МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ОДЕСЬКИЙ ДЕРЖАВНИЙ ЕКОЛОГІЧНИЙ УНІВЕРСИТЕТ

Факультет Магістерської та
аспірантської підготовки
Кафедра інформаційних технологій

МАГІСТЕРСЬКА КВАЛІФІКАЦІЙНА РОБОТА

на тему: Дослідження методу визначення тематики тексту
за його статистичними характеристиками

Виконав студент 2 курсу групи МК- 61
спеціальності 122 Комп'ютерні науки
та інформаційні технології
Яценко Анна Вячеславівна

Керівник к.т.н., доцент,
Онищенко Сергій Михайлович

Консультант

Рецензент к.геогр.н.
Лужбін Анатолій Михайлович

Одеса 2018

АНОТАЦІЯ

Тема магістерської роботи «Дослідження методу визначення тематики тексту за його статистичними характеристиками».

Актуальність магістерської роботи полягає в необхідності розробки методу класифікації текстів по тематиках та реалізація програмного додатку, який здійснює цілеспрямований пошук документів, що відносяться з тим або іншим ступенем до певної теми.

Об'єктом дослідження є природно-мовні тексти як форма представлення знань предметної області.

Предметом дослідження є методи і засоби підвищення продуктивності систем автоматичної текстової класифікації.

Метою цієї роботи є дослідження алгоритмів класифікації і розробка методу класифікації текстів по тематиках

Для реалізації поставленої мети були вирішені наступні питання. Проведено дослідження та аналіз існуючих систем класифікації текстів, методів відбору ознак та алгоритмів класифікації. Був проведений поетапний розбір поставленої задачі та методу її вирішення. Розглянуті деякі сучасні та найбільш популярні алгоритми для визначення тематики текстів, а також розроблена програма, за допомогою якої вирішується дана задача, був проведений аналіз її роботи і порівняння з аналогом. Робота програми складається в навчанні класифікатора на основі набору документів із завчасно відомою тематикою і визначення тематики довільних текстових документів, використовуючи отриману на етапі навчання базу знань. Застосовуючи наївний байесовський алгоритм, програма рахує імовірність приналежності тестованого документу до кожної з категорій у процентному співвідношенні.

Ключові слова: класифікація текстів, відбор ознак, наївний байесовський алгоритм, навчальна вибірка.

Магістерська робота містить 74 сторінок, 4 таблиці, 26 рисунків, 18 посилань та 3 листів додатоку.

ABSTRACT

The topic of master work "The method of determining the subject matter of the text by its statistical characteristics ".

The relevance of this master work is to develop of method of classification of texts on subjects and realization of programmatic application, that carries out the purposeful search of documents belonging with one or another degree to the certain theme.

Object of research – is naturally-language texts as form ofrepresentation of knowledge to the subject domain.

Subject of research – are methods and facilities of increaseof the productivity of the systems of automatic textclassification.

Purpose – the study of classification algorithms and the development of a method for classifying texts by subject

In this paper describes the basics of pattern recognition theory subjects texts, methods of feature selection, classification algorithms. Was carried out step-by-step analysis of the problem and the method of its solution. Discusses some modern and the most popular algorithms to determine topics of texts, as well as developed an application that allows you to solve this problem, an analysis of his work and a comparison with the analogous. The application performs training of classifier using documents with known categories and then determine category of arbitrary test documents. Applying naive Bayes algorithm the application calculates the probability that a test document belongs to each category as a percentage.

Key words: classification of texts, selection of features, naive Bayesian algorithm, training sample.

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ, УМОВНИХ ПОЗНАЧЕНЬ ТА ТЕРМІНІВ

ЗМІ – засоби масової інформації.

ПЗ – програмне забезпечення.

Предметна область інформаційної системи – це матеріальна система або система, що характеризує елементи матеріального світу, інформація про яку зберігається і обробляється. Предметна область розглядається як деяка сукупність реальних об'єктів і зв'язків між ними. Кожний об'єкт має певний набір властивостей (атрибутів).

Стемінг – це процес знаходження основи слова для заданого вихідного слова. Основа слова необов'язково збігається з морфологічним коренем слова. Даний процес застосовується в пошукових системах для узагальнення пошукового запиту користувача.

Стемер – це конкретні реалізації стемінга, тобто алгоритм стемінга.

Текстомайнінг (text mining) часто називають також текстовим дейтамайнінгом (text data mining), що розкриває взаємозв'язок двох цих технологій. Якщо дейтамайнинг дозволяє витягати нові знання (сховані закономірності, факти, невідомі взаємозв'язки і т.ін.) з більших об'ємів структурованої інформації (збереженої в базах даних), то текстомайнінг – знаходити нові знання в неструктурованих текстових масивах.

ВСТУП

Стрімкий розвиток мережі Інтернет привів до різкого зростання кількості електронних документів. По оцінкам експертів, у цей час близько 70% накопиченої і використовуваної суспільством цифрової інформації перебуває в неструктурованій (текстовій) формі і лише 30% становлять інші види даних. Експонентне із часом збільшення кількості неструктурованих даних привело власне кажучи до колапсу традиційної системи одержання і розподілу текстової інформації, перетворило рутинну операцію пошуку і аналізу необхідних відомостей у трудомісткий і малоефективний процес, що викликає інформаційне перевантаження користувачів. У цій ситуації особливу актуальність здобувають роботи зі створення систем обробки текстової інформації, тому що навіть висококваліфіковані експерти зазнають труднощів по організації пошуку документів і розподілу отриманих текстових даних за тематиками. Як показує практика, результати визначення предметної області документа «вручну», тобто шляхом експертного віднесення до наявної рубрики, звичайно не перевищують 80% .

Класифікація текстів – сортування текстових документів по заздалегідь визначеним категоріям – один зі способів структурування даних. Методи класифікації текстових документів лежать на стику двох областей – інформаційного пошуку і машинного навчання. Загальні частини двох цих підходів – способи подання документів і способи оцінки якості класифікації текстів, а розходження складаються тільки в способах власне пошуку.

Незважаючи на те, що проблеми класифікації текстових документів перебувають у центрі уваги цілого ряду наукових колективів, по багатьом питанням дотепер не знайдено задовільних відповідей. Точність різних методів істотно залежить від виконання апріорних припущень і допущень, а також структури текстових даних (кількість класів, розміри і однорідність класів, вид «прикордонної» області між класами).

При обробці текстової інформації виникають наступні труднощі. По-перше, кількість інформативних, тобто корисних для класифікації ознак або термінів, звичайно істотно перевершує кількість документів у вибірці, значно утрудняючи навчання методів і визначення найкращих оцінок параметрів. По-друге, об'єм обчислювальних операцій при обробці і аналізі текстових документів надзвичайно великий, що робить процес класифікації дорогим і вкрай трудомістким. По-третє, одержувана матриця «документ – термін» виявляється дуже сильно розрідженою, тому що велика кількість термінів зу-

стрічається тільки в одному або декількох документах. По-четверте, на відміну від структурованої інформації, що звичайно містить фактичні відомості у вигляді чисел, неструктурована інформація не має єдиного текстового формату і загальноприйнятих правил, що робить обробку і аналіз документів практично неможливим без розробки комплексної моделі процесу обробки текстової інформації.

Основними областями застосування класифікації текстів у сучасних пошукових Інтернет-системах є:

– фільтрація спама;

– фільтрація неприйнятних матеріалів;

– визначення мови документа;

– класифікація користувальницьких запитів;

– ранжирування новин;

– складання інтернет-каталогів;

– контекстна реклама;

– зняття неоднозначності слів при перекладі фраз в автоматичних перекладачах.

Для розв’язання перерахованих задач потрібне застосування методів класифікації на основі машинного навчання, оскільки склад і вміст аналізованих документів постійно змінюється, і одним зі шляхів адаптації до цієї динаміки є використання таких методів. Ціль методів машинного навчання для задачі класифікації текстових документів полягає в побудові моделі класифікації на основі навчального набору і застосуванні побудованої моделі для передбачання класу або набору класів, релевантних для нового документа. Навчальний набір для розглянутої задачі класифікації складається з документів, кожному з яких зіставлена множина класів, до яких даний документ відноситься. Такий підхід забезпечує якість класифікації, порівняну з якістю класифікації, зробленою людиною.

Традиційно в машинному навчанні існують два основні підходи.

– Навчання без вчителя (unsupervised learning). Є множина об'єктів $X=\{x_1, x_2,..., x_n\}$. Потрібно виявити внутрішні взаємозв'язки, залежності, закономірності, що існують між об'єктами множини X.

– Навчання з учителем (supervised learning). Дано множину об'єктів $X=\{x_1, x_2,..., x_n\}$ і множину можливих відповідей $Y=\{1,...,c\}$. Відомо навчальну множину пар «об'єкт-відповідь» $\{(x_1,y_1),...(x_n,y_n)\}$. На основі цих даних потрібно відновити залежність, тобто побудувати класифікатор, здатний для будь-якого об'єкта визначити до якого класу він належить.

Алгоритми класифікації працюють із деякою математичною моделлю визначення екземпляра (у цьому випадку – текстового документа). Найпоширенішою моделлю є визначення у вигляді набору ознак, що і буде розглянуто надалі. Визначення ознак і зіставлення їм ваг є істотно неформальним кроком і багато в чому впливає на результат класифікації.

Головною метою даної роботи є розробка методу класифікації текстів по тематиках. Для досягнення даної мети необхідно вибрати алгоритм класифікації документів, спроектувати і реалізувати програмний продукт, що робить класифікацію текстових документів на основі попередньо зробленого навчання на наборі документів із заздалегідь відомою тематикою.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ І ПОСТАНОВКА ЗАДАЧІ

1.1 Огляд аналогів

Поняття класифікації є дуже загальним і має багато різних додатків (в області інформаційного пошуку і за його межами). Наприклад, в області комп'ютерного зору класифікатор може користуватися для розподілу образів по класах, таких як ландшафт, портрет і ін.

Аналогом розроблювального в даному дипломному проекті додатка є системи для визначення тематики сайтів.

Linkfeedator – php-клієнт до бірж [1], також має функцію визначення тематики сторінок, з яких клієнти купують посилання. Як зазначено в описі цього клієнта, ця функція особливо корисна для нарощування Тематичного Індексу Цитування.

Сервіс cfilter.ru – автоматично визначає тематику web-сторінок і сайтів, а також фільтрує url-адреси [2].

Ashmanov.com – з так називаною технологією «Семантичне дзеркало» визначає тематику заданого тексту і сайтів [3].

Сервіс Extheme.ru – також дозволяє визначити тематику введеного тексту і сайтів, реалізований на основі нейронних мереж [4]. Система категорій Extheme.ru відповідає системі категорій Яндекс-каталог, є ієрархічною і містить 16 основних категорій («Розвага», «Відпочинок», «Спорт», «Навчання» і т.д.). Надає можливість роботи в пакетному режимі через api-інтерфейс xml-rpc, є плагін під FireFox. Використання системи Extheme.ru є платним, але надається необмежений доступ в демо-режимі, використовуючи веб.

Всі ці системи вимагають реєстрацію для їхнього використання, тільки Extheme.ru і Ashmanov.com дають можливість тестування можливостей у демо-версії (Achmanov.com обмежує число спроб до 20 у добу).

Для аналізу документів природною мовою розроблена велика кількість бібліотек, що містять набори базових алгоритмів для аналізу текстів, в основному англійською мовою. Найбільш відомими з них є OpenNLPii, NLTK[1], LingPipeiii. Відомі також інфраструктурні проекти GATE[2], Apache UIMA[3], що надають набір інструментів для текстової аналітики і розширювану архітектуру для додавання нових інструментів. Найбільш відомим пакетом інструментів є AOT [4]. Комерційні рішення надають компанії ABBYY, RCO, IBM і ін. Розв'язувані цими інструментами задачі, у більшості випадків відносяться до рівня морфології і синтаксису. Це пов'язано зі складністю по-

будови баз знань, на основі яких можна перейти від слів і термінів до їхніх значень. Для англійської мови існує тезаурус WordNet, що дозволяє визначити можливі значення слів і створити алгоритми дозволу лексичної багатозначності. Також розробляються аналоги: РуТез [5], YARN [6] і ін. Основними недоліками цих ресурсів є складність їхньої розробки, що вимагає залучення експертів, і їхня орієнтованість на покриття тільки загальної лексики, якої недостатньо для розв’язання предметно-орієнтованих задач.

У системі Texterra основна база знань автоматично витягається з Вікіпедії і більш ніж на порядок перевищує розмір WordNet. Крім того, розробляються інструменти для автоматичної побудови предметно-орієнтованих баз знань на основі аналізу документації, що значно розширює область застосування технології. Ще одним популярним способом використання інструментів текстового аналізу є їхнє подання у вигляді хмарних сервісів, що дозволяє створювати інтелектуальні Веб-додатки. Ця ідея лежить в основі декількох проектів, найбільш відомими з яких є AlchemyAPI, OpenCalais, Semantria і OpenAmplify. Користувальницькі додатки можуть передавати текст таким сервісам і одержувати як відповідь розв’язок складних задач обробки тексту. Texterra є першим проектом, що надає аналогічну функціональність. Основними відмінними рисами системи Texterra є: розширювана архітектура, автоматично поповнювана база знань, підтримка роботи з декількома базами знань, інструменти для аналізу лексичної семантики, що використовують базу знань, підтримка декількох мов, висока швидкість обробки даних.

У рамках проекту Texterra була створена технологія, що дозволяє розв’язувати широкий клас задач, пов'язаних з обробкою текстових даних. Залежно від розв'язуваної задачі Texterra може бути використана як бібліотека алгоритмів, розширюваний фреймворк або масштабований хмарний сервіс. На відміну від більшості існуючих систем обробки текстів, Texterra надає можливість переходу від роботи з окремими словами і термінами до роботи з їх значенням. Це дозволяє збільшити точність розв’язку багатьох прикладних задач. При цьому особлива увага при розробці технології приділялося продуктивності системи – на даний момент Texterra є одним з найшвидших розв’язків у даній області. Важливою перевагою технології Texterra є низькі витрати на впровадження і підтримку системи за рахунок автоматизації процесу побудови і відновлення бази знань. Як основна база знань використовується інформація, що автоматично витягається з Вікіпедії. Далі ця база знань розширюється інформацією з інших Веб-ресурсов і за рахунок аналізу текстових документів. Такий підхід дозволяє застосовувати розроблені методи не

тільки до заздалегідь визначеної предметної області, але і швидко адаптувати технологію до нових задач і мов.

STATISTICA Text Miner – це додаткове розширення програми STATISTICA Data Miner, призначене для перекладу неструктурованих текстових даних в інформацію, придатну для прийняття рішень. STATISTICA Text Miner дозволяє витягати з тексту необхідні дані, структурувати їх і представляти інформацію в графічному вигляді. У якості вхідних даних можна використовувати не тільки текстові документи або веб-сторінки, алі й файли інших типів. Програма забезпечує доступ до текстових документів у різних форматах, включаючи TXT, PDF, PS, HTML, XML, RTF і ін.

Документи можуть бути оброблені, перш ніж вони будуть проіндексовані (фактично ці процеси відбуваються одночасно). Програма написана таким чином, що підтримка додаткових мов здійснюється з мінімумом витрат. Засоби аналізу дозволяють одержати кількісний звіт по досліджуваному тексту. Шляхом статистичного аналізу можна оцінювати степінь подібності документів. На базі зіставлення документів за частотою появи в них різних слів можна встановити приналежність документа до тієї або іншої смислової категорії. Кластерний аналіз дозволяє ідентифікувати групи подібних за змістом документів. Передбачувальні методи добування даних дозволяють установлювати зв'язки між отриманими чисельними характеристиками документів з іншими індикаторами (наприклад, оцінити намір увести в оману, медичний діагноз і т.д.).

STATISTICA Text Miner має відкриту архітектуру. Програмне забезпечення для текстомайнінга може бути інтегроване з кожним ПО з лінійки продуктів STATISTICA: STATISTICA Data Miner workspace, Webstatistica або зі звичайними додатками STATISTICA.

«Медіалогія» – це система для проведення глибоких досліджень за відкритими джерелами інформації на базі технології аналізу масивів неструктурованої інформації [3].

Система «Медіалогія» не передбачає передачі програми замовникам, роблячи обслуговування клієнтів в онлайновому режимі. «Медіалогія» – це web-додаток, що представляє собою потужний розв'язок зі складною архітектурою, що і забезпечує безперервну обробку вхідної інформації, структуроване зберігання даних, розрахунки аналітичних параметрів, проведення аналізу за запитами користувача і зберігання настроювань і звітів.

Користувач, що має доступ до системи, створює запит і одержує готовий звіт, доступний для перегляду через систему або експорту на комп'ютер

користувача. Персональні настроювання і користувацький профіль також зберігаються на сервері. Така схема дозволяє зробити систему максимально продуктивною і не прив'язаною до конкретного комп'ютера.

Система «Медіалогія» щодня імпортує десятки тисяч повідомлень, що надходять із різних джерел (газет, журналів, телебачення, радіо, інформаційних агентств, інтернет-ресурсів). Ці повідомлення структуруються, оцінюються і проходять семантичну обробку. Отримані в результаті обробки розрахункові індекси і семантичні зв'язки є основою для проведення аналізу інформації.

«Медіалогія» призначена в тому числі для розв'язку наступних задач:

- конкурентний аналіз;
- інформаційна розвідка;
- керування репутацією;
- вивчення галузевого ринку;
- оперативний моніторинг ЗМІ;
- точний пошук інформації з відкритих джерел.

Система дозволяє робити пошук повідомлень по заданих параметрах і контексту із застосуванням технологій штучного і людського інтелекту. Система спеціалізується на аналізі інформаційного поля на основі інтелектуальної обробки даних у режимі реального часу. При цьому можливо виявлення зв'язків і відношень між персонами і компаніями, відстеження особливостей відображення ситуації окремими джерелами або авторами. Система дозволяє відслідковувати десятки типів зв'язків (партнер, конкурент, акціонер, друг) і взаємин (контакти, фінансові відношення, конфлікти) між об'єктами.

Результати запитів до системи представлені у формі інтуїтивно зрозумілої ділової графіки. Для вивчення регіонального розподілу інформації використовується геоіформаційна карта України. Спеціальні «семантичні карти» служать для візуалізації зв'язків об'єкта. Усі види, представлені в системі «Медіалогія», інтерактивні. Колірна розмітка текстів повідомлень дозволяє вільно орієнтуватися в тексті.

У системі також зберігаються матеріали в оригінальних форматах.

Джерела інформації проходять ретельний відбір – у базу «Медіалогії» попадають тільки найбільш значні у своїх областях ЗМІ. Програма дозволяє розрахувати так званий індекс інформаційного сприяння – розрахунковий показник, який дає можливість оцінити якісну складову інформаційної ситуації, яка склалася навколо персони, компанії або бренда.

1.2 Методи класифікації текстів

Класифікація текстових документів полягає у віднесенні документа до одного із заздалегідь відомих класів. Часто класифікацію стосовно текстових документів називають категоризацією або рубрикацією. Більшість методів класифікації текстів так чи інакше засновані на припущенні, що документи, що відносяться до однієї категорії, містять однакові ознаки (слова або словосполучення), і наявність або відсутність таких ознак в документі говорить про його приналежність або неприналежність до тієї або іншої теми. Таким чином, для кожної категорії повинна бути множина ознак:

$$F(C)=\cup F(c_r), \tag{1.1}$$

де $F(c_r)=\{ f_1,...,f_k,...,f_z\}$.

Таку множину ознак часто називають словником, оскільки вона складається з лексем, які включають слова і/або словосполучення, що характеризують категорію.

Подібно до категорій кожен документ також має ознаки, за якими його можна віднести з деякою мірою вірогідності до однієї або декількох категорій:

$$F(d_i)=\{ f_1^{\,i},...,f_k^{\,i},...,f_z^{\,i}\}. \tag{1.2}$$

Множина ознак всіх документів повинна співпадати з множиною ознак категорій, тобто:

$$F(C)=F(D)\cup F(d_i). \tag{1.3}$$

Необхідно відмітити, що дані набори ознак є відмітною межею класифікації текстових документів від класифікації об’єктів в Data Mining, які характеризуються набором атрибутів. Рішення про віднесення документа до категорії ухвалюється на підставі перетину:

$$F(d_i))=\cup F(c_r). \tag{1.4}$$

Задача методів класифікації полягає в тому, щоб найкращим чином обрати такі ознаки і сформулювати правила, на основі яких ухвалюватиметься

рішення про віднесення документа до рубрики.

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

а) машинне навчання, де передбачається наявність повчальної вибірки документів, але якому будується множина $F(C)$;

б) експертний метод, який припускає, 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. ін.) з документів повчальної вибірки. У неї повинні входити документи, які відносяться до кожної рубрики, щоб створити набір ознак (статистичну сигнатуру) для кожної рубрики, які згодом використовуватимуться для класифікації нових документів. Гідністю даного підходу є відсутність необхідності в словниках, які складно побудувати для великих наочних областей. Проте, щоб уникнути невірної класифікації, потрібно забезпечити вірне представлення документів для кожної рубрики.

У другому випадку формування словника (множини F(C)) може бути виконане на основі набору термінів наочної області і відносин між ними (основні терміни, синоніми і споріднені терміни). Класифікація може потім визначити рубрику документа відповідно до частоти, з якою з’являються виділені в тексті терміни (ключові поняття).

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

Перед тим, як приступати до будь-якої задачі класифікації, одна з найбільш фундаментальних процедур, яку необхідно виконати, це вибрати подання документа і відібрати ознаки.

Нехай необхідно класифікувати деякий набір текстів, тобто віднести кожен текст до одного з заздалегідь заданих класів – категорій текстів. Наприклад, потрібно класифікувати email-повідомлення як непотрібні (що містять спам, spam) і корисні (ham). Будемо вважати, що маємо у своєму розпорядженні деякий набір текстів email-повідомлень, що вже був заздалегідь класифікований (наприклад, користувач класифікував його вручну). Як правило, задача класифікації текстів за допомогою машинного навчання розв’язується у два етапи: етап навчання і етап застосування. На етапі на-

вчання будується алгоритм навчання. Вхідні дані для алгоритму – це навчальна вибірка, що складається із множини об'єктів, кожний з яких заданий вектором ознак і міткою одного із класів. Ознака об'єкта – це результат виміру деякої характеристики об'єкта. За навчальною вибіркою алгоритм навчання повинен побудувати класифікатор, що на етапі застосування буде передбачати мітки класів для об'єктів, що не входять у навчальну вибірку. У даному прикладі класифікатор повинен відповісти для заздалегідь невідомого йому email-повідомлення, чи є воно спамом. Точність класифікатора – це відношення правильно класифікованих об'єктів до загального числа класифікованих об'єктів. Основні труднощі класифікації текстів за допомогою машинного навчання полягають в дуже великій розмірності простору ознак. Власний простір ознак може складатися з унікальних термів (слів або фраз), які зустрічаються в корпусі документів, і може включати десятки і сотні тисяч термів навіть для корпуса середнього об'єму. Це непомірно багато для більшості алгоритмів навчання. Наприклад, більшість нейронних мереж нездатні сприйняти таку кількість вхідних вершин, а наївна байєсівська модель буде займати занадто багато часу на обчислення. Ця проблема в машинному навчанні носить назву «прокляття» розмірності. Таким чином, було б бажано зменшити розмірність простору ознак, не жертвуючи точністю класифікатора. Задача вибору оптимального набору ознак полягає в тім, щоб вибрати таку підмножину ознак з вихідного набору ознак, що точність класифікатора, що навчається на цій підмножині ознак, буде максимальною (по всіх підмножинах вихідної множини ознак).

Нехай $I(T)$ – алгоритм навчання, $T(X)$ – навчальна вибірка, X – множина ознак навчальної вибірки $T(X)$, $X' \subseteq X$ – підмножина множини ознак, $T(X')$ – навчальна вибірка, побудована з використанням підмножини ознак X', D – класифікатор: $D = I(T(\mathrm{X}))$, $Q(D)$ – точність класифікатора. Тоді оптимальний набір ознак $X\mathrm{opt}$ визначається так:

$$\mathrm{Xopt} = \arg\max Q(D),\ D = I(T(X')),\ X' \subseteq X.$$

Оскільки розподіл об'єктів і класів невідомий, то точність класифікатора потрібно оцінювати з використанням тестової вибірки. Метою виділення оптимального набору ознак є не тільки зменшення розмірності простору ознак. У наборі ознак можуть зустрітися шумові ознаки (noisy) – такі, які будучи доданими в модель класифікатора, зменшують його точність на нових наборах даних. Їхнє виключення дає поліпшення точності класифікатора. Наприклад, рідкий термін «квазісингулярність» не несе ніякої інформації про клас «спам», однак може так виявитися, що в навчальній вибірці цей термін

виявиться тільки в повідомленнях, класифікованих як спам. У цьому випадку (при використанні, наприклад, наївної байєсовської моделі навчання) ми можемо одержати класифікатор, що помилково класифікує всі email-повідомлення, що містять слово «квазісінгулярність», як спам-повідомлення. Таке некоректне узагальнення випадкових закономірностей з навчальних даних називається перенавчанням. Вибір оптимальних ознак може допомогти в вирішенні проблеми перенавчання класифікатора. Крім цього, виключення неінформативних ознак допомагає краще зрозуміти природу текстових даних, зменшити розміри сховищ для текстових документів. Задача вибору оптимального набору ознак відрізняється від задачі конструювання ознак. Конструювання ознак (feature construction) розв'язує подібну задачу зменшення розмірності вихідних даних, однак різниця полягає в тім, що методи конструювання ознак формують нові ознаки на основі вже наявних. До методів конструювання ознак відносяться такі методи, як:

– метод головних компонентів (PCA, principal component analysis) і його різновиди;

– кластерізація ознак (як нові ознаки вибираються центри кластерів);

– автокодировщик (спеціальний вид нейронної мережі);

– регуляризований випадковий ліс (RRF, regularized random forest) ;

– спектральні і хвильові перетворення ознак.

1.3 Подання документів і відбір ознак

Хоч відбір ознак і застосовується досить часто в інших задачах класифікації, він особливо важливий у задачі класифікації текстів через високу розмірність (велику кількість ознак) і наявність нерелевантних (шумових) ознак. У більшості випадків, подання тексту відбувається одним із двох способів. Перший – документ як набір слів, в якому документу зіставляються слова і частота їх зустрічаємості в ньому. Таке подання не залежить від порядку слів, у якому вони зустрічаються в тексті. Другий метод полягає в поданні тексту, власне, як набору рядків, у якому документ є послідовністю слів. Більшість алгоритмів класифікації текстів використовують перше подання через його простоту і зручність для задач класифікації.

Найбільш популярними способами відбору ознак є видалення стоп-слів і стемінг. При видаленні стоп-слів, ми визначаємо загальні слова документів, які не є специфічними або поділяючими для різних класів. При застосуванні стемінга різні форми одного слова поєднуються в одне слово (терм). Напри-

клад, так поєднуються слова різного роду/форми/часу/відмінку і ін. Множина методів відбору ознак була розглянута і експериментально перевірена на ефективність в [5].

Існує кілька підходів до вибору оптимальних інформаційних ознак. По розповсюдженій класифікації [1], існує три основних категорії методів вибору оптимальних інформаційних ознак: методи-фільтри (Filters), методи-обгортки (Wrappers) і убудовані методи (Embedded).

1.4 Методи-фільтри (Filters)

Методи-фільтри не взаємодіють із алгоритмом навчання і вибирають оптимальну підмножину ознак, використовуючи тільки інформацію, отриману з навчальної вибірки. Методи-фільтри виконуються на етапі передобробки, до виконання алгоритму навчання. Методи-фільтри можуть як незалежно оцінювати інформативність ознак для навчання, так і оцінювати підмножину ознак у сукупності. У першому випадку знадобиться визначити значення граничної константи (потрібної для того, щоб відкинути ті ознаки, інформативність яких для алгоритму навчання нижче значення порога). У другому випадку знадобиться проводити пошук по простору підмножин ознак. Даний підхід має найменшу обчислювальну складність серед розглянутих підходів, а також відрізняється масштабованістю і простотою застосування. Подібні методи показують досить гарні результати на практиці.

Методи-фільтри, засновані на незалежному ранжируванні ознак, означають, що для кожної ознаки обчислюється деяка функція оцінки (ранг). В оптимальній підмножині ознак залишаються тільки ті ознаки, ранг яких вище деякого заздалегідь заданого значення порога. Значення порогу можна підібрати на крос-валідації, або ж розділити тренувальну вибірку на три підмножини, одна з яких буде використовуватися для підгонки значення параметра (на ньому будуть тестуватися різні класифікатори, отримані при різних значеннях порога), друге – для навчання класифікатора на остаточній підмножині ознак, і третє – для тестування отриманого класифікатора.

Нехай існує множина документів D, множина класів (рубрик) $C=\{c_1,...,c_{|M|}\}$, до кожного з яких можуть відноситися документи, і існує деяка цільова залежність $a^*: D\times C \to \{0, 1\}$, значення якої відомі лише на кінцевому наборі документів навчальної вибірки

$$D^n = \{(d_1, c_1), \text{K}, (d_n, c_n)\} \subset D\times C. \qquad (1.5)$$

Вихідні дані:

T – навчальна вибірка, тобто множина пар $\{(x_k, y_k)\}$;

x_k – k-й вхідний об'єкт, $k = 1,\ldots, m$;

y_k – вихідні змінні (класи), $k = 1,\ldots, m$, $y_k \in \{c_1,..., c_{|M|}\}$;

M – число класів;

n – число ознак;

X – множина ознак;

x^i – i-та ознака, $i = 1,\ldots, n$.

Задачею класифікації (рубрикації) текстів є пошук найкращого наближення a (його також називають алгоритмом через необхідність реалізації на комп'ютері) цільової залежності a^* на основі навчальної вибірки. Якість алгоритму визначається на контрольній вибірці $D^k \subset D$ – порівнюються відповіді, видані алгоритмом на контрольній вибірці, з істинними заздалегідь відомими для них відповідями за допомогою обраного функціоналу якості.

1.4.1 Приріст інформації (Information gain)

Будемо говорити, що ознака x^i виділяє деякий об'єкт x_k, якщо $x_k^i = 1$. Інтуїтивно ознака є тим більше інформативною, чим більше вона виділяє об'єктів деякого свого класу і чим менше вона виділяє об'єктів інших класів. У теорії інформації кількість інформації визначається у такий спосіб. Нехай є виходи $\omega_1, \omega_2, ..., \omega_n$ з ймовірностями $q_1, q_2, ..., q_n$, тоді кількість інформації, пов'язаної з результатом ω_i, по визначенню дорівнює $-\log_2 q_i$. Це математичне очікування числа біт, необхідних для запису інформації про реалізації виходів ω_i при використанні оптимального (найбільш ощадливого) кодування. Ентропія визначається як математичне очікування кількості інформації:

$$H(q_1, q_2, ..., q_n) = -\Sigma q_i \log_2 q_i \qquad (1.6)$$

В даній задачі будемо вважати подією ω_j появу об'єкта класу c_j, $j = \overline{1, M}$. Тоді ентропія вибірки T із класами c_j, $j = \overline{1, M}$ обчислюється:

$$H(c_1, c_2, ..., c) = -\Sigma P(c_j) \log_2 P(c_j), \quad j = \overline{1, M}. \qquad (1.7)$$

Припустимо, ми включили в множину ознак ознаку x^i. Ця ознака приймає значення 1 з імовірністю $P(x^i)$ і значення 0 з імовірністю $P(x'^i)$.

Приріст інформації (інформаційний виграш, information gain) для ознаки x^i визначається як різниця ентропій вибірок, отриманих без використання інформації ознаки x^i і з використанням цієї інформації:

$$G(x_i) = H(c_1, c_2, ..., c) - H(c_1, c_2, ..., c \mid x^i) - H(c_1, c_2, ..., c \mid x'^i) \qquad (1.8)$$

Значення приросту інформації для ознаки говорить про різницю біт інформації, необхідних для того, щоб класифікувати об'єкт із використанням ознаки x^i і без її використання.

1.4.2 Взаємна інформація

Це розповсюджений метод вибору ознак, заснований на підрахунку величини $A(t,c)$ – очікуваної взаємної інформації про термін t і клас c [6]. Цей показник вимірює кількість інформації про приналежності до класу c, що несе наявність/відсутність терміна.

$$I(U,C) = \sum_{e_t \in \{1,0\}} \sum_{e_c \in \{1,0\}} P(U = e_t, C = e_c) \log \frac{P(U = e_t, C = e_c)}{P(U = e_t) P(C = e_c)}. \qquad (1.9)$$

Тут U – випадкова величина, що приймає значення $e_t = 1$ (документ містить термін t) і $e_t = 0$ (документ не містить термін t), а змінна C – випадкова величина, що приймає значення $e_c = 1$ (документ належить класу c) і $e_c = 0$ (документ не належить класу c). Ця рівність стає еквівалентною (1.10) [6]:

$$I(U;C) = \frac{N_{11}}{N} \log_2 \frac{NN_{11}}{N_1 N_1} + \frac{N_{01}}{N} \log_2 \frac{NN_{01}}{N_0 N_1} + \\ + \frac{N_{10}}{N} \log_2 \frac{NN_{10}}{N_1 N_0} + \frac{N_{00}}{N} \log_2 \frac{NN_{00}}{N_0 N_0} \qquad (1.10)$$

Тут N – кількість документів, а індекси відповідають значенням змінних e_t і e_c. Наприклад, N_{10} – кількість документів, що містять термін t (тобто $e_t = 1$) і не належному класу c (тобто $e_c = 0$). Число $N_{10}+N_{11}$ – це кількість документів, що містять термін t (тобто $e_t = 1$) незалежно від приналежності до класу c (тобто $e_c = 0$). Число $N_{11}+N_{01}+N_{10}+N_{00}$ – загальна кількість документів.

1.4.3 Критерій χі-квадрат

Критерій χ^2 – це розповсюджений метод вибору ознак [6]. У статистиці критерій χ^2 використовується для перевірки незалежності двох випадкових подій, де події *A* и *B* вважаються незалежними, якщо *P(A|B)=P(A)* і *P(B|A)=P(B)*. При виборі ознак за двома подіями є поява терміна і класу. Після цього терміни ранжуються у відповідності з наступною величиною:

$$X^2(D,t,c) = \sum_{e_t \in \{1,0\}} \sum_{e_c \in \{1,0\}} \frac{(N_{e_t e_c} - E_{e_t e_c})^2}{E_{e_t e_c}}. \qquad (1.11)$$

Тут e_t і e_c – величини, визначені в рівнянні (1.1), тобто e_t = 1 (документ містить термін *t*) і e_t = 0 (документ не містить термін *t*), а значення e_c = 1 (документ належить класу *c)* і e_c = 0 (документ не належить класу *c)*. *N* – спостережувана (емпірична) частота в множині *D*, а *E* – очікувана частота. Наприклад, E_{11} – очікувана частота одночасної появи терміна *t* у документі і документа в класі *c* за умови, що термін і клас є незалежними.

Статистику χ^2 можна виразити більше простою формулою [6]:

$$X^2(D,t,c) = \frac{(N_{11} + N_{10} + N_{01} + N_{00}) \times (N_{11}N_{00} - N_{10}N_{01})^2}{(N_{11} + N_{01}) \times (N_{11} + N_{10}) \times (N_{10} + N_{00}) \times (N_{01} + N_{00})} \qquad (1.12)$$

Величина χ^2 дозволяє судити про те, наскільки сильно очікувана частота *E* и спостережувана *N* відхиляються друг від друга. Велике значення χ^2 означає, що гіпотеза про незалежність, з якої слідує близькість очікуваної і спостережуваної частот, не виконується. Якщо дві події залежать друг від друга, то поява терміна спричиняє появу документа в класі з більшою (або меншою) імовірністю. Саме цей факт лежить в основі застосування критерію χ^2 для вибору ознак.

1.4.4 Потужність ознаки (term strength)

Даний метод специфічний для задачі класифікації текстів. Метод полягає в оцінці значимості ознаки на основі того, наскільки часто ця ознака зустрічається у зв'язаних об'єктах. Для оцінки зв'язаності документів використовується косинусна міра. Нехай дана тренувальна множина документів, і

нехай α і β – зв'язані документи, а x^i – ознака. Тоді визначимо потужність ознаки в такий спосіб:

$$s^2(x^i) = P(x^i \in \beta \mid x^i \in \alpha) \qquad (1.13)$$

На відміну від попередніх методів, даний метод не використовує інформацію про класи. Тому його можна застосовувати в задачах навчання без учителя, які часто виникають при обробці текстів.

1.4.5 Алгоритм Relief

Алгоритм Relief споконвічно розглядався для випадку бінарної класифікації, але він може бути узагальнений і на випадок M-арної класифікації (Relief-F) [3]. Нехай у нас даний об'єкт з навчальної вибірки x_k, $k = \overline{1,m}$, і нехай його клас c_j , $j = \overline{1,M}$. Введемо функції near-hit (найближче влучення) і near-miss (найближчий промах) як відстань до найближчого об'єкта навчальної вибірки із класом c_j і як відстань до найближчого об'єкта навчальної вибірки із класом, відмінним від c_j, відповідно. Ідея цього методу в тому, що ознака тим більше релевантна об'єкту x_k, чим краще він розділяє x_k і його near-miss, і чим гірше він розділяє x_k і його near-hit. Алгоритм працює в такий спосіб:

– випадковим образом вибирається P об'єктів навчальної вибірки;

– для кожного об'єкта обчислюється його near-hit і near-miss (для цього використовується евклідова відстань);

– обчислюється і нормалізується вектор ваг $W = (w_1, ..., w_n)$ для ознак за формулою:

$$w_i = \Sigma\delta(x_k^i, \text{near\_miss}(x_k)^i)^2 - \delta(x_k^i, \text{near\_hit}(x_k)^i)^2, \quad k=1,..., p, \qquad (1.14)$$

де $\delta(a, b)$ − символ Кронекера;

– відбираються ті ознаки x_i, вага w_i яких перевищує заздалегідь задане значення порога τ.

1.5 Методи-обгортки

Методи-обгортки використовують алгоритм навчання як чорний ящик для оцінки конкретних підмножин ознак і вибирають підмножини інформативних ознак на основі їхньої інформативності для алгоритму навчання.

Принцип методів обгортки полягає в наступному:

– виконується пошук по простору підмножин вихідної множини ознак (як правило, у задачах класифікації текстів неможливо виконати прямий перебір усіляких підмножин вихідного набору ознак, тому використовують різні алгоритми пошуку);

– для кожного кроку пошуку використовується інформація про якість навчання на поточній підмножині ознак.

Отже, для застосування методів-обгорток необхідно, по-перше, вибрати стратегію пошуку по простору підмножин вихідної множини ознак, і, по-друге, вибрати функцію оцінки якості навчання на поточній підмножині ознак.

Існує три основних категорії алгоритмів пошуку в застосуванні до задачі вибору оптимального набору інформаційних ознак [3]. Це експонентні, послідовні і рандомізовані алгоритми пошуку.

1.5.1 Експонентні алгоритми пошуку

До цієї категорії відносять алгоритми, чия складність по числу операцій вибору наступного елемента множини є $O(2^n)$. Це значить, що на кожному кроці вони вибирають більше одного наступного стану.

До цієї категорії відносяться, зокрема, наступні алгоритми:

– повний перебір (exaustive search);

– алгоритм FOCUS, що виконує перебір спочатку по всіх підмножинах потужності 1, потім по всіх підмножинах потужності 2 і так далі (або у зворотному порядку, починаючи з підмножин потужності n).

– алгоритм галузей і границь (branch and bound), застосовний у випадку монотонної функції $J(X)$, що гарантує знаходження оптимальної підмножини X', яка максимізує $J(X)$, але при цьому потребує в багато разів меншої кількості операцій оцінки підмножини [6].

1.5.2 Послідовні алгоритми пошуку

До цієї категорії відносять алгоритми, які на кожному кроці вибирають рівно один наступний стан. Такі алгоритми не можуть повертатися до вже пройдених станів. Складність таких алгоритмів є $O(n)$. Якщо на кожному кроці допустити вибір не більше k наступних станів, то складність буде оцінюватися як $O(n^k)$. До цієї категорії відносяться наступні алгоритми:

– прямий жадібний алгоритм (forward greedy selection): починаючи з $X'=\varnothing$, на кожному кроці переходить у такий стан, у якому до поточної підмножини додається один елемент;

– зворотний жадібний алгоритм (backward greedy elimination): починаючи з підмножини, що зіставляє $X' = X$, на кожному кроці переходить у такий стан, у якому з поточної підмножини видаляється один елемент. Емпірично зворотний жадібний алгоритм дає звичайно кращі результати в порівнянні із прямим жадібним алгоритмом [2, 3]. Це пов'язане з тим, що зворотний жадібний алгоритм ураховує ознаки, інформативні в сукупності, але неінформативні, якщо розглядати їх окремо;

– алгоритм сходження на вершину (greedy hill climbing): є комбінацією двох попередніх методів, починаючи з деякої стартової підмножини X' , на кожному кроці переходить у сусідній стан, у якому або з поточної підмножини видаляється один елемент, або в нього додається один елемент. Стартову підмножину X' можна вибрати випадковим образом, задати $X' = X$ або використати інші евристики. Послідовні алгоритми пошуку мають значний недолік: вони можуть зупинитися в локальному максимумі функції $J(X)$, так і не відшукавши її глобальний максимум.

1.5.3 Рандомізовані алгоритми пошуку

До цієї категорії відносять алгоритми, що використовують рандомізацію для переходу в наступний стан. Це дозволяє уникнути зупинки в локальних максимумах. Однак час роботи таких алгоритмів заздалегідь невідомий, тому що найкращий розв’язок вони можуть знаходити в загальному випадку як завгодно довго. До цієї категорії відносяться наступні відомі алгоритми:

– фільтр Лас-Вегас;

– алгоритм симуляції отжигу (simulated annealing) [8]. Алгоритм симуляції отжигу моделює фізичний процес, що відбувається при кристалізації речовини. Передбачається, що атоми речовини вже вибудувалися в кристалічні ґрати, однак припустимі переходи окремих атомів з однієї комірки в іншу. Цей перехід відбувається з деякою ймовірністю, що зменшується зі зниженням температури;

– генетичні алгоритми (genetic approach) [7]. Генетичний алгоритм розглядає елементи множини, по якому відбувається пошук, як об’єктів з певними генотипами. Генотип для даної задачі являє собою бінарний вектор

довжини n, у якому 1 означає наявність ознаки в підмножині ознак, і 0 – його відсутність. Виконання генетичного алгоритму починається з того, що створюється множина генотипів початкової популяції. Це звичайно робиться випадковим образом. Вони оцінюються з використанням фітнес-функції (функції пристосованості, fitness function), що у даній задачі збігається з функцією оцінки $J(X)$. Потім з отриманої множини розв'язків, з урахуванням значення фітнес-функції, вибираються об'єкти, до яких застосовуються схрещування (crossover) і мутація (mutation), націлені на рандомізовану зміну об'єктів. Для них також обчислюється значення фітнес-функції, і потім виконується селекція (selection) – відбір кращих розв'язків для наступного покоління. Цей набір дій повторюється ітеративно, так моделюється еволюційний процес, що триває кілька життєвих циклів (поколінь), поки не буде виконаний критерій зупинки алгоритму.

1.6 Вбудовані методи (Embedded)

Вбудовані методи виконують вибір підмножини ознак у якості одного з етапів навчання, і тому специфічні для конкретної моделі. Переваги цих методів полягають у наступному:

– добре пристосовані до конкретної моделі;

– немає необхідності виділяти спеціальну підмножину, на якій тестується поріг функції рангу для методів-фільтрів або виконується пошук найкращої підмножини для методів-обгорток;

– як наслідок з попереднього пункту, при використанні цього методу менше ризик перенавчання класифікатора [2].

Вбудовані методи вибору оптимального набору ознак є специфічними для кожної конкретної задачі.

1.7 Постановка задачі

Метою цієї роботи є дослідження алгоритмів класифікації і розробка методу класифікації текстів по тематиках.

Для досягнення даної мети необхідно виконати наступні задачі.

1) Дослідити алгоритми класифікації текстів і відбору ознак.
2) Розробити метод класифікації для данної задачі.
3) Реалізувати додаток на основі розробленого методу класифікації.

4) Сформувати базу знань для подальшого навчання класифікатора на основі документів із заздалегідь відомими тематиками.

5) Протестувати алгоритм на деякому наборі текстових документів і провести порівняння з аналогічними системами.

6) Провести аналіз ефективності методу класифікації текстів.

2 АЛГОРИТМИ КЛАСИФІКАЦІЇ

2.1 Метричні алгоритми класифікації

Одним з найпростіших способів класифікувати об'єкт є пошук схожого по деяких параметрах уже класифікованого об'єкта. Якщо міра подібності об'єктів вибрана досить вдало, то, як правило, виявляється, що схожим об'єктам дуже часто відповідають схожі відповіді. У задачах класифікації це означає, що класи утворять компактно локалізовані підмножини. Це припущення прийнято називати гіпотезою компактності. Для формалізації поняття «подібності» вводиться функція відстані в просторі об'єктів. Методи навчання, засновані на аналізі подібності об'єктів, називаються метричними, навіть якщо функція відстані не задовольняє всім аксіомам метрики (зокрема, аксіомі трикутника).

2.1.1 Класифікатор Роше

Деякі класифікатори використовують так званий профайл (profile, прототип документа) для визначення категорії. Профайл – це список зважених термів, присутність (або відсутність) яких дозволяє найточніше відрізняти конкретну категорію від інших категорій. Метод, запропонований Дж. Роччіо (J. Rocchio), відноситься до лінійних класифікаторів, в яких кожен документ представляється у вигляді вектора вагових значень термів.

Класифікатор Роше проводить рубрикацію документа виходячи з його близькості до еталонів рубрик. Еталоном для рубрики c є вектор (w_1, w_2,...) у ознаковому просторі, обчислений за формулою:

$$w_i = \frac{\alpha}{|POS(c)|}\sum_{\alpha \in POS(c)} w_{di} - \frac{\beta}{|NEG(c)|}\sum_{\alpha \in NEG(c)} w_{di}, \qquad (2.1)$$

де POS(c) і NEG(c) – множини документів з навчальної вибірки, які належать і не належать рубриці c відповідно,

w_{di} – ваги i-ї ознаки документа d.

Звичайно, позитивні приклади набагато важливіше негативних, тому $\alpha > \beta$. Якщо β=0, то еталоном рубрики буде просто центроїд всіх її документів. Звідси назва цього часткового випадку класифікатора Роше – центроїдний алгоритм.

Класифікатор Роше дуже легко реалізується, а також не є ресурсномістким. Його якість, проте, теж посередня – особливо, якщо є рубрики, що представляють собою об'єднання незв'язаних кластерів.

Практичне дослідження методу Роше показало, що даний метод володіє високою ефективністю в розв’язанні задачі класифікації текстів. Однією з головних його особливостей є можливість змінювати вектор зваженого центроїда рубрики, без перенавчання класифікатора. Ця властивість може бути корисною, наприклад, у випадках, коли повчальна колекція часто поповнюється новими документами, а перенавчання займає дуже багато часу. Завдяки своїй результативності і простоті метод Роше став одним з найпопулярніших в області класифікації текстових документів і часто використовується як базовий, для порівняння ефективності різних класифікаторів.

Визначення направленості документа (позитивна чи негативна), визначається шляхом обчислення відстані між центроїдом кожної з характеристик – позитивної та негативної і вектором листа, що класифікується. Якщо відстань не перевершує деякого, заздалегідь заданого порогу, лист вважається позитивним.

Загальний опис алгоритму.

1) Занесення вхідних даних (текстів та файлу позитивних наборів) до масивів алгоритму.

2) Формування навчальної вибірки позитивних значень – POS.

3) Формування вектору документа d, що класифікується

4) Обчислення ваги документа w_k та центроїду вибірки позитивних значень c_k на кожному кроці k.

5) Обчислення центроїду профайлу документа .

6) Знаходження відстані між центроїдом документа та центроїдом кожного елементу з вибірки позитивних значень.

7) Підрахунок кількості позитивних відстаней.

8) Підрахунок кількості негативних відстаней як різниці між загальною кількістю та кількістю позитивних значень.

2.1.2 Класифікатори на основі вирішальних правил

У класифікаторах на основі вирішальних правил простір даних моделюється набором правил [7], на лівій стороні якого знаходиться умова на набір ознак, а на правій – мітка класу. Набір правил генерується з навчальної вибірки. Для даного тестового об'єкта ми одержуємо набір правил, для яких

він задовольняє їхню умову на лівій стороні. Потім визначаємо мітку класу як функцію від отриманих міток класу правил. Простір відповідей повинний бути покритий хоча б одним правилом, що досягається побудовою різних наборів правил для кожного класу і одного стандартного правила, що покриває всі інші.

У більш загальному випадку ліва сторона правила є логічною умовою, вираженою у вигляді диз'юнктивної нормальної форми. Наприклад, Honda & Toyota $\Rightarrow$ Машини, але частіше об'єднання умов, якщо це можливо, замінюється на кілька правил, у даному прикладі на два: правила Honda $\Rightarrow$ Машини і Toyota $\Rightarrow$ Машини без втрати інформативності.

2.2 Лінійні класифікатори

Лінійними називають класифікатори, в яких передбачене значення обчислюється у вигляді $c = w \cdot d + b$, де $d = (d_1, \ldots, d_l)$ – нормалізований вектор з частот слів у документі, w – вектор лінійних ваг тієї ж розмірності, що і ознаковий простір, а b – деяке скалярне значення. Природною інтерпретацією для c в дискретному випадку буде розподільна гіперплощина між різними класами.

Нарешті, прості нейронні мережі також є однією з форм лінійних класифікаторів. Найпростіша з них, відома як персептрон (або одношарова нейронна мережа) сама по собі створена для лінійного поділу і добре працює для класифікації документів. Проте, використовуючи кілька шарів нейронів, можливо узагальнити підхід для нелінійного поділу. Далі ми розглянемо різні лінійні методи для класифікації текстів.

2.2.1 Класифікатор SVM

У методі опорних векторів кожний документ відображається як вектор у багатовимірному просторі, кожний вимір якого відповідає квантитативній характеристиці лексем зі словника аналізованих текстових масивів.

Машина опорних векторів (Support Vectors Machine, SVM) була запропонована в [8] для числових даних. Головним принципом SVM є визначення роздільника в шуканому просторі, що розділяє класи щонайкраще. Наприклад, на рис. 2.1 ми маємо два класи, позначені через «x» і «o». На ньому також позначені три гіперплощини – *A*, *B*, і *C*. Очевидно, що гіперплощина *A* щонайкраще розділяє класи, тому що відстань від неї до будь-яких точок ма-

ксимально. Іншими словами, гіперплощина *A* максимізує зазор. Одна з переваг SVM-методу в тім, що так як він намагається визначити оптимальний напрямок поділу ознакового простору, розглядаючи комбінації ознак, він досить стійкий до великих розмірностей. Текстові дані ідеально підходять для класифікатора SVM через розріджені дані великої розмірності, що є відмінною рисою текстів.

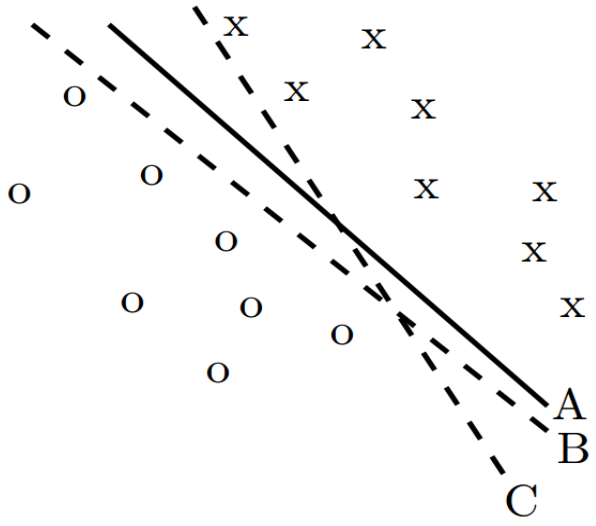

Рисунок 2.1 – Вибір кращої поділяючої поверхні

Задача пошуку кращого роздільника є задачею оптимізації, яка може бути зведена до задачі квадратичного програмування. Так, більшість методів використовують метод Ньютона для ітераційної мінімізації опуклої функції. Він буває досить повільний, особливо для даних високої розмірності, таких, як текстові документи. Розбиваючи велику задачу квадратичного програмування на набір менших задач, можна знайти ефективний розв’язок.

2.2.2 Алгоритм «найближчого сусіда»

Алгоритм методу найближчих сусідів ґрунтується на порівнянні відомих статистичних даних з новими елементами. Для нового запису, продовження якого необхідно спрогнозувати, знаходять найбільш подібні записи у минулому та ідентифікують їх як найближчих сусідів. Метод найближчих сусідів спирається на так звану «гіпотезу компактності» [8]. Ця гіпотеза стверджує, що різні відображення одного і того ж образу у просторі ознак породжують геометрично близькі точки, утворюючи компактні «згустки».

Алгоритм «найближчого сусіда» починається в довільній точці та поступово «відвідує» кожну найближчу точку, яка ще не була «відвідана». Пункти обходу плану послідовно включаються до маршруту. Причому кожен наступний пункт, що включається до маршруту, повинен бути найближчим до останнього вибраного пункту серед усіх інших, ще не включених до складу маршруту. Алгоритм завершується, коли «відвідано» всі точки. Точками маршруту є контентні складові документа.

Вхідні дані для множини точок V розмірністю N.

Вихідні дані: маршрут T, що складається з послідовності відвідування точок множини V.

Послідовність роботи алгоритму «найближчого сусіда» включає:

1) вибрати довільну точку V_1;

2) $T_1 = V_1$;

3) для $i = 2$ до $i = N$ виконати;

4) вибрати точку V_i , найближчу до точки T_{i-1};

5) $T_i = V_i$;

6) $T_{N+1} = V_1$;

7) кінець алгоритму та прийняття рішення про релевантне джерело [8]. Релевантне джерело обирається за мінімумом відстані V_i , T_{i-1}.

2.2.3 Нейронні мережі

Базовою одиницею нейронної мережі є нейрон. Кожен нейрон одержує набір входів, які будемо позначати d_i, які в нашій задачі будуть позначати входження термів в i-й документ. Кожен нейрон також асоційований з набором ваг w, які використовуються для підрахунку функції входів $f(u\cdot)$ [9]. Такою типовою функцією, що використовується в нейронній мережі, є лінійна:

$$p_i = w \cdot d_i . \qquad (2.2)$$

Таким чином, для вектора d_i і лексикона з d слів, вектор ваг w повинен також містити d елементів. Тепер розглянемо задачу бінарної класифікації, у якої мітки будуть із множини $\{+1, -1\}$. Припустимо також, що істинним класом d_i є y_i. У цьому випадку знак передбаченої функції p_i буде визначати саме мітку класу.

Головною ідеєю навчання є початковий довільний вибір ваг, а потім покрокове відновлення при наявності помилок функції на тренувальних да-

них. «Силу» відновлення на кожному кроці буде визначати параметр μ, який називається також швидкістю навчання. Таким чином ми одержали так званий персептронний алгоритм:

Вхід: Швидкість навчання μ, вибірка, що навчає (d_i, y_i), $i = 1 \ldots l$.

Вихід: Ваги w.

1) Ініціалізувати w довільними маленькими числами;

2) повторювати для $i = 1, \ldots, n$;

3) якщо знак $w \cdot d_i$ не збігається з y_i, то

4) обновити ваги w у відповідності зі швидкістю навчання μ;

5) кінець умови;

6) кінець циклу;

7) поки ваги w не стабілізуються.

Ваги w звичайно змінюються на величину, пропорційну $\mu\, d_i$. Також відзначимо, що багато різних способів поновлення було запропоновано в літературі. Наприклад, можна змінювати ваги кожного разу на μ, а не на $\mu \cdot d_i$. Це розумно робити в області класифікації текстів, де всі ознаки мають невеликі невід'ємні значення.

Природним питанням є те, як же використовувати нейронну мережу, коли класи можуть бути лінійно нероздільні. У цьому випадку, якщо використовувати багатошарові нейронні мережі, можна одержати більш повні класи поділюваних поверхонь. У таких мережах виходи одного шару подаються на входи нейронів наступного шару. Двох шарів досить для наближення скільки завгодно складних структур з багатокутників. Найпоширенішим алгоритмом навчання є алгоритм зворотного поширення помилки. Для класифікації текстів розгляд багатошарових нейронних мереж не дає значимого виграшу перед одношаровим персептроном [9].

2.3 Імовірнісні класифікатори

Імовірнісні класифікатори розглядають рішення про віднесення документа d до класу с як імовірність $P(c|d)$ належності цього документа до цього класу, і, відповідно, обчислюють її за теоремою Байєса:

$$P(c \mid d) = \frac{P(d \mid c)P(c)}{P(d)}. \qquad (2.3)$$

2.3.1 Наївний класифікатор Байєса

Метод Naive Bayes припускає обчислення вірогідності приналежності текстового документа до кожної рубрики. Рішення про приналежність приймається по максимальній вірогідності.

Властивості наївної класифікації: використання всіх змінних і визначення всіх залежностей між ними; наявність двох припущень відносно змінних; всі змінні є однаково важливими; всі змінні є статистично незалежними, тобто значення одній змінній нічого не говорить про значення іншої; більшість інших методів класифікації припускають, що перед початком класифікації вірогідність того, що об’єкт належить тому або іншому класу однакова, але це не завжди вірно.

Імовірність $P(d)$ не вимагає обчислень через те, що це константа для всіх рубрик. Щоб обчислити $P(d|c)$, потрібно зробити деякі припущення щодо документа *d.* При поданні документа у вигляді вектора ознак ($d = (t_1,...,t_n)$) найбільш загальним є припущення, що всі координати незалежні, тобто

$$P(d \mid c) = \prod_i P(w_i \mid c). \qquad (2.4)$$

Класифікатори, засновані на цьому припущенні, називають наївними класифікаторами Байєса. Їх називають «наївними», тому що припущення (2.4) ніколи не перевіряється. Проте, спроби опустити це припущення і скористатися імовірнісними моделями із залежностями координат поки не дали ніяких помітних поліпшень.

3 ПРОЕКТУВАННЯ СИСТЕМИ

Головною задачею, поставленою в даній роботі, є реалізація алгоритму класифікації тексту на основі методів машинного навчання.

Розв’язання цієї задачі можна розбити на наступні етапи.

1) Вибір ознак, за якими буде здійснюватися класифікація.

2) Вибір і реалізація алгоритма класифікації.

3) Створення навчальної вибірки.

4) Перевірка ефективності алгоритмів.

3.1 Інструментальні засоби

Для реалізації класифікатора використовувалися Java, JavaFX і бібліотека The Apache Lucene.

Java – об’єктно-орієнтована мова програмування, розроблена компанією Sun Microsystems (надалі придбаною компанією Oracle). Додатки Java звичайно транслюються в спеціальний байт-код, тому вони можуть працювати на будь-якій віртуальній Java-машині поза залежністю від комп'ютерної архітектури.

Мова Java – це об’єктно-орієнтована мова програмування, що веде свою історію від відомої мови C++. Але, на відміну від останньої, Java є мовою, що інтерпретується. Програми, написані цією мовою, здатні працювати в різних місцях мережі і не залежать від платформи, на якій виконуються написані на ній додатки. Java свідомо уникає арифметики з покажчиками й іншими ненадійними елементами, які присутні у C++, тому, розробляючи на ній додатки, ми позбавляємося багатьох проблем, звичайних при створенні програмного забезпечення.

Переваги мови Java.

1) Одна з основних переваг мови Java – незалежність від платформи, на якій виконуються програми: один і той же код можна запускати під управлінням операційних систем Windows, Solaris, Linux, Machintosh та ін. Це дійсно необхідно, коли програми завантажуються через Інтернет для подальшого виконання під управлінням різних операційних систем.

2) Синтаксис мови Java схожий на синтаксис мови C ++, і програмістам, що знають мови C і C ++, його вивчення не задає труднощів.

3) Крім того, Java – повністю об'єктно-орієнтована мова, навіть більшою мірою, ніж C ++. Всі сутності в мові Java є об'єктами, за винятком неба-

гатьох основних типів (primitive types), наприклад чисел. (За допомогою об'єктно-орієнтованого програмування легко розробляти складні проекти.)

4) Висока надійність. Автори забезпечили мову Java засобами, що дозволяють виключити саму можливість створювати програми, в яких були б приховані найбільш поширені помилки.

Для цього в мові Java зроблено наступне:

– виключена можливість явного виділення і звільнення пам'яті. Пам'ять у мові Java звільняється автоматично за допомогою механізму збірки сміття. Програміст гарантований від помилок, пов'язаних з неправильним використанням пам'яті;

– введені істинні масиви і заборонена арифметика покажчиків. Тепер програмісти в принципі не можуть стерти дані з пам'яті внаслідок неправильного використання покажчиків;

– виключена можливість переплутати оператор присвоювання з оператором порівняння на рівність;

– виключено множинне спадкування. Воно замінено новим поняттям – інтерфейсом, запозиченим з мови Objective C.

Інтерфейс дає програмісту майже все, що той може отримати від множинного спадкоємства, уникаючи при цьому складнощів, що виникають при управлінні ієрархіями класів.

3.1.1 Технологія JavaFX

JavaFX – платформа для створення Rich Internet Applications (RIA– вебдодаток, доступний через Інтернет, насичений функціональністю традиційних настільних додатків, що надається або унікальною специфікою браузера, або через плагін, або шляхом «пісочниці» (віртуальної машини)), дозволяє будувати уніфіковані додатки з насиченим графічним інтерфейсом користувача для безпосереднього запуску з-під операційних систем, роботи в браузерах і на мобільних телефонах, у тому числі, що працюють із мультимедійним вмістом.

JavaFX включає в себе набір утиліт, за допомогою яких веб-розробники та дизайнери можуть швидко створювати та надавати розвинуті інтернет застосунки для мобільних пристроїв, телебачення та інших платформ. Технологія JavaFX дозволяє створювати прикладні програми для роботи з мультимедійним контентом, графічні інтерфейси користувача для бізнес-прикладних програм та ін.

Прикладна програма JavaFX створюються за допомогою декларативної мови програмування JavaFX Script. Для розробки прикладних програм на мові JavaFX Script необхідно викачати і встановити JavaFX 1.0 SDK.

З коду, написаного на мові JavaFX Script, можна звертатися до будь-яких бібліотек Java. Тому спільне використання мов Java і JavaFX Script дозволяє вирішувати різноманітні задачі, наприклад, логіка бізнес-прикладних програм може бути написана на Java, а графічний інтерфейс користувача – на JavaFX Script (рис.3.1).

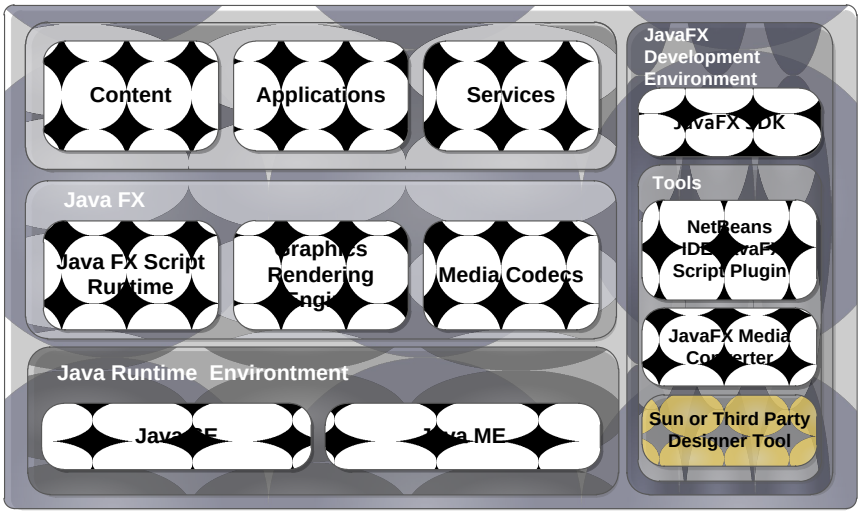

Рисунок 3.1 – Компоненти для створення RIA – прикладних програм

Прикладні програми, написані на мові JavaFX Script, можуть запускатися на комп'ютерах зі встановленим середовищем виконання Java 1.5 і вище. Нині підтримуються наступні операційні системи: Windows, Mac OS X, GNU/Linux і Solaris.

Дана платформа була обрана, тому що за допомогою неї зручно створювати прості користувальницькі інтерфейси на базі xml файлів.

3.1.2 Apache Lucene

The Apache Lucene – це вільна бібліотека для високошвидкісного повнотекстового пошуку, написана на Java. Може бути використана для пошуку в інтернеті і при розв’язанні різних задач обчислювальної лінгвістики. Наприклад, Lucene використовується як компонент у децентралізованій пошуковій системі YaCy (вільне ПЗ).

Lucene – найвідоміший з пошукових движків, споконвічно орієнтований саме на вбудовування в інші програми. Зокрема, його широко використовують в Eclipse (пошук по документації) і навіть в IBM (продукти із серії OmniFind). В плюсах проекту – розвинені можливості пошуку, хороша система побудови та зберігання індексу, який може одночасно поповнюватися і оптимізуватися разом з пошуком. Доступний і паралельний пошук по безлічі індексів з об'єднанням результатів. Сам індекс побудований із сегментів, проте для поліпшення швидкості рекомендується його періодично оптимізувати. Спочатку присутні варіанти аналізаторів для різних мов, включаючи російську з підтримкою стемінга (приведення слів до нормальної форми). Однак мінусом є все ж дуже низька швидкість індексації (особливо в порівнянні з Sphinx), складність роботи з базами даних і відсутність API (крім рідної Java). І хоча для досягнення серйозних показників Lucene може кластеризуватися і зберігати індекси в розподіленої файлової системи або базі даних, для цього потрібно сторонні рішення, так само як і для всіх інших функцій – наприклад, спочатку він уміє індексувати тільки звичайний текст. Але саме в плані використання в складі сторонніх продуктів Lucene «попереду планети всієї» – жоден інший движок не має стільки портів на інші мови. Одним з чинників такої популярності є і дуже вдалий формат файлів індексів, який використовують сторонні рішення.

Можливості:

– масштабована і високошвидкісна індексація: понад 95GB на годину на сучасному обладнанні; розмір індексу приблизно 20-30% від розміру початкового тексту;

– потужний, точний і ефективний пошуковий алгоритм: ранжований пошук – найкращі результати показуються першими; безліч потужних типів запитів: запит фрази, шаблонні запити, пошук інтервалів і т.д.; пошук заснований на «полях» (таких як заголовок, автор, текст); можливість сортувати по різних полях; множинний індексний пошук з можливістю об'єднання результатів; можливість одночасного пошуку та оновлення індексу;

– крос-платформене рішення: вихідний код повністю написаний на Java; наявність портів на інші мови програмування (C, C + +, Perl та ін.).

Побудова пошукового індексу.

Основним джерелом даних, використовуваним в Lucene в процесі пошуку, є індекс. Індекс являє собою сховище, яке включає в себе безліч доку-

ментів. Основною складовою документів є поля з даними. У Lucene основним типом даних є String. Існує також підтримка числових типів і типів Date з різною точністю збереження, але за замовчуванням Lucene надає досить небагато коштів для підтримки цих типів даних.

Для побудови пошукового індексу потрібно створити множину документів з відповідними значеннями полів і додати створені документи в індекс. Схематично процес побудови індексу представлений на рис.3.2.

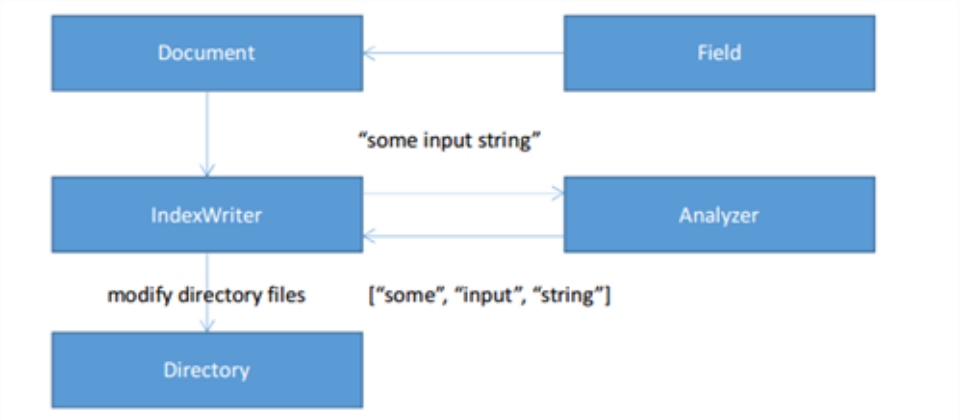

Рисунок 3.2 – Побудова індексу

При побудові індексу потрібно вказати використовуваний аналізатор (Analyzer) і директорію (Directory). Директорія відповідає за зберігання даних індексу. Формально директорія – це абстрактне сховище, що реалізує заданий інтерфейс. Інтерфейс включає в себе методи для роботи з файлами – їх створення, зміна та видалення. Кожен файл характеризується ім'ям і вмістом. Таким чином, логіка роботи директорії прямо не пов'язана зі структурою індексу. Найбільш поширеними реалізаціями цього інтерфейсу є класи SimpleFSDirectory і RAMDirectory, що дозволяють зберігати вміст індексу в файлової системи і в пам'яті відповідно.

Аналізатор відповідає за перетворення значень полів документа в послідовність токенів. Приклад такого перетворення показаний на рис.3.2. Токени – це ключові слова, за якими ведеться пошук. В індексі зберігається приналежність тих чи інших токенів до відповідних полів документів, що додаються.

IndexWriter є сполучною ланкою між цими класами і виконує роботу по збереженню даних про документи індексу, їх полях і токенах з використанням наданого аналізатора і директорії. Ця бібліотека використовується для виділення морфеми слова.

3.2 Алгоритм відбору ознак

Для поліпшення алгоритму класифікації будемо використовувати один з алгоритмів відбору ознак. Це дозволить при побудові моделі вибрати тільки самі показові ознаки (наприклад, слова) і відсіяти інші.

У задачі класифікації документів показова ознака – це слова, які несуть інформацію про клас, до якого відноситься документ. Наприклад, у контексті автомобільної тематики слово «Nissan», швидше за все, буде показовою ознакою, а от слово «новий» – навряд чи. Використання для класифікації не всіх слів, а тільки показових, дає кілька переваг.

По-перше, відбір ознак дозволяє значно зменшити кількість параметрів моделі (використовуваних для класифікації слів), і, як наслідок, знижує вимоги до об'єму пам'яті, необхідної для класифікації.

По-друге, відбір ознак може підвищити точність алгоритму за рахунок видалення з моделі слів з низьким співвідношенням сигнал/шум. Наприклад, слово «аскетичний» зустрілося в навчальній вибірці 3 рази, 1 раз у рамках класу «Авто» і 2 рази в рамках класу «Одяг». Формально воно переважає на користь класу «Одяг», але навряд чи це слово можна вважати гарною класифікаційною ознакою. Швидше за все, перевага убік одягу – це випадковість.

Якщо взяти будь-який досить великий корпус документів і підрахувати, скільки в ньому слів, що зустрічаються рівно 1 раз, 2 рази, 3 рази і т.д., то виявиться, що більшість слів зустрічаються 1 раз.

Для виділення відбірних ознак використовується статистика χ^2, тому що вона проста в реалізації, має високу ефективність і швидкість роботи.

3.3 Векторна модель мови

Якість майбутнього класифікатора залежить від векторної моделі (моделі представлення текстів). Іншими словами, обраний набір ознак вектора має величезний вплив на точність майбутнього класифікатора.

Одна з найбільш часто використовуваних моделей – модель *n*-грам. В рамках моделі, вектори ознак являють собою послідовності з *n* слів. *n* може бути будь-яким цілим числом більше нуля; якщо *n* дорівнює одиниці, то така модель називається «уніграмм», якщо *n* дорівнює двом – «біграмм», трьом – «триграм», і так далі. У найпростішому випадку моделі уніграмм, вектор ознак являє собою набір всіх слів їх тексту, – тобто кожне слово є термін. У разі біграмм термін утворює пари слів з тексту. Це означає, що пара слів

розглядається як ознака вектора. Для прикладу розглянемо пропозицію ‘Jane prefers coffee to tea’. Воно може бути представлене як вектор уніграмм: [Jane; prefers; coffee; to; tea]. Крім цього, воно може бути виражено як набір біграмм: [[Jane prefers]; [Prefers coffee]; [Coffee to]; [To tea]]. Також може розглядатися комбінація уніграмм і біграмм окремо: [Jane; prefers; coffee; to; tea; [Jane prefers]; [prefers coffee]; [coffee to]; [to tea]].

Символьні *n*-грами повинні бути розглянуті окремо. Символьна *n*-грама означає *n* символів з тексту, що йдуть підряд. Наприклад, символьні біграми слова 'word' будуть виглядати наступним чином: '_w', 'wo', 'or', 'rd' and " d_. Символьні триграми того ж слова будуть: '_wo', 'wor', 'ord' and 'rd_'. Це може здаватися примітивним способом, однак, подібна векторна модель мови показує непогані результати в деяких роботах [8, 9].

В першу чергу, необхідно представити колекцію документів у вигляді матриці термін-документ. Рядки будуть позначати окремі документи (тексти), а колонки – словник вибірки, що містить в собі всі слова в колекції документів.

Проілюструємо прикладом: дано два коротких документа; вони є прикладом найпростішої навчальної вибірки.

‘Jane likes coffee and tea’

‘Jane also likes cookies’

На даному етапі може бути сформований словник вибірки; він буде містити всі слова з текстів навчальної вибірки. Далі можна буде сформувати матрицю термін-документ: значення в кожному осередку показує, чи зустрічається певний термін в определенія документі (1) чи ні (0) (табл.3.1).

Таблиця 3.1 – Матриця термін-документ

	Jane	likes	coffee	and	tea	also	cookies
1ий текст	1	1	1	1	1	0	0
2ий текст	1	1	0	0	0	1	1

Таким чином, виходять бінарні вектори: d_1 = [1,1,1,1,1,0,0] і d_2 = =[1,1,0,0,0,1,1].

За допомогою моделі n-грам можна отримати набір векторів, які в подальшому можуть бути порівняні один з одним, шляхом обчислень відстаней між ними. Чим ближче вектори один до одного – тим більше імовірність того, що вони належать до однієї предметної області.

Під відстанню між векторами може розумітися, наприклад, Евклідова відстань:

$$\rho(x, x') = \sqrt{\sum_{i}^{n} (x_i - x_i^{'})^2}\,, \tag{3.1}$$

або максимум модулів:

$$\rho(x, x') = \max_{i} \left| x_i - x_i^{'} \right|, \tag{3.2}$$

або інша функція відстані.

Вектор ознак – це алгебраїчна модель представлення текстів:

$$d_{\text{j}} = (w_{1\text{j}}, w_{2\text{j}}, ..., w_{\text{nj}}), \tag{3.3}$$

де d_{j} – векторне подання документа *j*;

w_{ij} – вага терміна *i* в документі *j;*

n – кількість всіх термінів у вибірці.

Для повноцінного розуміння принципу векторної моделі тексту необхідно пояснити, як саме розраховуються ваги векторів. Існує кілька базових функцій ваг. Pang і співавтори в своєму дослідженні [10] виявили, що бінарна функція зважування векторів більш ефективна. Це означає, що наявність терміна в документі важливіше, ніж його частота. Бінарні вектори представлені як послідовність нулів і одиниць: якщо конкретний термін зі словника вибірки зустрічається в тексті – вага терміна буде дорівнювати 1, інакше – 0. Частотні вектори формуються на основі кількості входжень певного терміну в класі документів.

3.3 Алгоритм класифікації

В основі NBC (Naive Bayes Classifier) лежить теорема Байєса:

$$P(c \mid d) = \frac{P(d \mid c) P(c)}{P(d)}, \tag{3.4}$$

де $P(c|d)$ – імовірність того, що документ *d* належить класу *c*, саме її нам треба розрахувати;

$P(d|c)$ – імовірність зустріти документ d серед всіх документів класу c;

$P(c)$ – безумовна імовірність зустріти документ класу c у корпусі документів;

$P(d)$ – безумовна імовірність документа d у корпусі документів.

Класифікатор Байєса використовує оцінку апостеріорного максимума (Maximum a posteriori estimation) для визначення найбільш імовірного класу. Інакше кажучи, це клас із максимальною ймовірністю.

$$c_{map} = \arg\max_{c \in C} \frac{P(d \mid c)P(c)}{P(d)}. \tag{3.5}$$

Тобто нам треба розрахувати ймовірність для всіх класів і вибрати той клас, що має максимальну ймовірність. Знаменник (імовірність документа) є константою і ніяк не може вплинути на ранжирування класів, тому в нашій задачі ми можемо його ігнорувати.

$$c_{map} = \arg\max_{c \in C} \left[P(d \mid c)P(c) \right]. \tag{3.6}$$

Далі робиться припущення, яке пояснює, чому цей алгоритм називають наївним.

Припущення умовної незалежності.

У натуральній мові імовірність появи слова сильно залежить від контексту. Класифікатор Байєса представляє документ як набір слів, імовірності яких умовно не залежать друг від друга. Цей підхід іноді ще називається bag of words model. Виходячи із цього припущення, умовна ймовірність документа апроксимується добутком умовних імовірностей всіх слів, що входять у документ.

$$P(d \mid c) \approx P(w_1 \mid c)P(w_2 \mid c)...P(w_n \mid c) = \prod_{i=1}^{n} P(w_i \mid c). \tag{3.7}$$

Підставивши отриманий вираз в (3.4) одержимо:

$$c_{map} = \arg\max_{c \in C} \left[P(c) \prod_{i=1}^{n} P(w_i \mid c) \right]. \tag{3.8}$$

Проблема арифметичного переповнення.

При досить великій довжині документа доведеться перемножувати велика кількість дуже маленьких чисел. Для того, щоб при цьому уникнути арифметичного переповнення знизу, найчастіше користуються властивістю логарифма добутку $\log ab = \log a + \log b$. Так як логарифм – функція монотонна, її застосування до обох частин виразу змінить тільки його чисельне значення, але не параметри, при яких досягається максимум. При цьому, логарифм від числа близького до нуля буде числом від'ємним, але в абсолютному значенні істотно більшим, ніж вихідне число, що робить логарифмічні значення ймовірностей більш зручними для аналізу. Тому перепишемо нашу формулу з використанням логарифма.

$$c_{map} = \arg\max_{c \in C} \left[\log P(c) + \sum_{i=1}^{n} \log P(w_i \mid c) \right]. \tag{3.9}$$

Основа логарифма в цьому випадку не має значення. Можна використовувати як натуральний, так і будь-який інший логарифм.

Оцінка параметрів моделі Байєса.

Оцінка ймовірностей $P(c)$ і $P(w_i|c)$ здійснюється на навчальній вибірці. Імовірність класу ми можемо оцінити як:

$$P(c) = \frac{D_c}{D}, \tag{3.10}$$

де D_c – кількість документів, що належать класу c,

D – загальна кількість документів у навчальній вибірці.

Оцінка ймовірності слова в класі:

$$P(w_i \mid c) = \frac{W_{ic}}{\sum_{i' \in V} W_{i'c}}, \tag{3.11}$$

де W_{ic} – кількість разів, скільки i-те слово зустрічається в документах класу c;

V – словник корпуса документів (список всіх унікальних слів).

Інакше кажучи, чисельник описує скільки разів слово зустрічається в документах класу (включаючи повтори), а знаменник – це сумарна кількість слів у всіх документах цього класу.

Проблема невідомих слів. З формулою (3.11) є одна невелика проблема. Якщо на етапі класифікації нам зустрінеться слово, якого ми не бачили на етапі навчання, то значення W_{ic}, а відповідно і $P(w_i|c)$ будуть дорівнюють нулю. Це приведе до того, що документ із цим словом не можна буде класифікувати, тому що він буде мати нульову ймовірність по всіх класах. Позбутися цієї проблеми шляхом аналізу більшої кількості документів не можна. Ми ніколи не зможемо скласти навчальну вибірку, утримуючу всі можливі слова, включаючи неологізми, помилки, синоніми і т.д. Типовим рішенням проблеми невідомих слів є адитивне згладжування (згладжування Лапласа). Ідея полягає в тому, що ми вдаємо, ніби бачили кожне слово на один раз більше, тобто додаємо одиницю до частоти кожного слова.

$$P(w_i \mid c) = \frac{W_{ic} + 1}{\sum_{i' \in V} (W_{i'c} + 1)} = \frac{W_{ic} + 1}{|V| + \sum_{i' \in V} W_{i'c}}. \qquad (3.12)$$

Підставивши обрані нами оцінки у формулу (3.9) ми одержуємо остаточну формулу, по якій відбувається класифікація по Байєсу.

$$c_{map} = \arg\max_{c \in C} \left[\log \frac{D_c}{D} + \sum_{i=1}^{n} \log \frac{W_{ic} + 1}{|V| + \sum_{i' \in V} W_{i'c}} \right]. \qquad (3.13)$$

Класифікатор.

Для реалізації класифікатора Байєса нам необхідна навчальна вибірка, в якій проставлені відповідності між текстовими документами і їхніми класами. Потім нам необхідно зібрати наступну статистику з вибірки, що буде використовуватися на етапі класифікації:

– відносні частоти класів у корпусі документів. Тобто, як часто зустрічаються документи того або іншого класу;

– сумарна кількість слів у документах кожного класу;

– відносні частоти слів у межах кожного класу;

– розмір словника вибірки. Кількість унікальних слів у вибірці.

Сукупність цієї інформації ми будемо називати моделлю класифікатора. Потім на етапі класифікації необхідно для кожного класу розрахувати значення наступного виразу і після цього вибрати клас із максимальним значенням.

Спрощений запис формули (3.13)

$$c_{map} = \log \frac{D_c}{D} + \sum_{i=Q}^{n} \log \frac{W_{ic} + 1}{|V| + L_c}, \tag{3.14}$$

де D_c – кількість документів у навчальній вибірці, що належить класу c;

D – загальна кількість документів у навчальній вибірці;

$|V|$ – кількість унікальних слів у всіх документах навчальної вибірки;

L_c – сумарна кількість слів у документах класу c у навчальній вибірці;

W_{ic} – скільки разів i-те слово зустрічалося в документах класу c у навчальній вибірці;

Q – множина слів документа, що класифікується (включаючи повтори).

Формування імовірнісного простору.

Оцінки, які видає алгоритм, не задовольняють двом формальним властивостям, яким повинні задовольняти всі імовірнісні оцінки:

– вони повинні бути в діапазоні від нуля до одиниці;

– їхня сума повинна дорівнювати одиниці.

Для того, щоб розв’язати цю задачу, необхідно з логарифмічних оцінок сформувати імовірнісний простір. А саме: позбутися від логарифмів і нормувати суму по одиниці.

$$P(c \mid d) = \frac{e^{q_c}}{\sum_{c' \in C} e^{q_{c'}}}. \tag{3.15}$$

Тут q_c – це логарифмічна оцінка алгоритму для класу c, а піднесення e (основа натурального логарифма) у степінь оцінки використовується для того, щоб позбутися логарифма $a^{\log_a x} = x$.

3.4 Взаємодія користувача з додатком

1) У користувача повинна бути можливість імпортувати навчальну вибірку в програму за допомогою текстового файлу зі зручною структурою;

2) можливість зберігати навчений класифікатор;

3) можливість імпортувати навчений класифікатор у програму;

4) текст для визначення тематики можна завантажити через файл або записати у відповідному полі;

5) відображення результатів у вигляді процентного відношення обумовленого тексту до кожної з категорій.

4 ПРОГРАМНА РЕАЛІЗАЦІЯ АЛГОРИТМУ КЛАСИФІКАЦІЇ

4.1 UML діаграма і опис класів

Основні класи додатка розділені на пакети.

1) Пакет dataobjects містить у собі класи для зручного зберігання даних.

Document – зручна структура даних для використовування документів у програмі.

а) Map<String, Integer> tokens список токенів (ключових слів) і кількість їхніх повторень у документі;

б) String category – категорія документа.

FeatureStats – містить різну статистику документів.

а) int documentCount – загальна кількість документів;

б) Map<String, Map<String, Integer>> featureCategoryJointCount – дані про те, скільки разів кожне ключове слово повторилося в кожному класі документів;

в) Map<String, Integer> categoryCounts – інформація про те, скільки документів було привласнено до кожної з категорій.

NaiveBayesKnowledgeBase – база знань класифікатора.

а) int categoryCount – загальна кількість класів документів;

б) int documentCount – загальна кількість документів;

в) int featuresCount – загальна кількість ключових слів;

г) Map<String, Double> logPriors – апріорна ймовірність кожної категорії;

д) Map<String, Map<String, Double>> logLikelihoods – оцінка ймовірності кожного слова в кожному класі.

2) Пакет features містить у собі класи для роботи з ознаками документів.

FeatureExtraction – клас для добування ознак документів, містить у собі 2 методи:

а) extractFeatureStats – створює об'єкт FeatureStats зі списку документів (Document);

б) chisquare – реалізація алгоритму відбору ознак Хі-квадрат.

TextTokenizer – використовується для токенізації тексту і конвертації його в об'єкт Document.

а) LuceneMorphology luceneMorph – об'єкт класу LuceneMorphology, що є класом бібліотеки Lucene. Використовується для нормалізації слів;

б) String preprocess(String text) – метод для попередньої обробки тексту, видалення пунктуаційних знаків, зайвих пробілів, зведення тексту до нижнього регістра;

в) String[] extractKeywords(String text) – метод для добування з тексту ключових слів;

г) Map<String, Integer> getKeywordCounts – метод для одержання кількості входжень ключових слів усередині тексту.

3) Пакет classifier містить в собі єдиний клас NaiveBayes, що є реалізацією наївного класифікатора Байєса:

а) NaiveBayesKnowledgeBase knowledgeBase – база знань класифікатора;

б) List<Document> preprocessDataset(Map<String, String[]> trainingDataset) – метод для попередньої обробки набору даних. Виділення ключових слів, очищення тексту від «зайвих» символів. Конвертація набору даних в об'єкт типу Document. Створення списку типу Document з набору даних;

в) FeatureStats selectFeatures(List<Document> dataset) – метод для відбору ознак зі списку документів. У ньому використовуються методи extractFeatureStats класу FeatureExtraction і chisquare;

г) Map<String, Double> createProbabilitySpace(Map<String, Double> categoryProbabilities) – формування імовірнісного простору;

д) void train(Map<String, String[]> trainingDataset, Map<String, Double> categoryPriors) – навчання Наївного класифікатора Байєса за мультіноміальною моделлю;

е) Map<String, Double> predict – метод для прогнозування категорії невідомого тексту за допомогою навченого класифікатора. Повертає Map с імовірностями відношення невідомого документа до кожної з категорій.

4) У пакеті ui_examples, містяться класи, відповідальні за користувальницький інтерфейс.

5) Інші класи:

а) Utils – містить різні методи для роботи з файлами і серіалізації об'єктів.

Загальна UML-діаграма основних класів зображена на рис.4.1.

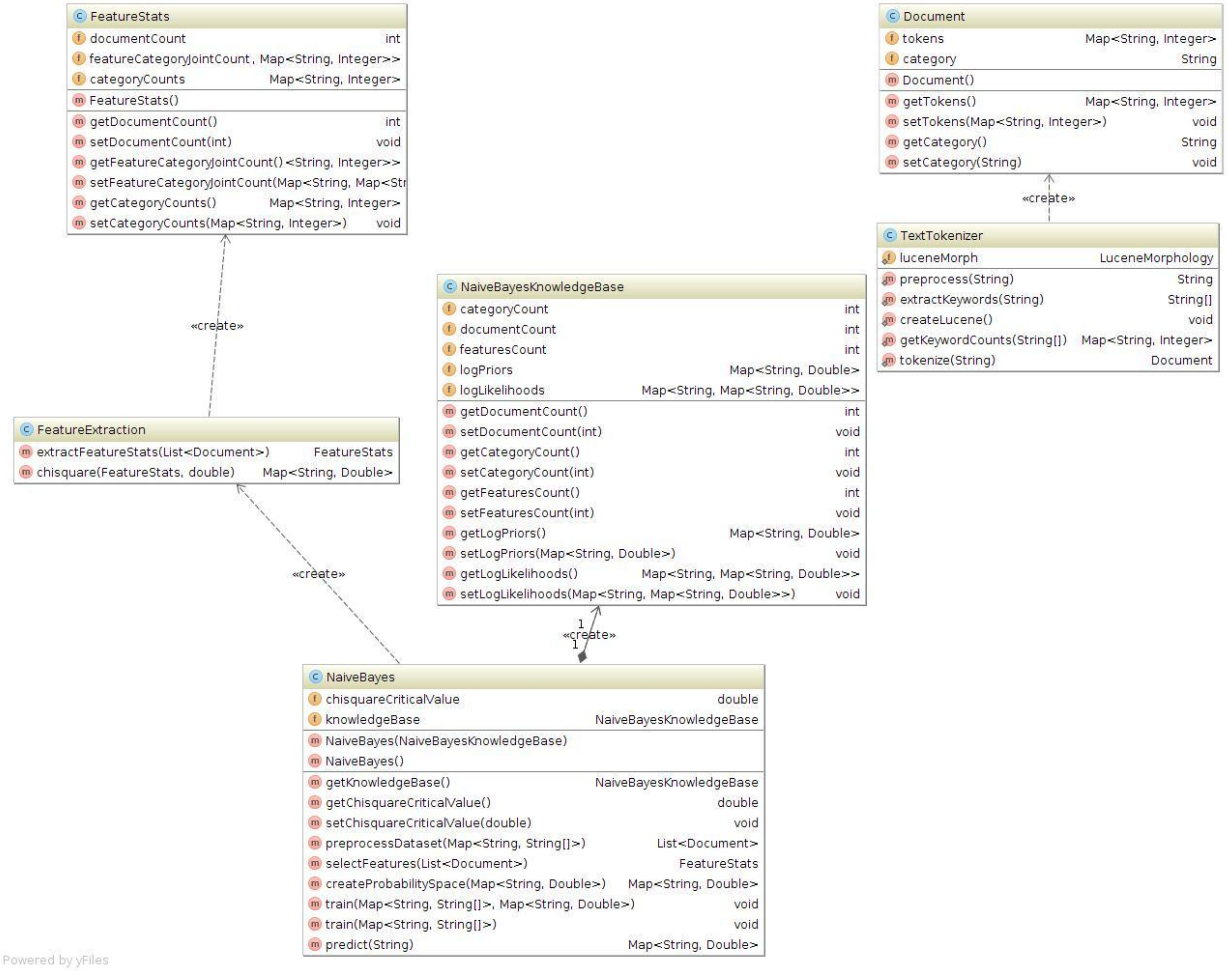

Рисунок 4.1 – Загальна UML-діаграма основних класів

4.2 Алгоритм роботи додатка

Алгоритм навчання класифікатора.

1) Завантаження навчальної вибірки документів в пам’ять програми.

2) Одержання списку документів у вигляді List<Document> dataset.

3) Відбір ознак методом Хі-квадрат зі списку документів dataset за допомогою об'єкта FeatureExtraction. Одержання об'єкта FeatureStats.

4) Створення бази знань класифікатора. Навчання класифікатора.

а) Такі поля, як categoryCount (кількість документів бази знань), featuresCount (кількість обраних слів) і categoryCount (кількість категорій) одержуємо з об'єкта FeatureStats.

б) Обчислення поля logPriors (апріорна ймовірність кожного класу) відбувається за формулою (2.3).

в) Одержуємо загальне число ключових слів для кожної категорії.

г) Обчислення ймовірності знаходження кожного слова в кожному із класів документів виконується за формулою 2.3.

Формат навчальної вибірки являє собою набір текстових файлів, по одному файлу для кожної категорії, у якому розташовані всі документи навчальної вибірки даної категорії, розділені рядками, що починаються із символів «:::». Ім'я файлу з текстами деякої категорії збігається з назвою категорії. Приклади такої навчальної вибірки представлені на рис. 4.2, 4.3.

http://sport.bigmir.net/football/countryteam/1712147-Sbornaja-Ukrainy-primet-Belarus--vo-L-vove

В среду, 6 мая, Исполком Федерации футбола Украины определил место проведения матча квалификационного раунда чемпионата Европы 2016 года между национальными сборными командами Украины и Беларуси.

Поединок, который состоится 5 сентября, решено провести на стадионе Арена Львов во Львове. Начало встречи - в 19:00.

Напомним, первый матч между этими командами, который проходил в прошлом году в Борисове (Беларусь), завершился победой Украины со счетом 2:0

Рисунок 4.2 – Приклад першого документа з навчальної вибірки для категорії «Спорт»

http://lenta.ru/news/2014/04/01/gusev/

Полузащитник киевского «Динамо» Олег Гусев частично потерял память в результате столкновения с вратарем «Днепра» Денисом Бойко, сообщает украинский телеканал «Футбол 1». Инцидент произошел 30 марта в Днепропетровске в первом тайме матча этих команд в рамках чемпионата Украины. Жизнь Гусева находилась под угрозой.

В ходе борьбы за мяч Бойко попал Гусеву коленом в голову. Киевлянин потерял сознание, причем первую помощь ему оказали футболисты. Полузащитник «Днепра» Джаба Канкава сумел вытащить запавший язык Гусева, чем спас ему жизнь. Футболиста унесли с поля на носилках, после чего увезли на машине «скорой помощи».

Позднее в Киеве Гусеву сделали компьютерную томографию головного мозга. Как рассказал заведующий отделением Игорь Конивец, игроку поставлен диагноз «закрытая черепно-мозговая травма, сотрясение головного мозга, ретроградная амнезия». Гусев помнит, как подавался штрафной у ворот «Днепра», а затем – как он оказался в машине «скорой помощи». Как ему оказывали помощь и выносили со стадиона, игрок не помнит.

Матч «Днепра» с киевским «Динамо» завершился со счетом 2:0 в пользу команды из Днепропетровска. В турнирной таблице чемпионата Украины «Днепр» идет на втором месте, «Динамо» – на третьем. Лидирует донецкий «Шахтер».

Рисунок 4.3 – Приклад другого документа з навчальної вибірки для категорії «Спорт»

Алгоритм визначення категорії для невідомого документа.

1) Завантаження невідомого документа в пам’ять програми.
2) Одержання об'єкта Document із завантаженого тексту.
3) Визначення категорії невідомого тексту (рис.4.4).

```
  predict(String text)
for по каждой априорной вероятности {
    for по словам неизвестного док-та Document {
        if(в базе знаний нет i-го слова) {
            пропускаем его
        }

        occur = кол-во вхождений слова в текст,
        который необходимо классифицировать

        logprob = /*Сумма произведений числа
        вхождений слов в
        документ на число вероятности
        встречи слова в
        этой категории

                        P(c) + вероятность
                        встретить слово
                        feature в категории category*
    }
}
return вероятность отношения текста к каждой категории
```

Рисунок 4.4 – Алгоритм визначення категорії для невідомого документа

4.3 Розробка користувальницького інтерфейсу

У додатка є консольний інтерфейс і віконний користувальницький інтерфейс, реалізований відповідно до паттерну MVC.

Класи, відповідальні за віконний користувальницький інтерфейс, знаходяться у пакеті ui_examples:

1) Controller – містять оброблювачі події, пов'язані із взаємодією користувача з додатком;

а) onClickTrain – обробляє натискання кнопки «Навчання класифікатора». Відкриває діалогове вікно для вибору файлу з навчальною вибіркою;

б) onClickLoad – обробляє натискання кнопки «Завантажити базу знань». Відкриває діалогове вікно для вибору серіалізованої бази знань;

в) onClickPredict – залежно від стану radioGroup відкриває діалогове вікно, у якому потрібно вибрати файл із текстом для визначення тематики, або зчитує текст із відповідної форми;

г) showKnowledgeBaseInfo – метод для виводу інформації про базу знання класифікатора;

д) List<File> createChooser(String title) – метод для створення діалогу вибору файлу;

е) void createErrorDialog – метод для створення діалогу з повідомленням про помилку.

2) Model – клас для зберігання в пам'яті додатка моделі класифікатора.

а) List<File> trainingFiles – список файлів для навчання класифікатора;

б) List<File> testingFiles – список файлів з невідомими документами, які потрібно класифікувати;

в) knowledgeBase – база знань класифікатора;

г) void trainClassifier() – метод-обгортка для навчання класифікатора;

д) Map<String, Map<String, Double>> predict – метод-обгортка для класифікування невідомого документа.

3) NaiveBayesUIExample – точка входу в додаток. Визначення дизайну додатка з файлу ui.fxml.

4) Файл ui.fxml – файл, у якому описаний дизайн користувальницького інтерфейсу.

У пакеті console_example знаходиться клас NaiveBayesConsoleExample, що відповідає за консольний користувальницький інтерфейс.

4.4 Застосування додатку

Після запуску додатка з'являється вікно (рис.4.5).

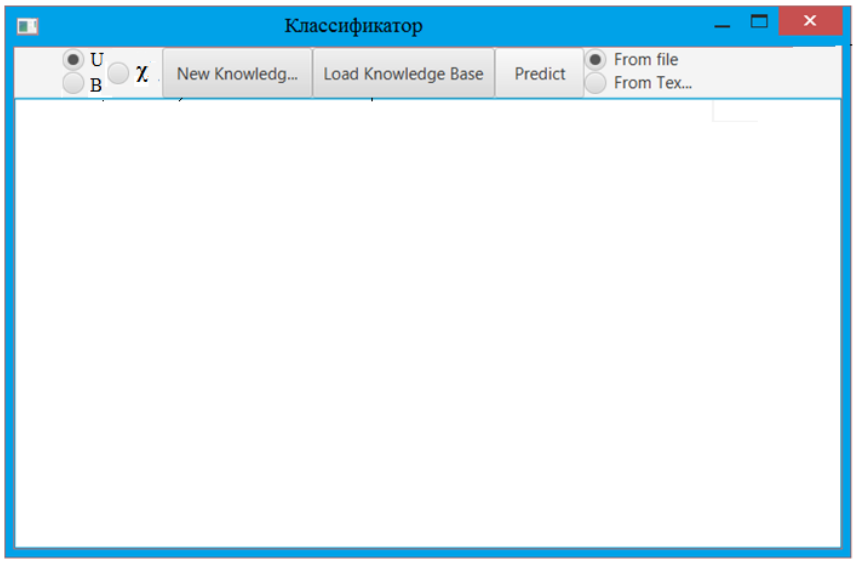

Рисунок 4.5 – Початок роботи із класифікатором

В ньому користувачеві пропонується створити базу знань (кнопка NewKnowledgeBase), завантажити базу знань (кнопка LoadKnowledgeBase), почати класифікацію тексту (кнопка Predict), а так само два «радіо-баттона», за допомогою яких можна вибрати, звідки брати текст для класифікації – з файлу, або ввести його вручну (From file й From text відповідно).

Для початку роботи із класифікатором необхідно створити базу знань.

При натисканні на кнопку NewKnowledgeBase з'явиться спливаюче вікно з можливістю вибору текстових файлів для тренування класифікатора (рис. 4.6).

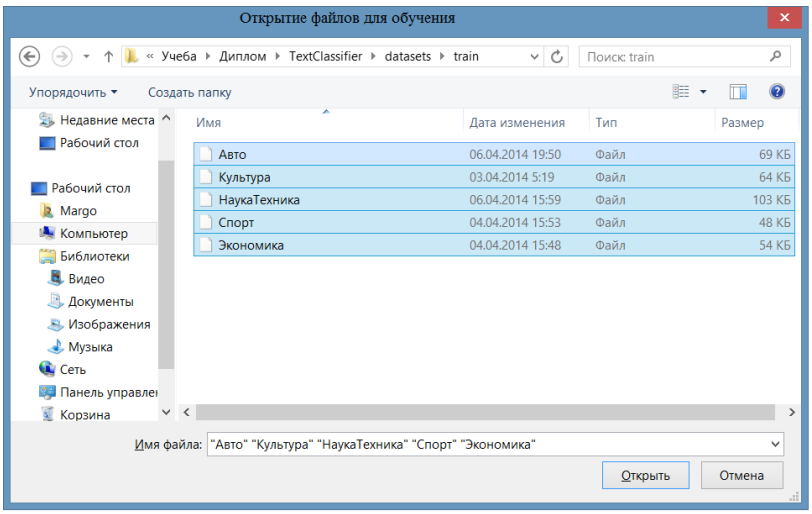

Рисунок 4.6 – Створюємо базу знань класифікатора

Після навчання класифікатора користувачеві показується загальна інформація про кількість навчальних текстів і ключових слів (рис.4.7).

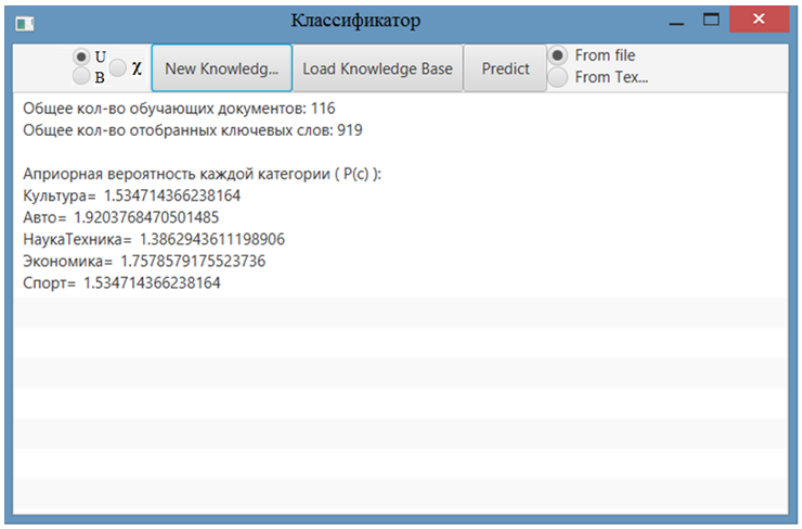

Рисунок 4.7 – Загальна інформація про базу знань після навчання

Після того, як користувач навчив класифікатор, він може перевіряти тематику тексту як з файлу, так і з введеного вручну у відведеному для цього полі тексту. Спливаюче вікно вибору файлу з'являється після натискання кнопки Predict, якщо перемикач в режимі From file (рис.4.8), а поле для введення тексту після перемикання «радіо-баттона» в режим From text (рис.4.9).

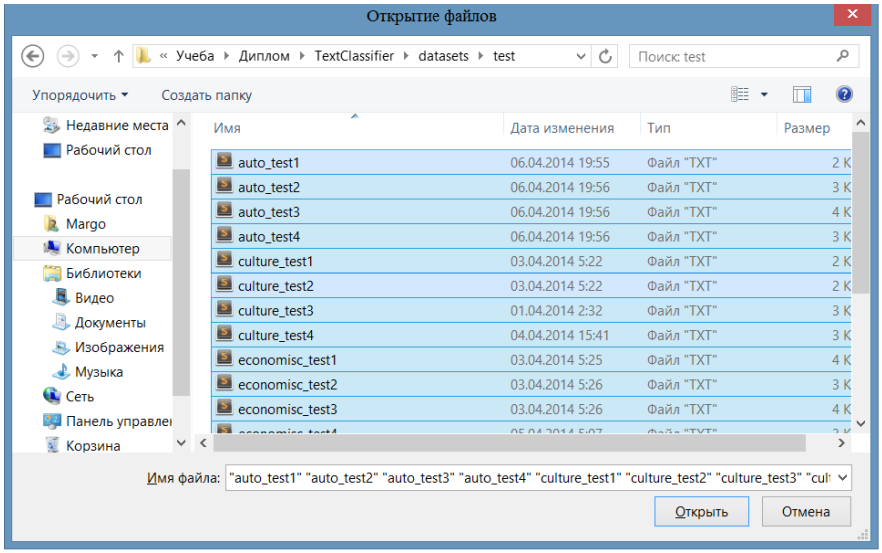

Рисунок 4.8 – Вибір необхідного файлу для визначення його тематики

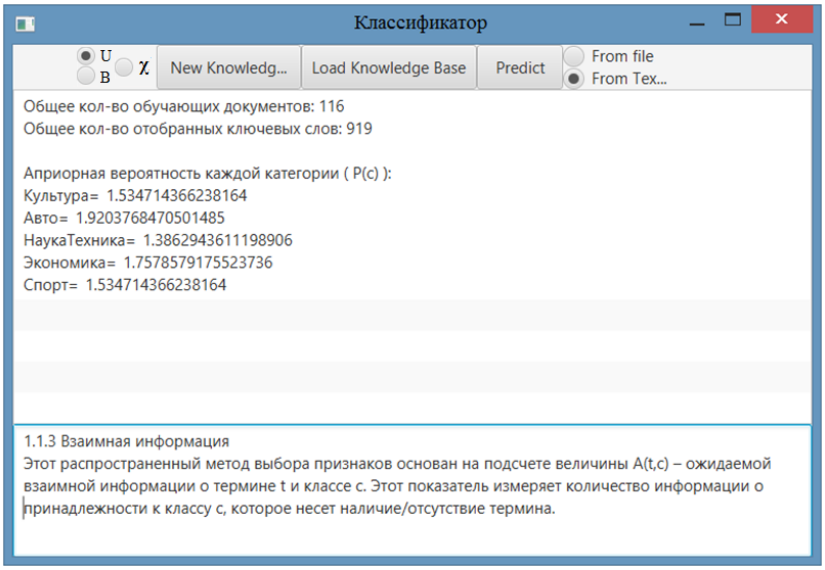

Рисунок 4.9 – Ввод тексту безпосередньо в текст-бокс

Потім, після відкриття необхідного файлу для класифікації, програма видає результат відношення тексту до відповідної категорії (рис.4.10).

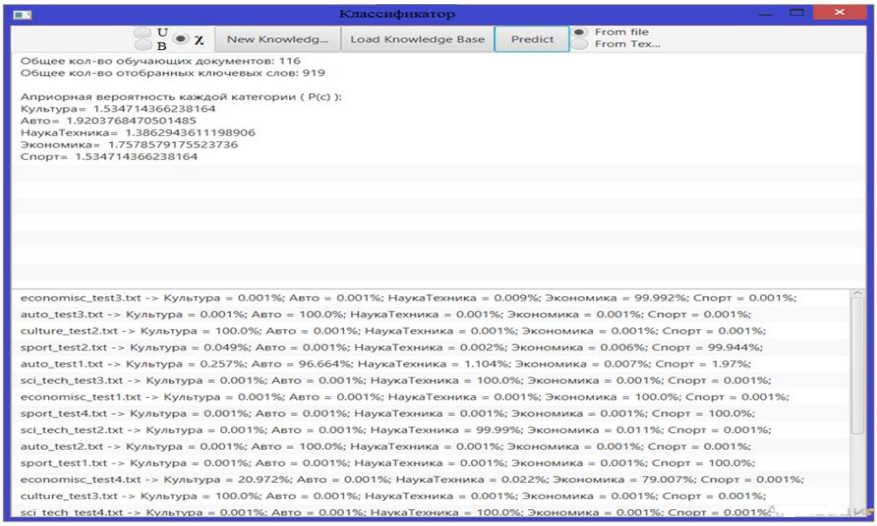

Рисунок 4.10 – Результат класифікації обраних файлів

Якщо текст був введений у поле для введення тексту (рис.4.11):

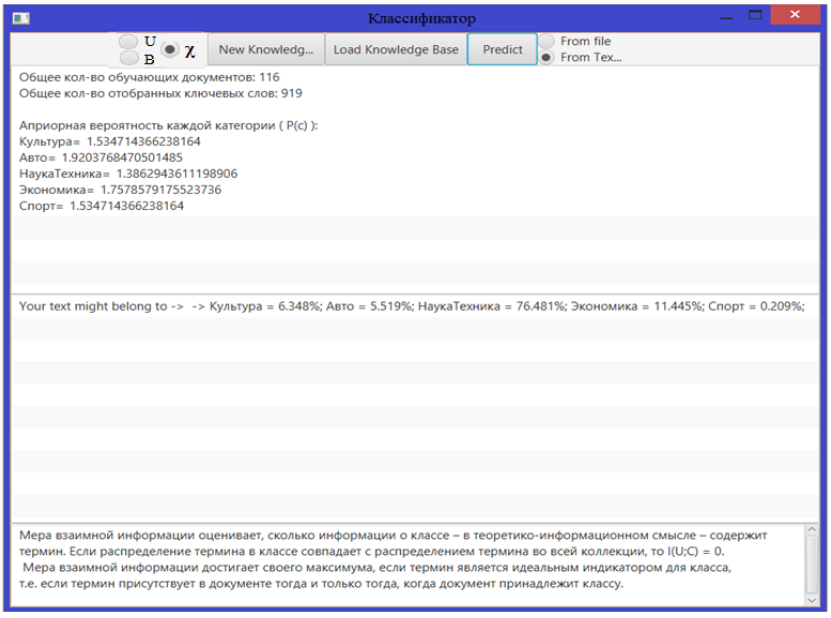

Рисунок 4.11 – Результат, отриманий після введення тексту вручну

Також користувачеві надається можливість завантажити вже наявну базу знань по натисканню кнопки LoadKnowledgeBase. Наприклад, якщо база знань уже була створена раніше (рис. 4.12).

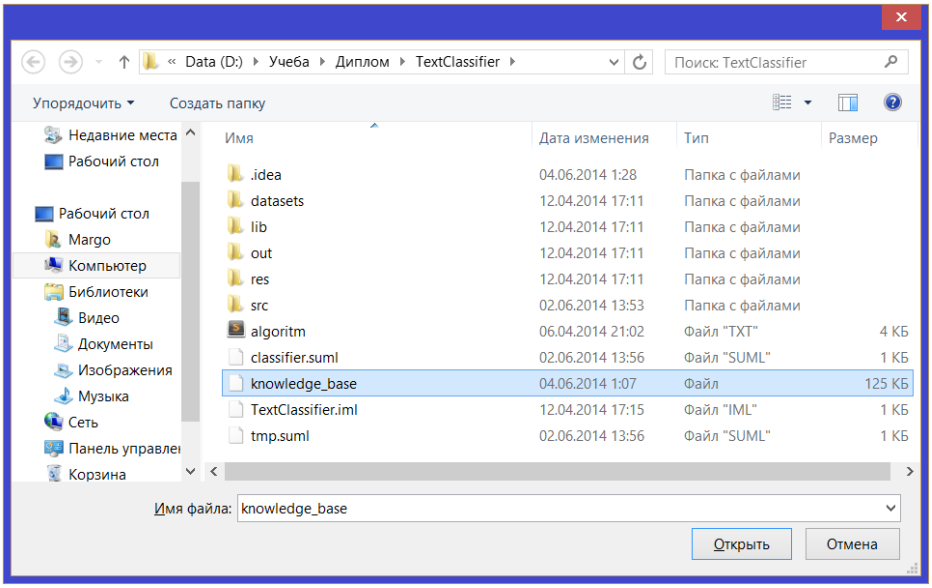

Рисунок 4.12 – Завантаження вже наявної бази знань

5 ТЕСТУВАННЯ І АНАЛІЗ РЕЗУЛЬТАТІВ РОБОТИ АЛГОРИТМУ КЛАСИФІКАЦІЇ

Для тестування алгоритмів класифікації текстів був використаний метод крос-валідації (або, по-іншому, перехресної перевірки). Процедура крос-валідації виконується наступним чином.

1) Фіксуються множини розбиття навчальної вибірки на власне навчальну та тестову.

2) Для кожного розбиття відбувається навчання алгоритму на навчальній підвибірці і тестування на тестовій.

3) Результатом крос-валідації алгоритму є середні значення оцінок ефективності для тестових підвибірок.

База знань для даного класифікатора була сформована на основі статей з lenta.ru і veddro.com, що відносяться до наступних тематик: «Культура», «Авто», «НаукаТехніка», «Економіка», «Спорт».

Для початкового тестування були взяті різні новинні статті, розміщені в текстових файлах. Використовувалися статті з ресурсів autonews.autoua.net, segodnya.ua, sciential.ru, elementy.ru і isport.ua. Результати тестування алгоритму Байєса, для якого вибір ознака визначається статистикою χ^2, представлені в табл. 5.1.

Таблиця 5.1 – Результати тестування додатка

Тематика	Кількість документів	Кількість правильно розпізнаних документів	Відсоток правильно розпізнаних	Мін. імовірність відношення до категорії	Макс. імовірність відношення до категорії
Культура	100	97	87,5%	0,002%	100%
Авто	100	100	100%	96,67%	100%
Наука Техніка	100	100	100%	99,87%	100%
Економіка	100	100	100%	66,54%	100%
Спорт	100	100	100%	98,76%	100%

Як можна побачити, для одного з текстових документів категорії «Культура» класифікатор дав невірний результат. Цей текст був узятий з новин-

ної статті, що відносилася до категорії «Культура», але несла в собі інформацію економічного характеру.

В якості метрик правильності класифікації текстів були обрані точність (precision) і повнота (recall). Точність в межах класу – це частка текстів, які дійсно належать даному класу, щодо всіх текстів, зарахованих класифікатором до цього класу. Повнота системи – відношення числа знайдених класифікатором текстів, що належать класу, до числа всіх текстів цього класу в тестовій колекції. Повнота і точність визначаються за наступними формулами:

$$Precision = \frac{TP}{TP + FP},$$
$$Recall = \frac{TP}{TP + FN}, \qquad (5.1)$$

де TP – істино-позитивне рішення;

TN – істино-негативне рішення;

FP – ложно-позитивне рішення;

FN – ложно-негативне рішення.

Для 1600 текстів за 4 тематиками було зроблено 4 підрозбиття (1200 текстів – навчальна вибірка, 400 – тестова). У кожній навчальній і тестовій підвибірці містилася однакова кількість текстів про Культуру, Авто, Економіку, Спорт. Для кожної групи розраховувалися оцінки ефективності, а потім вираховувались їх середні значення. Таким чином, вийшли усереднені оцінки ефективності (рис.5.1). Результати усереднених оцінок ефективності алгоритму Байєса для розглянутих ознак в табл. 5.2.

Таблиця 5.2 – Оцінка ефективності класифікатора Байєса

Векторна модель	Уніграми		Біграми	
ваги векторів	Точність, %	Повнота, %	Точність, %	Повнота, %
бінарні вектори	84,5	88,25	85,3	91,5
частотні вектори	85,2	87,4	86,2	89,4

Наївний байєсівський класифікатор показав дуже гарні результати, найбільш ефективною формою в плані точності виявилося поєднання біграм і частотної функції зважування – точність 86%.

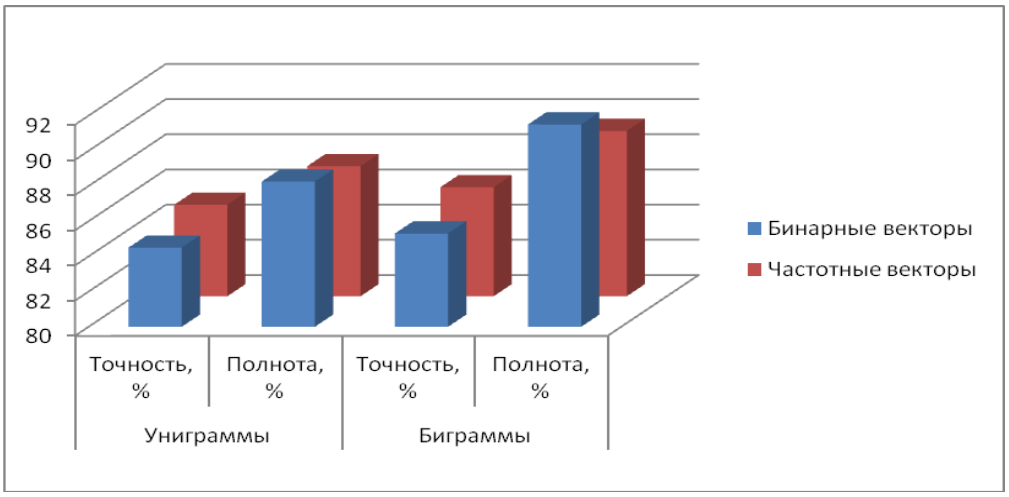

Рисунок 5.1– Оцінка ефективності класифікатора Байєса

Результати тестування Байєсівського класифікатора (рис.5.2).

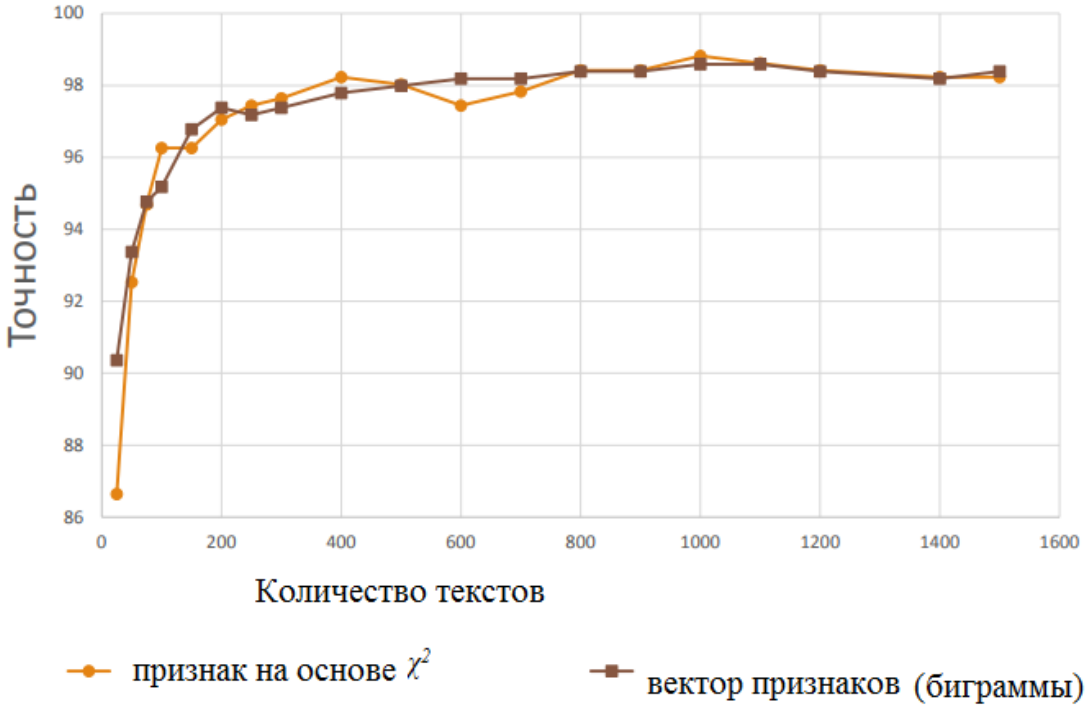

Рисунок 5.2– Порівняння точності алгоритму Байеса з різними ознаками

5.1 Порівняння з аналогічною системою

В якості тестових документів для класифікації були обрані окремо п'ять перших екзаменаційних питань по предмету «Захист інформації» і результат даної програми був порівняний із системою Extheme, заснованою на нейронних мережах (рис.5.3, 5.4).

Сервіс ExTheme потрібен для автоматичного визначення тематики будь-яких сайтів і текстів.

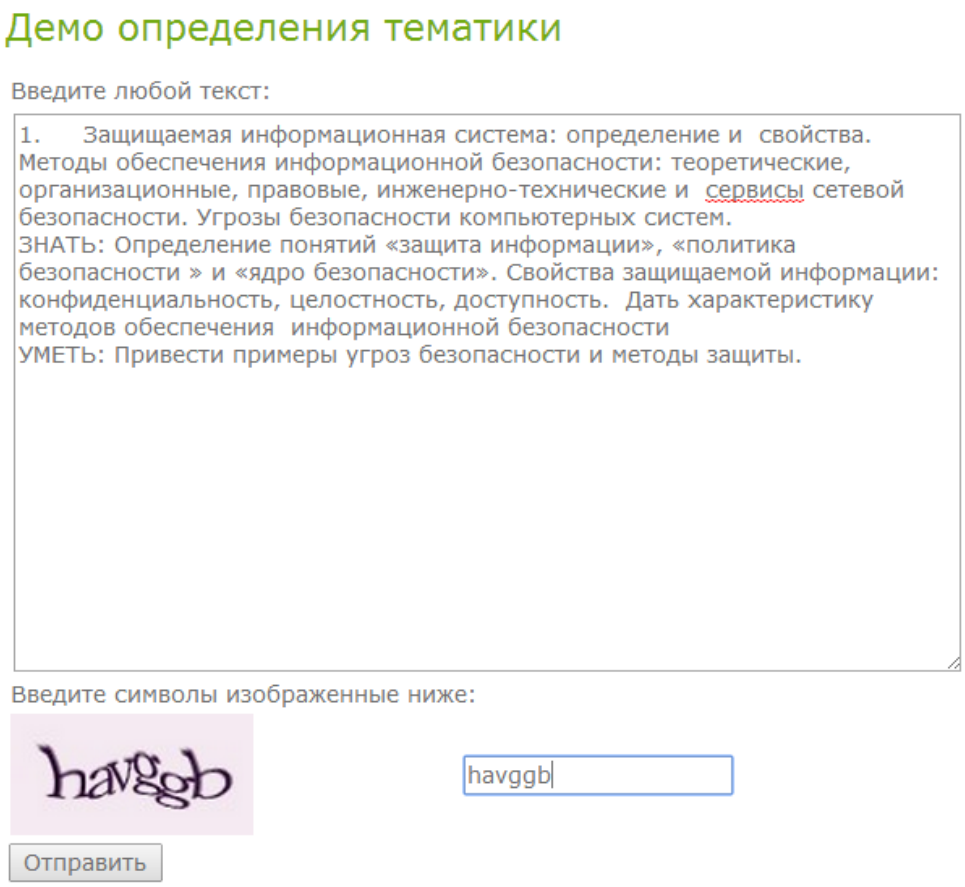

Рисунок 5.3 – Ввод першого питання в систему Extheme

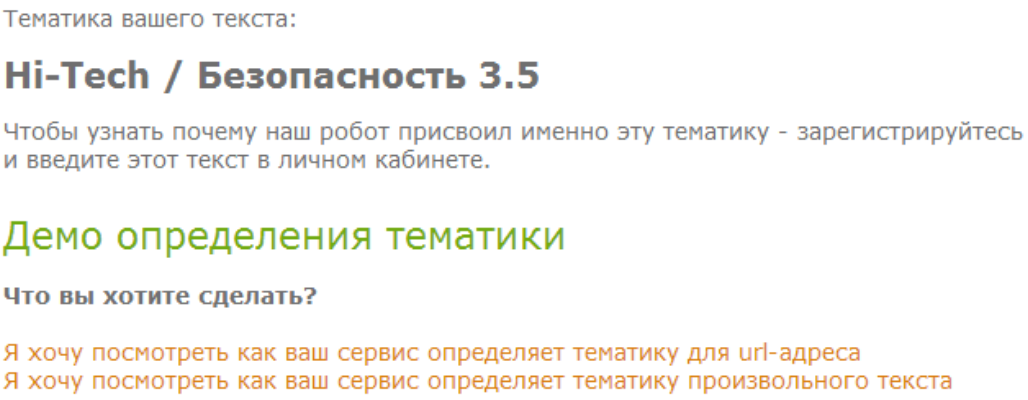

Рисунок 5.4 – Результат визначення тематики для першого питання

Проведемо тестування розробленого в дипломному проекті класифікатора. Для цього вводимо аналогічний текст і програма видає результат відношення тексту до відповідної категорії (рис.5.5).

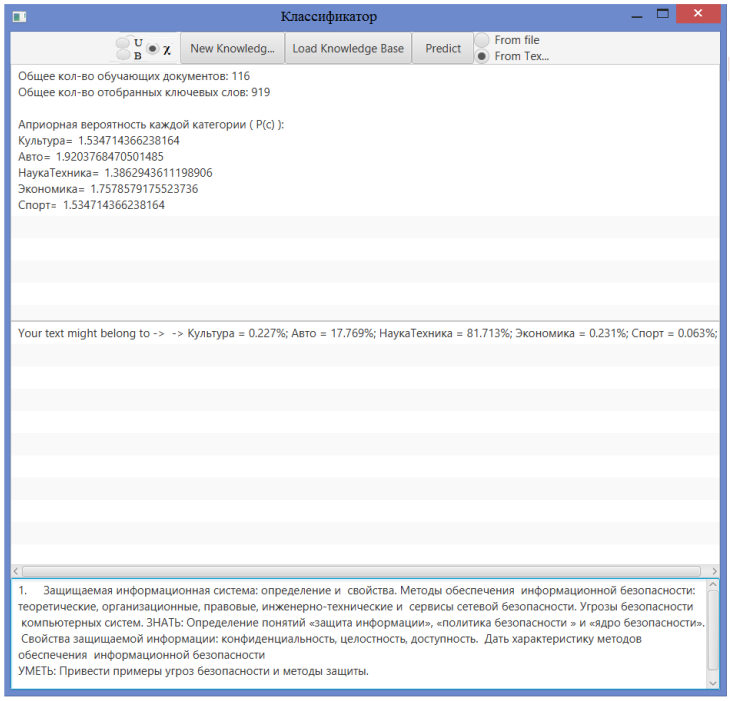

Рисунок 5.5 – Результати роботи розробленого в дипломній роботі класифікатора

Тепер у системі Extheme виконаємо класифікацію тематики тексту по другому питанню екзаменаційного квитка по предмету «Захист інформації». Результат роботи даної програми представлений на рис. 5.6, 5.7.

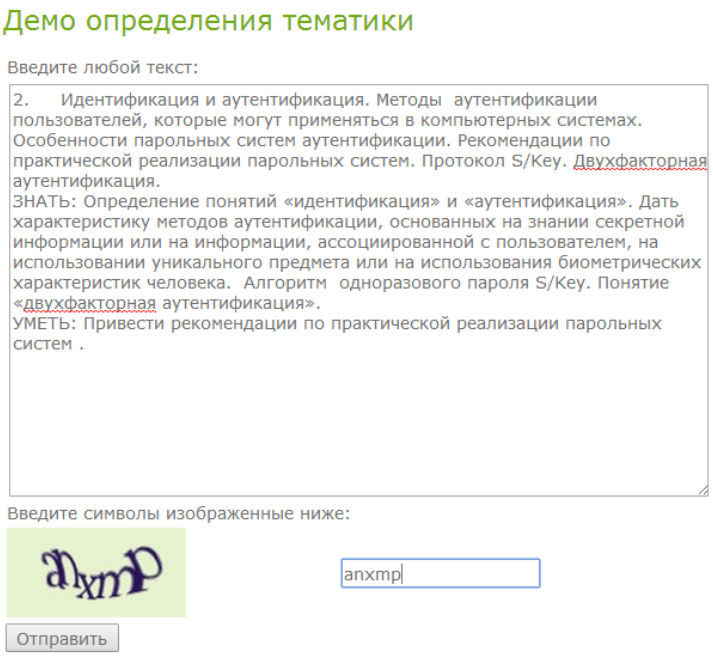

Рисунок 5.6 – Ввод другого питання в систему Extheme

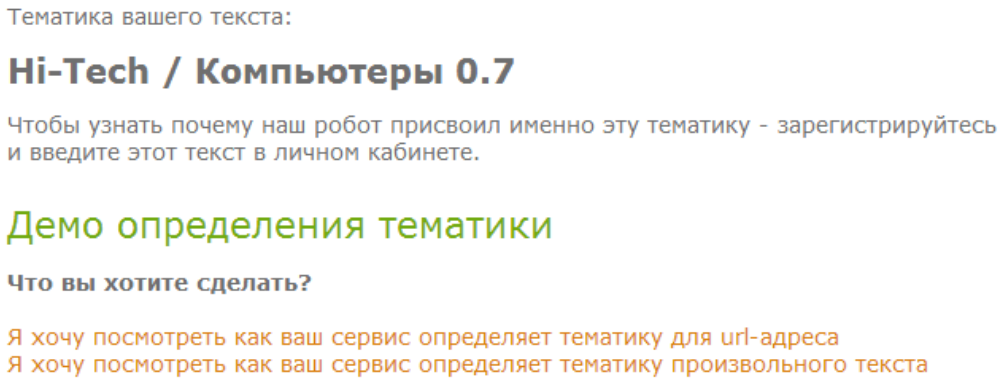

Рисунок 5.7 – Результат визначення тематики системою Extheme для другого питання

Результати тестування розробленого в дипломному проекті класифікатора на прикладі другого питання екзаменаційного квитка по предмету «Захист інформації» представлений на рис. 5.8.

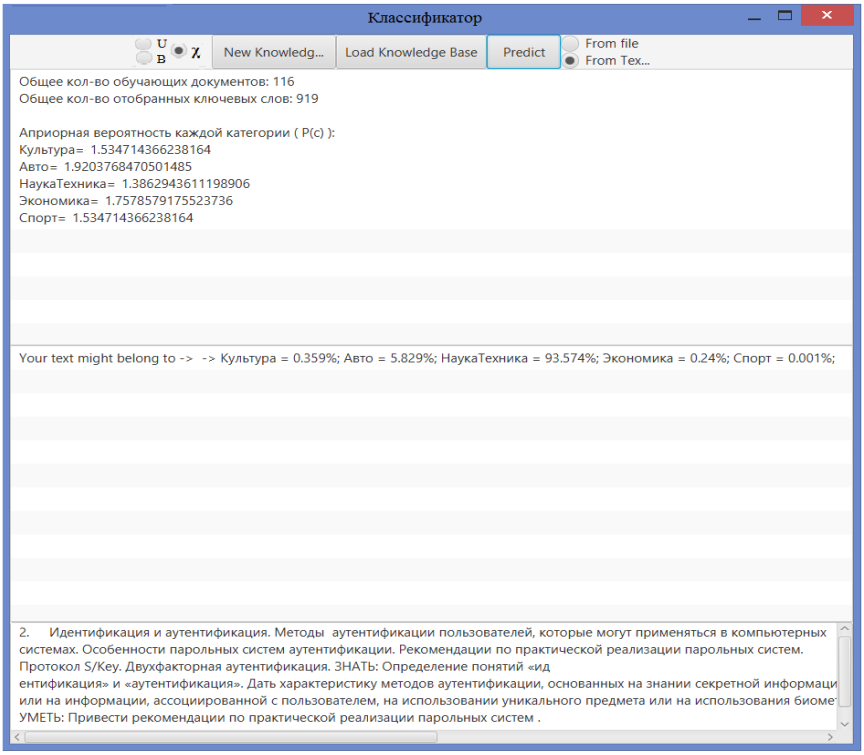

Рисунок 5.8 – Результати роботи розробленого класифікатора

Результати третього тестування по тематиці "Керування криптографічними ключами" системою Extheme і системою, розробленою в дипломному проекті представлені на рис. 5.9, 5.10.

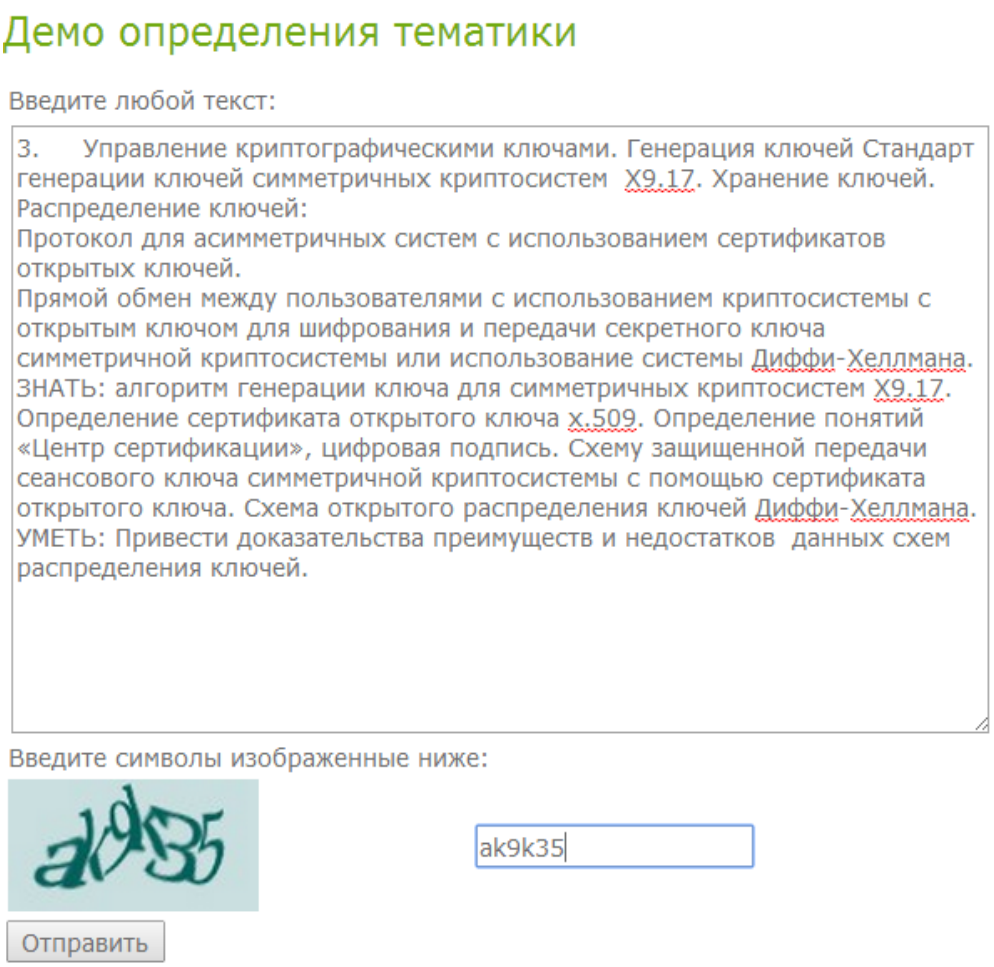

Рисунок 5.9 – Ввод третього питання в систему Extheme

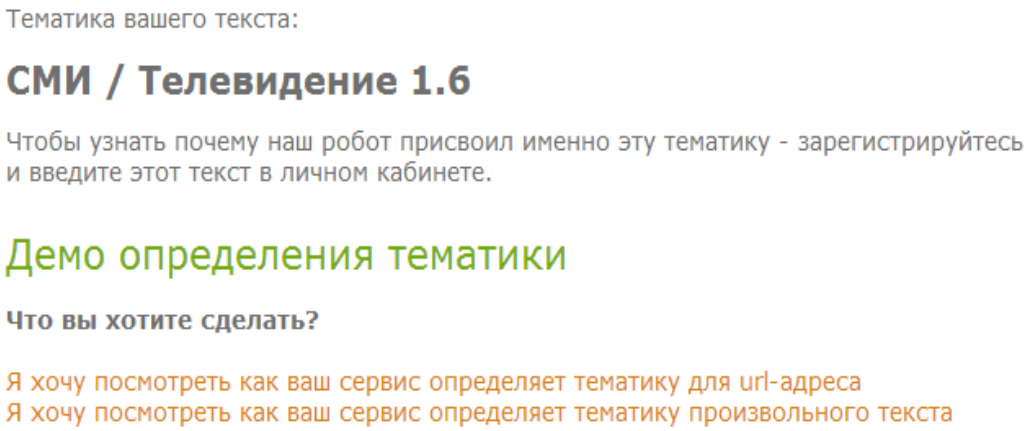

Рисунок 5.10 – Результат визначення тематики системою Extheme для третього питання

Результати передбачання розробленого в дипломній роботі класифікатора представлені на рис. 5.11.

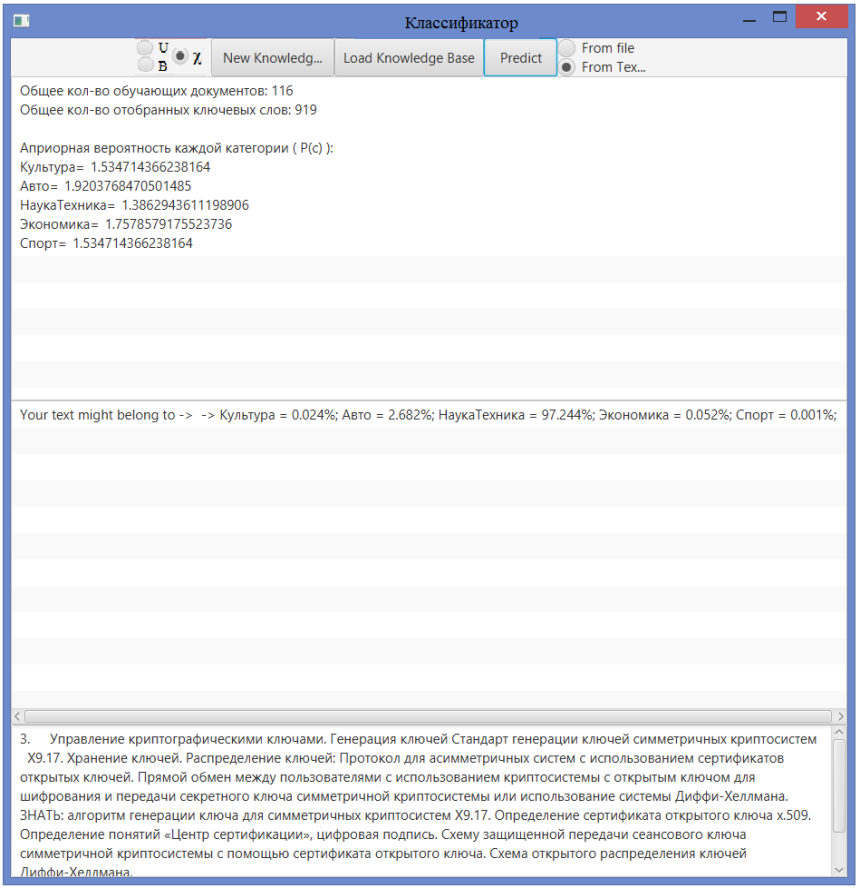

Рисунок 5.11 – Результати передбачання розробленого в дипломній роботі класифікатора

Питання, які були використані для тестування по предмету «Захист інформації».

1) Інформаційна система, що захищає: визначення і властивості. Методи забезпечення інформаційної безпеки: теоретичні, організаційні, правові, інженерно-технічні і сервіси мережевої безпеки. Погрози безпеки комп'ютерних систем.

ЗНАТИ: Визначення понять «захист інформації», «політика безпеки» і «ядро безпеки». Властивості захищаємої інформації, такі як: конфіденційність, цілісність, доступність. Дати характеристику методів забезпечення інформаційної безпеки

ВМІТИ: Привести приклади погроз безпеки і методи захисту.

2) Ідентифікація і аутентифікація. Методи аутентифікації користувачів, які можуть застосовуватися в комп'ютерних системах. Особливості парольних систем аутентифікації. Рекомендації із практичної реалізації парольних систем. Протокол S/Key. Двофакторна аутентифікація.

ЗНАТИ: Визначення понять «ідентифікація» і «аутентифікація». Дати характеристику методів аутентифікації, заснованих на знанні секретної інформації або на інформації, асоційованої з користувачем, на використанні унікального предмета або на використання біометричних характеристик людини. Алгоритм одноразового пароля S/Key. Поняття «двофакторна аутентифікація».

ВМІТИ: Привести рекомендації із практичної реалізації парольних систем .

3) Керування криптографічними ключами. Генерація ключів Стандарт генерації ключів симетричних криптосистем X9.17. Зберігання ключів. Розподіл ключів: Протокол для асиметричних систем з використанням сертифікатів відкритих ключів. Прямий обмін між користувачами з використанням криптосистеми з відкритим ключем для шифрування і передачі секретного ключа симетричної криптосистеми або використання системи Діффі-Хеллмана.

ЗНАТИ: алгоритм генерації ключа для симетричних криптосистем X9.17. Визначення сертифіката відкритого ключа x.509. Визначення понять «Центр сертифікації», цифровий підпис. Схему захищеної передачі сеансового ключа симетричної криптосистеми за допомогою сертифіката відкритого ключа. Схема відкритого розподілу ключів Діффі-Хеллмана.

ВМІТИ: Привести доказ переваг і недоліків даних схем розподілу ключів.

4) Протокол аутентифікації і розподілу ключів для симетричних криптосистем (система Kerberos).

ЗНАТИ: Архітектуру системи Kerberos: сервер аутентифікації (AS) і сервер видачі мандатів (TGT). Поняття область Kerberos. Схема роботи Kerberos v4,v5.

ВМІТИ: Привести приклад роботи системи Kerberos в ОС.

5) Сучасні симетричні криптосистеми. Режими застосування блокових шифрів: ECB,CBC,OFB,CFB.

ЗНАТИ: Структуру симетричних алгоритмів шифрування. Характеристики сучасних симетричних криптосистем: Blowfish, RC6, Serpent, Mars, AES(Rijndael), ДЕРЖСТАНДАРТ 28147. Схеми режимів симетричного шифрування.

ВМІТИ: Пояснити, який режим краще застосовувати при шифруванні інформації.

Електронний цифровий підпис. Схема створення і перевірки ЕЦП. Стандарти ЕЦП. Сліпий підпис.

ЗНАТИ: Функції ЕЦП. Схема створення і перевірки ЕЦП. Для чого використовується хеш-функції в ЕЦП. Алгоритми хеш-функції. Алгоритми ЕЦП. Стандарти ЕЦП у Європі, у Росії, в Україні. Поняття «маскуючі і демаскуючі функції». Алгоритм сліпого підпису.

ВМІТИ: Продемонструвати етапи створення ЕЦП і її перевірки.

Була отримана статистика, що наведена в табл. 5.3.

Таблиця 5.3 – Порівняння додатка із системою Extheme.ru

№ питання	Результат системи Extheme	Результат розробленої системи
1	Hi-Tech/Безпека 3.5	НаукаТехніка 81.71%
2	Hi-Tech/Комп'ютери 0.7	НаукаТехніка 93,57%
3	ЗМІ/Телебачення 1.6	НаукаТехніка 97,24%
4	Hi-Tech/Інтернет 1	НаукаТехніка 97,02%
5	Hi-Tech/Програми 0.6	НаукаТехніка 95,08%

Можна побачити, що для третього питання система Extheme видала результат, що зовсім не стосується реальної тематики введеного тексту – «ЗМІ / Телебачення 1.6», тоді як розроблений у данній дипломній роботі класифікатор визначив текст як «НаукаТехніка» більш, ніж на 97%. Можливо, це пов'язане з тим, що набір категорій у системі Extheme ширше, ніж у навчальній вибірці, що використовувалася при тестуванні додатка.

ВИСНОВКИ

У рамках данної роботи були розглянуті різні алгоритми класифікації текстів, реалізований класифікатор на основі наївного алгоритму Байєса з відбором ознак за алгоритмом хі-квадрат та вектором ознак.

Визначення тематики тексту – дуже перспективний і затребуваний науковий напрямок. На сучасний момент не існує універсальних алгоритмів, а надточні працюють довго. Як показав огляд аналогів, дуже мало доступних і зручних користувальницьких систем для розпізнавання тематики текстів. Тому для кожної конкретної задачі варто користуватися окремо підходящими сервисами. Дану галузь класифікації варто розвивати, тому що вона усе більш стає невід'ємною частиною для користувачів сучасного Інтернету.

У данній роботі наївний алгоритм Байєса був обраний тому, що для нього не потрібна велика навчальна вибірка, він швидко визначає коефіцієнти приналежності. Незважаючи на його простоту, відомо, що більш слабкі класифікатори мають перевагу в статистичній класифікації текстів, і на обмежених навчальних множинах більш слабкі моделі часто є кращими.

Був реалізований віконний додаток, що дозволяє навчити класифікатор наборам текстових документів, завантажити в програму вже створену базу знань, класифікувати текстові документи, а також довільно введений текст.

Було проведене тестування на різних документах, порівняння з аналогічними системами. На тестовій вибірці було отримане розпізнавання з точністю 97,5%. Був проведений порівняльний аналіз розпізнавання тематики тексту з популярною системою Extheme, що показав високу ефективність розробленої системи. Додаток має зручний інтерфейс, високу швидкодію.

Подальшими можливостями розвитку даного проекту є застосування додаткових алгоритмів класифікації для більш точної класифікації, класифікація веб-сайтів, застосування ієрархічної структури категорій, застосування технології Data Mining для самонавчання системи, статистичний аналіз біграм слів.

ПЕРЕЛІК ПОСИЛАНЬ

1. Сервис классификации тематики сайтов [Електронний ресурс] – Режим доступу: http://linkfeedator.ru/?task=tematika.
2. Сервис классификации тематики сайтов и фильтра url-страниц [Електронний ресурс] – Режим доступу: http://cfilter.ru/help.php
3. Система определения тематики сайтов и текстов [Електронний ресурс] – Режим доступу: http://sm.ashmanov.com/demo/
4. Система определения тематики сайтов и текстов [Електронний ресурс] – Режим доступу: http://extheme.ru/demo
5. Yang Y. A comparative study on feature selection in text categorization / Y.Yang, J.O.Pedersen – Waltham: Morgan Kaufmann Publishers, 1997. – 412-420 p.
6. Manning C.D. An introduction to information retrieval, first edition / C.D. Manning, P. Raghavan, H. Schütze – Cambridge: Cambridge University Press, 2009. – 544 p.
7. Liu B. Integrating classification and association rule mining / B. Liu, W. Hsu Y. Ma – New York: Lower Kent Ridge Road, 1998. – 80-86 p.
8. Cortes C. Support vector networks / C. Cortes, V. Vapnik – Boston: Kluwer Academic Publishers, 1995. – 297 p.
9. Wiener E. A neural network approach to topic spotting / E.Wiener, J.O. Pedersen, A.S. Weigend – Boulder: Xerox Palo Alto Research Center, 1995. – 216p.
10. Ивченко А.Ю. Практические аспекты задачи распознавания образов/ Ивченко А.Ю., Орлов Ю.Н. – Препринты ИПМ им. М.В. Келдыша. 2016. № 17. – 20с. [Електронний ресурс] – Режим доступу: http://library. keldysh.ru/ preprint.asp?id=2016-17
11. Арутюнов А.А. Статистические закономерности европейских языков и анализ рукописи Войнича / Арутюнов А.А., Борисов Л.А., Зенюк Д.А., Ивченко А.Ю., Кирина-Лилинская Е.П., Орлов Ю.Н., Осминин К.П., Федоров С.Л., Шилин С.А. – Препринты ИПМ им. М.В. Келдыша. 2016. № 52. –36 с. [Електронний ресурс] – Режим доступу: http://library. keldysh.ru/ preprint.asp? id=2016-52
12. Шевелев О.Г. Методы автоматической классификации текстов на естественном языке: Учебное пособие. Томск: ТМЛ-Пресс, 2007. – 144с.
13. Афифи А., Эйзен С. Статистический анализ: Подход с использованием ЭВМ. / Пер. с англ. – М.: Мир, 1982. – 488 с.

14. Автоматическая обработка текста [Електронний ресурс] – Режим доступу: http://www.aot.ru
15. Большакова Е.И. Автоматическая обработка текстов на естественном языке и компьютерная лингвистика/ Большакова Е.И. и др. – М.: МИЭМ, 2011. – 272 с.
16. Баженов Д. О задачах классификации [Електронний ресурс] – Режим доступу: http://bazhenov.me/blog
17. Гладкий А. В. Синтаксические структуры естественного языка в автоматизированных системах общения. М.: «Наука», 1985. – 143 с.

ДОДАТОК А

NaiveBayes.java

```
package com.datumbox.opensource.classifiers;
import com.datumbox.opensource.dataobjects.Document;
import com.datumbox.opensource.dataobjects.FeatureStats;
import com.datumbox.opensource.dataobjects.NaiveBayesKnowledgeBase;
import com.datumbox.opensource.features.FeatureExtraction;
import com.datumbox.opensource.features.TextTokenizer;
import java.util.ArrayList;
import java.util.HashMap;
import java.util.Iterator;
import java.util.List;
import java.util.Map;
public class NaiveBayes {
   private double chisquareCriticalValue = 10.83; //equivalent to pvalue 0.001. It is used
by feature selection algorithm

   private NaiveBayesKnowledgeBase knowledgeBase;

   public NaiveBayes(NaiveBayesKnowledgeBase knowledgeBase) {
      this.knowledgeBase = knowledgeBase;
   }

   public NaiveBayes() {
      this(null);
   }

   public NaiveBayesKnowledgeBase getKnowledgeBase() {
      return knowledgeBase;
   }

   public double getChisquareCriticalValue() {
      return chisquareCriticalValue;
   }

   public void setChisquareCriticalValue(double chisquareCriticalValue) {
      this.chisquareCriticalValue = chisquareCriticalValue;
   }

   private List<Document> preprocessDataset(Map<String, String[]> trainingDataset) {
      List<Document> dataset = new ArrayList<>();
```

```
        String category;
        String[] examples;

        Document doc;

        Iterator<Map.Entry<String, String[]>> it = trainingDataset.entrySet().iterator();

        //loop through all the categories and training examples
        while(it.hasNext()) {
            Map.Entry<String, String[]> entry = it.next();
            category = entry.getKey();
            examples = entry.getValue();

            for(int i=0;i<examples.length;++i) {
  //for each example in the category tokenize its text and convert it into a Document ob-
ject.
                doc = TextTokenizer.tokenize(examples[i]);
                doc.category = category;
                dataset.add(doc);

                //examples[i] = null; //try freeing some memory
            }

            //it.remove(); //try freeing some memory
        }

        return dataset;
    }

    private FeatureStats selectFeatures(List<Document> dataset) {
        FeatureExtraction featureExtractor = new FeatureExtraction();

        FeatureStats stats = featureExtractor.extractFeatureStats(dataset);
  Map<String, Double> selectedFeatures = featureExtractor.chisquare(stats,
chisquareCriticalValue);

        //clip from the stats all the features that are not selected
        Iterator<Map.Entry<String, Map<String, Integer>>> it =
stats.featureCategoryJointCount.entrySet().iterator();
        while(it.hasNext()) {
            String feature = it.next().getKey();

            if(selectedFeatures.containsKey(feature)==false) {
                //if the feature is not in the selectedFeatures list remove it
```

```
            it.remove();
        }
    }

    return stats;
}

/*
 * Trains a Naive Bayes classifier by using the Multinomial Model by passing
 * the trainingDataset and the prior probabilities.
 */
public void train(Map<String, String[]> trainingDataset, Map<String, Double>
categoryPriors) throws IllegalArgumentException {
    //preprocess the given dataset
    List<Document> dataset = preprocessDataset(trainingDataset);
    FeatureStats featureStats =  selectFeatures(dataset);
    //intiliaze the knowledgeBase of the classifier
    knowledgeBase = new NaiveBayesKnowledgeBase();
    knowledgeBase.n = featureStats.n; //number of observations
    knowledgeBase.d = featureStats.featureCategoryJointCount.size(); //number of fea-
tures
    //check is prior probabilities are given
    if(categoryPriors==null) {
        //if not estimate the priors from the sample
        knowledgeBase.c = featureStats.categoryCounts.size(); //number of cateogries
        knowledgeBase.logPriors = new HashMap<>();

        String category;
        int count;
        for(Map.Entry<String, Integer> entry : featureStats.categoryCounts.entrySet()) {
            category = entry.getKey();
            count = entry.getValue();

            knowledgeBase.logPriors.put(category,
Math.log((double)count/knowledgeBase.n));
        }
    }
    else {
        //if they are provided then use the given priors
        knowledgeBase.c = categoryPriors.size();

        //make sure that the given priors are valid
        if(knowledgeBase.c!=featureStats.categoryCounts.size()) {
            throw new IllegalArgumentException("Invalid priors Array: Make sure you
pass a prior probability for every supported category.");
```

```
            }

            String category;
            Double priorProbability;
            for(Map.Entry<String, Double> entry : categoryPriors.entrySet()) {
               category = entry.getKey();
               priorProbability = entry.getValue();
               if(priorProbability==null) {
 throw new IllegalArgumentException("Invalid priors Array: Make sure you pass a prior probability for every supported category.");
               }
               else if(priorProbability<0 || priorProbability>1) {
                  throw new IllegalArgumentException("Invalid priors Array: Prior probabilities should be between 0 and 1.");
               }

               knowledgeBase.logPriors.put(category, Math.log(priorProbability));
            }
         }
         Map<String, Double> featureOccurrencesInCategory = new HashMap<>();

         Integer occurrences;
         Double featureOccSum;
         for(String category : knowledgeBase.logPriors.keySet()) {
            featureOccSum = 0.0;
            for(Map<String, Integer> categoryListOccurrences :
featureStats.featureCategoryJointCount.values()) {
               occurrences=categoryListOccurrences.get(category);
               if(occurrences!=null) {
                  featureOccSum+=occurrences;
               }
            }
            featureOccurrencesInCategory.put(category, featureOccSum);
         }

         //estimate log likelihoods
         String feature;
         Integer count;
         Map<String, Integer> featureCategoryCounts;
         double logLikelihood;
         for(String category : knowledgeBase.logPriors.keySet()) {
            for(Map.Entry<String, Map<String, Integer>> entry :
featureStats.featureCategoryJointCount.entrySet()) {
               feature = entry.getKey();
               featureCategoryCounts = entry.getValue();
```

```
            count = featureCategoryCounts.get(category);
            if(count==null) {
               count = 0;
            }

            logLikelihood =
Math.log((count+1.0)/(featureOccurrencesInCategory.get(category)+knowledgeBase.d));
            if(knowledgeBase.logLikelihoods.containsKey(feature)==false) {
               knowledgeBase.logLikelihoods.put(feature, new HashMap<String, Dou-
ble>());
            }
            knowledgeBase.logLikelihoods.get(feature).put(category, logLikelihood);
         }
      }
      featureOccurrencesInCategory=null;
   }

   public void train(Map<String, String[]> trainingDataset) {
      train(trainingDataset, null);
   }

   public String predict(String text) throws IllegalArgumentException {
      if(knowledgeBase == null) {
         throw new IllegalArgumentException("Knowledge Bases missing: Make sure
you train first a classifier before you use it.");
      }

      //Tokenizes the text and creates a new document
      Document doc = TextTokenizer.tokenize(text);


      String category;
      String feature;
      Integer occurrences;
      Double logprob;

      String maxScoreCategory = null;
      Double maxScore=Double.NEGATIVE_INFINITY;

      //Map<String, Double> predictionScores = new HashMap<>();
      for(Map.Entry<String, Double> entry1 : knowledgeBase.logPriors.entrySet()) {
         category = entry1.getKey();
         logprob = entry1.getValue(); //intialize the scores with the priors
```

```
            //foreach feature of the document
            for(Map.Entry<String, Integer> entry2 : doc.tokens.entrySet()) {
               feature = entry2.getKey();

               if(!knowledgeBase.logLikelihoods.containsKey(feature)) {
                  continue; //if the feature does not exist in the knowledge base skip it
               }

               occurrences = entry2.getValue(); //get its occurrences in text

               logprob += occurrenc-
es*knowledgeBase.logLikelihoods.get(feature).get(category); //multiply loglikelihood
score with occurrences
            }
            //predictionScores.put(category, logprob);

            if(logprob>maxScore) {
               maxScore=logprob;
               maxScoreCategory=category;
            }
         }

         return maxScoreCategory; //return the category with heighest score
      }
}
```