this "3Vs" model for describing big data.

In 2012, Gartner updated its definition as follows: "Big data is high volume, high velocity, and/or high variety information assets that require new forms of processing to enable enhanced decision making, insight discovery and process optimization." Gartner's definition of the 3Vs is still widely used, and in agreement with a consensual definition that states that "Big Data represents the Information assets characterized by such a High Volume, Velocity and Variety to require specific Technology and Analytical Methods for its transformation into Value". Additionally, a new V "Veracity" is added by some organizations to describe it,

revisionism challenged by some industry authorities. The 3Vs have been expanded to other complementary characteristics of big data:

Volume: big data doesn't sample; it just observes and tracks what happens

Velocity: big data is often available in real-time

Variety: big data draws from text, images, audio, video; plus it completes missing pieces through data fusion

Machine Learning: big data often doesn't ask why and simply detects patterns

Digital footprint: big data is often a cost-free byproduct of digital interaction

The growing maturity of the concept more starkly delineates the difference between big data and Business Intelligence:

Business Intelligence uses descriptive statistics with data with high information density to measure things, detect trends, etc..

Big data uses inductive statistics and concepts from nonlinear system identification to infer laws (regressions, nonlinear relationships, and causal effects) from large sets of data with low information density to reveal relationships and dependencies, or to perform predictions of outcomes and behaviors.

In a popular tutorial article published in IEEE Access Journal, the authors classified existing definitions of big data into three categories: Attribute Definition, Comparative Definition and Architectural Definition. The authors also presented a big-data technology map that illustrates its key technological evolutions.

Characteristics

Big data can be described by the following characteristics:

Volume

The quantity of generated and stored data. The size of the data determines the value and potential insight- and whether it can actually be considered big data or not.

Variety

The type and nature of the data. This helps people who analyze it to effectively use the resulting insight.

Velocity

In this context, the speed at which the data is generated and processed to meet the demands and challenges that lie in the path of growth and development.

Variability

Inconsistency of the data set can hamper processes to handle and manage it.

Veracity

The quality of captured data can vary greatly, affecting accurate analysis.

Factory work and Cyber-physical systems may have a 6C system:

Connection (sensor and networks)

Cloud (computing and data on demand)

Cyber (model and memory)

Content/context (meaning and correlation)

Community (sharing and collaboration)

Customization (personalization and value)

Data must be processed with advanced tools (analytics and algorithms) to reveal meaningful information. For example, to manage a factory one must consider both visible and invisible issues with various components. Information generation algorithms must detect and address invisible issues such as machine degradation, component wear, etc. on the factory floor.

Architecture

In 2000, Seisint Inc. (now LexisNexis Group) developed a C++-based distributed file-sharing framework for data storage and query. The system stores and distributes structured, semi-structured, and unstructured data across multiple servers. Users can build queries in a C++ dialect called ECL. ECL uses an "apply schema on read" method to infer the structure of stored data when it is queried, instead of when it is stored. In 2004, LexisNexis acquired Seisint Inc. and in 2008 acquired ChoicePoint, Inc. and their high-speed parallel processing platform. The two platforms were merged into HPCC (or High-Performance Computing Cluster) Systems and in 2011, HPCC was open-sourced under the Apache v2.0 License. Currently, HPCC and Quantcast File System are the only publicly available platforms capable of analyzing multiple exabytes of data.

In 2004, Google published a paper on a process called MapReduce that uses a similar architecture. The MapReduce concept provides a parallel processing model, and an associated implementation was released to process huge amounts of data. With MapReduce, queries are split and distributed across parallel nodes and processed in parallel (the Map step). The results are then gathered and delivered (the Reduce step). The framework was very successful, so others wanted to replicate the algorithm. Therefore, an implementation of the MapReduce framework was adopted by an Apache open-source project named Hadoop.

MIKE2.0 is an open approach to information management that acknowledges the need for revisions due to big data implications identified in an article titled "Big Data Solution Offering". The methodology addresses handling big data in terms of useful permutations of data sources, complexity in interrelationships, and difficulty in deleting (or modifying) individual records.

Recent studies show that a multiple-layer architecture is one option to address the issues that big data presents. A distributed parallel architecture distributes data across multiple servers; these parallel execution environments can dramatically improve data processing speeds. This type of architecture inserts data into a parallel DBMS, which implements the use of MapReduce and Hadoop frameworks. This type of framework looks to make the processing power transparent to the end user by using a front-end application server.

Big Data Analytics for Manufacturing Applications can be based on a 5C architecture (connection, conversion, cyber, cognition, and configuration).

The data lake allows an organization to shift its focus from centralized control to a shared model to respond to the changing dynamics of information management. This enables quick segregation of data into the data lake, thereby reducing the overhead time.

Technologies

A 2011 McKinsey Global Institute report characterizes the main components and ecosystem of big data as follows:

Techniques for analyzing data, such as A/B testing, machine learning and natural language processing

Big Data technologies, like business intelligence, cloud computing and databases

Visualization, such as charts, graphs and other displays of the data

Multidimensional big data can also be represented as tensors, which can be more efficiently handled by tensor-based computation, such as multilinear subspace learning. Additional technologies being applied to big data include massively parallel-processing (MPP) databases, search-based applications, data mining, distributed file systems, distributed databases, cloud-based infrastructure (applications, storage and computing resources) and the Internet.

Some but not all MPP relational databases have the ability to store and manage petabytes of data. Implicit is the ability to load, monitor, back up, and optimize the use of the large data tables in the RDBMS.

DARPA's Topological Data Analysis program seeks the fundamental structure of massive data sets and in 2008 the technology went public with the launch of a company called Ayasdi.

The practitioners of big data analytics processes are generally hostile to slower shared storage, preferring direct-attached storage (DAS) in its various forms from solid state drive (Ssd) to high capacity SATA disk buried inside parallel processing nodes. The perception of shared storage architectures—Storage area network (SAN) and Network-attached storage (NAS) —is that they are relatively slow, complex, and expensive. These qualities are not consistent with big data analytics systems that thrive on system performance, commodity infrastructure, and low cost.

Real or near-real time information delivery is one of the defining characteristics of big data analytics. Latency is therefore avoided whenever and wherever possible. Data in memory is good—data on spinning disk at the other end of a FC SAN connection is not. The cost of a SAN at the scale needed for analytics applications is very much higher than other storage techniques.

There are advantages as well as disadvantages to shared storage in big data analytics, but big data analytics practitioners as of 2011 did not favour it.

Data mining

Data mining is an interdisciplinary subfield of computer science. It is the computational process of discovering patterns in large data sets involving methods at the intersection of artificial intelligence, machine learning, statistics, and database systems. The overall goal of the data mining process is to extract information from a data set and transform it into an understandable structure for further use. Aside from the raw analysis step, it involves database and data management aspects, data pre-processing, model and inference considerations, interestingness metrics, complexity considerations, post-processing of discovered structures, visualization, and online updating. Data mining is the analysis step of the "knowledge discovery in databases" process, or KDD.

The term is a misnomer, because the goal is the extraction of patterns and knowledge from large amounts of data, not the extraction (mining) of data itself. It also is a buzzword and is frequently applied to any form of large-scale data or information processing (collection, extraction, warehousing, analysis, and statistics) as well as any application of computer decision support system, including artificial intelligence, machine learning, and business intelligence. The book Data mining: Practical machine learning tools and techniques with Java (which covers mostly machine learning material) was originally to be named just Practical machine learning, and the term data mining was only added for marketing reasons. Often the more general terms (large scale) data analysis and analytics – or, when referring to actual methods, artificial intelligence and machine learning – are more appropriate.

The actual data mining task is the automatic or semi-automatic analysis of large quantities of data to extract previously unknown, interesting patterns such as groups of data records (cluster analysis), unusual records (anomaly detection), and dependencies (association rule mining). This usually involves using database techniques such as spatial indices. These patterns can then be seen as a kind of summary of the input data, and may be used in further analysis or, for example, in machine learning and predictive analytics. For example, the data mining step might identify multiple groups in the data, which can then be used to obtain more accurate prediction results by a decision support system. Neither the data collection, data preparation, nor result

interpretation and reporting is part of the data mining step, but do belong to the overall KDD process as additional steps.

The related terms data dredging, data fishing, and data snooping refer to the use of data mining methods to sample parts of a larger population data set that are (or may be) too small for reliable statistical inferences to be made about the validity of any patterns discovered. These methods can, however, be used in creating new hypotheses to test against the larger data populations.

Etymology

In the 1960s, statisticians used terms like "Data Fishing" or "Data Dredging" to refer to what they considered the bad practice of analyzing data without an a-priori hypothesis. The term "Data Mining" appeared around 1990 in the database community. For a short time in 1980s, a phrase "database mining"™, was used, but since it was trademarked by HNC, a San Diego-based company, to pitch their Database Mining Workstation; researchers consequently turned to "data mining". Other terms used include Data Archaeology, Information Harvesting, Information Discovery, Knowledge Extraction, etc. Gregory Piatetsky-Shapiro coined the term "Knowledge Discovery in Databases" for the first workshop on the same topic (KDD-1989) and this term became more popular in AI and Machine Learning Community. However, the term data mining became more popular in the business and press communities. Currently, Data Mining and Knowledge Discovery are used interchangeably. Since about 2007, "Predictive Analytics" and since 2011, "Data Science" terms were also used to describe this field.

In the Academic community, the major forums for research started in 1995 when the First International Conference on Data Mining and Knowledge Discovery (KDD-95) was started in Montreal under AAAI sponsorship. It was co-chaired by Usama Fayyad and Ramasamy Uthurusamy. A year later, in 1996, Usama Fayyad launched the journal by Kluwer called Data Mining and Knowledge Discovery as its founding Editor-in-Chief. Later he started the SIGKDDD Newsletter SIGKDD Explorations. The KDD International conference became the primary highest quality conference in Data Mining with an acceptance rate of research paper submissions below 18%. The Journal Data Mining and Knowledge Discovery is the primary research journal of the field.

Background

The manual extraction of patterns from data has occurred for centuries. Early methods of identifying patterns in data include Bayes' theorem (1700s) and regression analysis (1800s). The proliferation, ubiquity and increasing power of computer technology has dramatically increased data collection, storage, and manipulation ability. As data sets have grown in size and complexity, direct "hands-on" data analysis has increasingly been augmented with indirect, automated data processing, aided by other discoveries in computer science, such as neural networks, cluster analysis, genetic algorithms (1950s), decision trees and decision rules (1960s), and support vector machines (1990s). Data mining is the process of applying these methods with the intention of uncovering hidden patterns in large data sets. It bridges the gap from applied statistics and artificial intelligence (which usually provide the mathematical background) to database management by exploiting the way data is stored and indexed in databases to execute the actual learning and discovery algorithms more efficiently, allowing such methods to be applied to ever larger data sets.

Process

The Knowledge Discovery in Databases (KDD) process is commonly defined with the stages:

(1) Selection

(2) Pre-processing

(3) Transformation

(4) Data Mining

(5) Interpretation/Evaluation.

It exists, however, in many variations on this theme, such as the Cross Industry Standard Process for Data Mining (CRISP-DM) which defines six phases:

(1) Business Understanding

(2) Data Understanding

(3) Data Preparation

(4) Modeling

(5) Evaluation

(6) Deployment

or a simplified process such as (1) pre-processing, (2) data mining, and (3) results validation.

Polls conducted in 2002, 2004, 2007 and 2014 show that the CRISP-DM methodology is the leading methodology used by data miners. The only other data mining standard named in these polls was SEMMA. However, 3–4 times as many people reported using CRISP-DM. Several teams of researchers have published reviews of data mining process models, and Azevedo and Santos conducted a comparison of CRISP-DM and SEMMA in 2008.

Pre-processing

Before data mining algorithms can be used, a target data set must be assembled. As data mining can only uncover patterns actually present in the data, the target data set must be large enough to contain these patterns while remaining concise enough to be mined within an acceptable time limit. A common source for data is a data mart or data warehouse. Pre-processing is essential to analyze the multivariate data sets before data mining. The target set is then cleaned. Data cleaning removes the observations containing noise and those with missing data.

Data mining

Data mining involves six common classes of tasks:

Anomaly detection (Outlier/change/deviation detection) – The identification of unusual data records, that might be interesting or data errors that require further investigation.

Association rule learning (Dependency modelling) – Searches for relationships between variables. For example, a supermarket might gather data on customer purchasing habits. Using association rule learning, the supermarket can determine which products are frequently bought together and use this information for marketing purposes. This is sometimes referred to as market basket analysis.

Clustering – is the task of discovering groups and structures in the data that are in some way or another "similar", without using known structures in the data.

Classification – is the task of generalizing known structure to apply to new data. For example, an e-mail program might attempt to classify an e-mail as "legitimate" or as "spam".

Regression – attempts to find a function which models the data with the least error.

Summarization – providing a more compact representation of the data set, including visualization and report generation.

Sentiment analysis using product review data

Sentiment analysis

Sentiment analysis or opinion mining is one of the major tasks of NLP (Natural Language Processing). Sentiment analysis has gain much attention in recent years. In this paper, we aim to tackle the problem of sentiment polarity categorization, which is one of the fundamental problems of sentiment analysis. A general process for sentiment polarity categorization is proposed with detailed process descriptions. Data used in this study are online product reviews collected from Amazon.com. Experiments for both sentence-level categorization and review-level categorization are performed with promising outcomes. At last, we also give insight into our future work on sentiment analysis.

Keywords

Sentiment analysis; Sentiment polarity categorization; Natural language processing; Product reviews

Introduction

Sentiment is an attitude, thought, or judgment prompted by feeling. Sentiment analysis, which is also known as opinion mining, studies people's sentiments towards certain entities. Internet is a resourceful place with respect to sentiment information. From a user's perspective, people are able to post their own content through various social media, such as forums, micro-blogs, or online social networking sites. From a researcher's perspective, many social media sites release their application programming interfaces (APIs), prompting data collection and analysis by researchers and developers. For instance, Twitter currently has three different versions of APIs available, namely the REST API, the Search API, and the Streaming API. With the REST API, developers are able to gather status data and user information; the Search API allows developers to query specific Twitter content, whereas the Streaming API is able to collect Twitter content in realtime. Moreover, developers can mix those APIs to create their own applications. Hence, sentiment analysis seems having a strong fundament with the support of massive online data.

However, those types of online data have several flaws that potentially hinder the process of sentiment analysis. The first flaw is that since people can freely post their own content, the quality of their opinions cannot be guaranteed. For example, instead of sharing topic-related opinions, online spammers post spam on forums. Some spam are meaningless at all, while others have irrelevant opinions also known as fake opinions. The second flaw is that ground truth of such online data is not always available. A ground truth is more like a tag of a certain opinion, indicating whether the opinion is positive, negative, or neutral. The Stanford Sentiment 140 Tweet Corpus is one of the datasets that has ground truth and is also public available. The corpus contains 1.6 million machine-tagged Twitter messages. Each message is tagged based on the emoticons (☺as positive, ☹as negative) discovered inside the message.

Data used in this paper is a set of product reviews collected from Amazon, between February and April, 2014. The aforementioned flaws have been somewhat overcome in the following two ways: First, each product review receives inspections before it can be posted a. Second, each review must have a rating on it that can be used as the ground truth. The rating is based on a star-scaled system, where the highest rating has 5 stars and the lowest rating has only 1 star (Figure 1).

This paper tackles a fundamental problem of sentiment analysis, namely sentiment polarity categorization. Figure 2 is a flowchart that depicts our proposed process for categorization as well as the outline of this paper. Our contributions mainly fall into Phase 2 and 3. In Phase 2: 1) An algorithm is proposed and implemented for negation phrases identification; 2) A mathematical approach is proposed for sentiment score computation; 3) A feature vector generation method is presented for sentiment polarity categorization. In Phase 3: 1) Two sentiment polarity categorization experiments are respectively performed based on sentence level and review level; 2) Performance of three classification models are evaluated and compared based on their experimental results.

The rest of this paper is organized as follows: In section ‘Background and literature review’, we provide a brief review towards some related work on sentiment analysis. Software package and classification models used in this study are presented in section ‘Methods’. Our detailed approaches for sentiment analysis are proposed in section ‘Background and literature review’. Experimental results are presented in section ‘Results and discussion’. Discussion and future work is presented in section ‘Review-level categorization’. Section ‘Conclusion’ concludes the paper.

Background and literature review

One fundamental problem in sentiment analysis is categorization of sentiment polarity. Given a piece of written text, the problem is to categorize the text into one specific sentiment polarity, positive or negative (or neutral). Based on the scope of the text, there are three levels of sentiment polarity categorization, namely the document level, the sentence level, and the entity and aspect level. The document level concerns whether a document, as a whole, expresses negative or positive sentiment, while the sentence level deals with each sentence’s sentiment categorization; The entity and aspect level then targets on what exactly people like or dislike from their opinions.

Research design and methdology

Data collection

Data used in this paper is a set of product reviews collected from amazon.com. From February to April 2014, we collected, in total, over 5.1 millions of product reviewsb in which the products belong to 4 major categories: beauty, book, electronic, and home (Figure 3(a)). Those online reviews were posted by over 3.2 millions of reviewers (customers) towards 20,062 products. Each review includes the following information: 1) reviewer ID; 2) product ID; 3) rating; 4) time of the review; 5) helpfulness; 6) review text. Every rating is based on a 5-star scale(Figure 3(b)), resulting all the ratings to be ranged from 1-star to 5-star with no existence of a half-star or a quarter-star.

Sentiment sentences extraction and POS tagging

It is suggested by Pang and Lee that all objective content should be removed for sentiment analysis. Instead of removing objective content, in our study, all subjective content was extracted for future analysis. The subjective content consists of all sentiment sentences. A sentiment sentence is the one that contains, at least, one positive or negative word. All of the sentences were firstly tokenized into separated English words.

Every word of a sentence has its syntactic role that defines how the word is used. The syntactic roles are also known as the parts of speech. There are 8 parts of speech in English: the verb, the noun, the pronoun, the adjective, the adverb, the preposition, the conjunction, and the interjection. In natural language processing, part-of-speech (POS) taggers have been developed to classify words based on their parts of speech. For sentiment analysis, a POS tagger is very useful because of the following two reasons: 1) Words like nouns and pronouns usually do not contain any sentiment. It is able to filter out such words with the help of a POS tagger; 2) A POS tagger can also be used to distinguish words that can be used in different parts of speech. For instance, as a verb, “enhanced" may conduct different amount of sentiment as being of an adjective. The POS tagger used for this research is a max-entropy POS tagger developed for the Penn Treebank Project. The tagger is able to provide 46 different tags indicating that it can identify more detailed syntactic roles than only 8. As an example, Table 1 is a list of all tags for verbs that has been included in the POS tagger.

Each sentence was then tagged using the POS tagger. Given the enormous amount of sentences, a Python program that is able to run in parallel was written in order to improve the speed of tagging. As a result, there are over 25 million adjectives, over 22 million adverbs, and over 56 million verbs tagged out of all the sentiment sentences, because adjectives, adverbs, and verbs are words that mainly convey sentiment.

Negation phrases identification

Words such as adjectives and verbs are able to convey opposite sentiment with the help of negative prefixes. For instance, consider the following sentence that was found in an electronic device's review: "The built in speaker also has its uses but so far nothing revolutionary." The word, "revolutionary" is a positive word according to the list. However, the phrase "nothing revolutionary" gives more or less negative feelings. Therefore, it is crucial to identify such phrases. In this work, there are two types of phrases have been identified, namely negation-of-adjective (NOA) and negation-of-verb (NOV).

Most common negative prefixes such as not, no, or nothing are treated as adverbs by the POS tagger. Hence, we propose Algorithm 1 for the phrases identification. The algorithm was able to identify 21,586 different phrases with total occurrence of over 0.68 million, each of which has a negative prefix. Table 2 lists top 5 NOA and NOV phrases based on occurrence, respectively.

Sentiment score computation for sentiment tokens

A sentiment token is a word or a phrase that conveys sentiment. Given those sentiment words proposed in [27], a word token consists of a positive (negative) word and its part-of-speech tag. In total, we selected 11,478 word tokens with each of them that occurs at least 30 times throughout the dataset. For phrase tokens, 3,023 phrases were selected of the 21,586 identified sentiment phrases, which each of the 3,023 phrases also has an occurrence that is no less than 30. Given a token t, the formula for t's sentiment score (SS) computation is given as:

O c c u r r e n c e i (t) is t's number of occurrence in i-star reviews, where i=1,...,5. According to Figure 3, our dataset is not balanced indicating that different number of reviews were collected for each star level. Since 5-star reviews take a majority amount through the entire dataset, we hereby introduce a ratio, $\gamma_{5,i}$, which is defined as:

In equation 3, the numerator is the number of 5-star reviews and the denominator is the number of i-star reviews, where i=1,...,5. Therefore, if the dataset were balanced, $\gamma_{5,i}$ would be set to 1 for every i. Consequently, every sentiment score should fall into the interval of [1,5]. For positive word tokens, we expect that the median of their sentiment scores should exceed 3, which is the point of being neutral according to Figure 1. For negative word tokens, it is to expect that the median should be less than 3.

As a result, the sentiment score information for positive word tokens is showing in Figure 4(a). The histogram chart describes the distribution of scores while the box-plot chart shows that the median is above 3. Similarly, the box-plot chart in Figure 4(b) shows that the median of sentiment scores for negative word tokens is lower than 3. In fact, both the mean and the median of positive word tokens do exceed 3, and both values are lower than 3, for negative word tokens (Table 3).

The ground truth labels

The process of sentiment polarity categorization is twofold: sentence-level categorization and review-level categorization. Given a sentence, the goal of sentence-level categorization is to classify it as positive or negative in terms of the sentiment that it conveys. Training data for this categorization process require ground truth tags, indicating the positiveness or negativeness of a given sentence. However, ground truth tagging becomes a really challenging problem, due to the amount of data that we have. Since manually tagging each sentence is infeasible, a machine tagging approach is then adopted as a solution. The approach implements a bag-of-word model that simply counts the appearance of positive or negative (word) tokens for every sentence. If there are more positive tokens than negative ones, the sentence will be tagged as positive, and vice versa. This approach is similar to the one used for tagging the Sentiment 140 Tweet Corpus. Training data for review-level categorization already have ground truth tags, which are the star-scaled ratings.

Feature vector formation

Sentiment tokens and sentiment scores are information extracted from the original dataset. They are also known as features, which will be used for sentiment categorization. In order to train the classifiers, each entry

of training data needs to be transformed to a vector that contains those features, namely a feature vector. For the sentence-level (review-level) categorization, a feature vector is formed based on a sentence (review). One challenge is to control each vector's dimensionality. The challenge is actually twofold: Firstly, a vector should not contain an abundant amount (thousands or hundreds) of features or values of a feature, because of the curse of dimensionality; secondly, every vector should have the same number of dimensions, in order to fit the classifiers. This challenge particularly applies to sentiment tokens: On one hand, there are 11,478 word tokens as well as 3,023 phrase tokens; On the other hand, vectors cannot be formed by simply including the tokens appeared in a sentence (or a review), because different sentences (or reviews) tend to have different amount of tokens, leading to the consequence that the generated vectors are in different dimensions.

Since we only concern each sentiment token's appearance inside a sentence or a review,to overcome the challenge, two binary strings are used to represent each token's appearance. One string with 11,478 bits is used for word tokens, while the other one with a bit-length of 3,023 is applied for phrase tokens. For instance, if the ith word (phrase) token appears, the word (phrase) string's ith bit will be flipped from "0" to "1". Finally, instead of directly saving the flipped strings into a feature vector, a hash value of each string is computed using Python's built-in hash function and is saved. Hence, a sentence-level feature vector totally has four elements: two hash values computed based on the flipped binary strings, an averaged sentiment score, and a ground truth label. Comparatively, one more element is exclusively included in review-level vectors. Given a review, if there are m positive sentences and n negative sentences, the value of the element is computed as: $-1\times m+1\times n$.

Results and discussion

Evaluation methods

Performance of each classification model is estimated base on its averaged F1-score

where P i is the precision of the ith class, R i is the recall of the ith class, and n is the number of classes. P i and R i are evaluated using 10-fold cross validation. A 10-fold cross validation is applied as follows: A dataset is partitioned into 10 equal size subsets, each of which consists of 10 positive class vectors and 10 negative class vectors. Of the 10 subsets, a single subset is retained as the validation data for testing the classification model, and the remaining 9 subsets are used as training data. The cross-validation process is then repeated 10 times, with each of the 10 subsets used exactly once as the validation data. The 10 results from the folds are then averaged to produce a single estimation. Since training data are labeled under two classes (positive and negative) for the sentence-level categorization, ROC (Receiver Operating Characteristic) curves are also plotted for a better performance comparison.

Sentence-level categorization

Sentence-level categorization

Result on manually-labeled sentences

200 feature vectors are formed based on the 200 manually-labeled sentences. As a result, the classification models show the same level of performance based on their F1-scores, where the three scores all take a same value of 0.85. With the help of the ROC curves (Figure 5), it is clear to see that all three models performed quite well for testing data that have high posterior probability. (A posterior probability of a testing data point, A, is estimated by the classification model as the probability that A will be classified as positive, denoted as P(+|A).) As the probability getting lower, the Naïve Bayesain classifier outperforms the SVM classifier, with a larger area under curve. In general, the Random Forest model performs the best.

Result on machine-labeled sentences

2-million feature vectors (1 million with positive labels and 1 million with negative labels) are generated from 2-million machine-labeled sentences, known as the complete set. Four subsets are obtained from the complete set, with subset A contains 200 vectors, subset B contains 2,000 vectors, subset C contains 20,000 vectors, and subset D contains 200,000 vectors, respectively. The amount of vectors with positive labels equals

the amount of vectors with negative labels for every subset. Performance of the classification models is then evaluated based on five different vector sets (four subsets and one complete set, Figure 6)

While the models are getting more training data, their F1 scores are all increasing. The SVM model takes the most significant enhancement from 0.61 to 0.94 as its training data increased from 180 to 1.8 million. The model outperforms the Naïve Bayesain model and becomes the 2nd best classifier, on subset C and the full set. The Random Forest model again performs the best for datasets on all scopes. Figure 7 shows the ROC curves plotted based on the result of the full set.

Review-level categorization

3-million feature vectors are formed for the categorization. Vectors generated from reviews that have at least 4-star ratings are labeled as positive, while vectors labeled as negative are generated from 1-star and 2-star reviews. 3-star reviews are used to prepare neutral class vectors. As a result, this complete set of vectors are uniformly labeled into three classes, positive, neutral, and negative. In addition, three subsets are obtained from the complete set, with subset A contains 300 vectors, subset B contains 3,000 vectors, subset C contains 30,000 vectors, and subset D contains 300,000 vectors, respectively.

Figure 8 shows the F1 scores obtained on different sizes of vector sets. It can be clearly observed that both the SVM model and the Naïve Bayesain model are identical in terms of their performances. Both models are generally superior than the Random Forest model on all vector sets. However, neither of the models can reach the same level of performance when they are used for sentence-level categorization, due to their relative low performances on neutral class.

The experimental result is promising, both in terms of the sentence-level categorization and the review-level categorization. It was observed that the averaged sentiment score is a strong feature by itself, since it is able to achieve an F1 score over 0.8 for the sentence-level categorization with the complete set. For the review-level categorization with the complete set, the feature is capable of producing an F1 score that is over 0.73. However, there are still couple of limitations to this study. The first one is that the review-level categorization becomes difficult if we want to classify reviews to their specific star-scaled ratings. In other words, F1 scores obtained from such experiments are fairly low, with values lower than 0.5. The second limitation is that since our sentiment analysis scheme proposed in this study relies on the occurrence of sentiment tokens, the scheme may not work well for those reviews that purely contain implicit sentiments. An implicit sentiment is usually conveyed through some neutral words, making judgement of its sentiment polarity difficult. For example, sentence like “Item as described.", which frequently appears in positive reviews, consists of only neutral words.

With those limitations in mind, our future work is to focus on solving those issues. Specifically, more features will be extracted and grouped into feature vectors to improve review-level categorizations. For the issue of implicit sentiment analysis, our next step is to be able to detect the existence of such sentiment within the scope of a particular product. More future work includes testing our categorization scheme using other datasets.

Conclusion

Sentiment analysis or opinion mining is a field of study that analyzes people’s sentiments, attitudes, or emotions towards certain entities. This paper tackles a fundamental problem of sentiment analysis, sentiment polarity categorization. Online product reviews from Amazon.com are selected as data used for this study. A sentiment polarity categorization process (Figure 2) has been proposed along with detailed descriptions of each step. Experiments for both sentence-level categorization and review-level categorization have been performed.

Methods

Software used for this study is scikit-learn, an open source machine learning software package in Python. The classification models selected for categorization are: Naïve Bayesian, Random Forest, and Support Vector Machine

Naïve Bayesian classifier

The Naïve Bayesian classifier works as follows: Suppose that there exist a set of training data, D, in which each tuple is represented by an n-dimensional feature vector, X=x 1,x 2,..,x n , indicating n measurements made on the tuple from n attributes or features. Assume that there are m classes, C 1,C 2,...,C m . Given a tuple X, the classifier will predict that X belongs to C i if and only if: P(C i |X)>P(C j |X), where i,j∈[1,m]a n d i≠j. P(C i |X) is computed as:

Random forest

The random forest classifier was chosen due to its superior performance over a single decision tree with respect to accuracy. It is essentially an ensemble method based on bagging. The classifier works as follows: Given D, the classifier firstly creates k bootstrap samples of D, with each of the samples denoting as D i . A D i has the same number of tuples as D that are sampled with replacement from D. By sampling with replacement, it means that some of the original tuples of D may not be included in D i , whereas others may occur more than once. The classifier then constructs a decision tree based on each D i . As a result, a “forest" that consists of k decision trees is formed. To classify an unknown tuple, X, each tree returns its class prediction counting as one vote. The final decision of X’s class is assigned to the one that has the most votes.

The decision tree algorithm implemented in scikit-learn is CART (Classification and Regression Trees). CART uses Gini index for its tree induction. For D, the Gini index is computed as:

Support vector machine

Support vector machine (SVM) is a method for the classification of both linear and nonlinear data. If the data is linearly separable, the SVM searches for the linear optimal separating hyperplane (the linear kernel), which is a decision boundary that separates data of one class from another. Mathematically, a separating hyperplane can be written as: W·X+b=0, where W is a weight vector and W=w 1,w2,...,w n. X is a training tuple. b is a scalar. In order to optimize the hyperplane, the problem essentially transforms to the minimization of ‖W‖, which is eventually computed as: ∑i=1nαiyixi∑i=1nαiyixi, where α i are numeric parameters, and y i are labels based on support vectors, X i . That is: if y i =1 then ∑i=1nwixi≥1∑i=1nwixi≥1; if y i =−1 then ∑i=1nwixi≥−1∑i=1nwixi≥−1.

the SVM uses nonlinear mapping to transform the data into a higher dimension. It then solve the problem by finding a linear hyperplane. Functions to perform such transformations are called kernel functions. The kernel function selected for our experiment is the Gaussian Radial Basis Function (RBF):

K(Xi,Xj)=e−γ‖Xi−Xj‖2/2

K(Xi,Xj)=e−γ‖Xi−Xj‖2/2

where X i are support vectors, X j are testing tuples, and γ is a free parameter that uses the default value from scikit-learn in our experiment. Figure 9 shows a classification example of SVM based on the linear kernel and the RBF kernel.

If the data is linearly inseparable,