МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Тернопільський національний економічний університет
Факультет комп'ютерних інформаційних технологій
Кафедра комп'ютерних наук

ПОВОРОЗНИК Володимир Степанович

Математичне та програмне забезпечення задачі розсилання електронної кореспонденції/ Mathematical base and software for task of sending electronic mails

спеціальність: 8.05010301 - Програмне забезпечення систем
магістерська програма - Програмне забезпечення систем

Магістерська робота

Виконав студент групи ПЗСм-21
В. С. Поворозник
_____

Науковий керівник:
к.т.н., доцент ПІГОВСЬКИЙ Ю.Р.
_____

Магістерську роботу допущено
до захисту:

"___" _____ 20___ р.

Завідувач кафедри
_____ **А. В. Пукас**

ТЕРНОПІЛЬ - 2016

РЕЗЮМЕ

Дипломна робота містить 96 сторінок, 16 таблиць, 25 рисунків, список використаних джерел із 21 найменуваня, 5 додатків.

Метою дипломної роботи є розробка математичного і програмного забезпечення для задачі розсилання електронної кореспонденції, оптимізація процесу розсилання електронної кореспонденції.

Об'єктом дослідження є процес оптимізації розсилання електронної кореспонденції за допомогою кластерного аналізу.

Предметом дослідження є математичне та програмне забезпечення задачі розсилання електронної кореспонденції для покращення ведення бізнесу.

Одержані висновки та їх новизна полягають у тому, що удосконалено інформаційну технологію розсилання електронної кореспонденції на основі кластерного аналізу засобами програмно-лінгвістичного аналізу, що сприяло підвищенню ефективності здійснення популяризації товарів чи бізнесу.

Ключові слова: застосунок, Data Mining, кластеризація, API, Text Mining, інтерфейс, технології C#, .NET, Windows Forms, dll, мова, стемінг, стоп слова , k-середніх, TF-IDF, електронна кореспонденція, маркетинг.

SUMMARY

Thesis contains 96 pages, 16 tables, 25 figures, list of sources with 21 titles, 5 additions.

The aim of the thesis is to develop mathematical and programming software for the task of sending e-mails, optimization of e-mails sending process.

Object of research is is to optimize the e-mails sending process using cluster analysis method.

The subject of research is mathematical and programming software for the task of sending e-mail to improve business.

The resulting conclusions and innovations are in improvement of information technology for sending e-mails based on cluster analysis by software and linguistic analysis instruments, thereby increasing the efficiency of popularizing products or business.

Keywords: application, Data Mining, clustering, API, Text Mining, interface, C#, .NET, Windows Forms, dll, language, stemming, stop words, k-means, TF-IDF, e-mails, marketing.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

Кластеризація	Кластерний аналіз - задача розбиття заданої вибірки об'єктів (ситуацій) на підмножини, що називаються кластерами, так, щоб кожен кластер складався з схожих об'єктів, а об'єкти різних кластерів істотно відрізнялися.
Маркетинг	діяльність, спрямована на досягнення цілей підприємств, установ, організацій шляхом формування попиту та максимального задоволення потреб споживачів.
Кластер	група однакових або подібних елементів, зібраних разом або близько один до одного
Додаток	користувацька комп'ютерна програма, що дає змогу вирішувати конкретні прикладні задачі користувача.
tf–idf	частота терміну – зворотня частота терміну у документах
ПС	програмна система
API	інтерфейс прикладного програмування
ОС	операційна система
ПЗ	програмне забезпечення
ПК	персональний комп’ютер
Прикладне ПЗ	ПЗ, що обслуговує користувача
Data Mining	Отримання да́них - виявлення прихованих закономірностей або взаємозв'язків між змінними у великих масивах необроблених даних. Як правило поділяється на задачі класифікації, моделювання та прогнозування.
Стемінг	процес скорочення слова до основи шляхом відкидання допоміжних частин, таких як закінчення чи суфікс
ПП	Програмний продукт
Text Mining	Інтелектуальний аналіз тексту, добування текстових даних

ВСТУП

Актуальність теми.

Автоматизація і комп’ютеризація невпинно захоплюють усе нові й нові галузі людської діяльності. Доступ до інтернету є уже навіть у найвіддаленіших куточках нашої планети, майже кожен сучасний гаджет як правило тим чи іншим чином з’єднується з мережею. Скоро у людське життя прийде інтернет речей, який ще більше розповсюдить і збільшить вплив інтернет павутини на життя людини.

І тому для популяризації свого бізнесу чи розшерення його рамок, ніхто не взмозі нехтувати комп’ютерними засобами і підходами. Одним з засобів популяризації свого бізнесу, товару або групи товарів є розсилання електронної кореспонденції. Але оскільки розсилання електронної кореспонденції не є безплатним процесом і остаточно ефективним методом, то постає питання в оптимізації цього процесу, наприклад, через вибір певним чином отримувачів цієї кореспонденції, для того, щоб вона була надіслана тільки тим, кому це насправді потрібно і цікаво, що зекономить кошти і ефективніше дозволить розповсюджувати інформацію, не вдаючись до грубого спаму користувачів.

Для оптимізації цього процесу нам потрібно знайти зв'язок між кориспонденцію для розсилання і тим, що ми знаємо про клієнтів, наприклад їхні інтереси. Для цього нам може допомогти Data Mining, що дозволяє знаходити приховані закономірності і зв'язки у величезних об'ємах несортовах даних у мережі чи сховищах підприємства.

Data Mining дає можливість автоматичного аналізу знайдених даних завдяки застосуванню різних методів математичної статистики, штучних нейронних мереж, теорії нечітких множин або генетичних алгоритмів. Метою такого роду аналізу є виявлення правил і закономірностей у даних для подальшого застосування цих закономірностей з метою покращення або автоматизації бізнесу.

Зокрема, коли ми маємо справу з текстовими даними, наприклад, електронною кореспонденцією, аналіз яких на відміну від цифрових, є набагато складнішою задачею, в силу того, що у кожної мови є свої особливості, і передбачити усі можливі варіанти майже неможливо без людського втручання, для цієї задачі існує один з напрямів Data Mining – Text Mining, який працює з текстовими масивами, колекціями документів і отримує з них інформацію базуючись на застосуванні ефективних методів машинного навчання та обробки природної мови. Для аналізу отриманої інформації використовуються ті ж підходи, що й Data Mining.

Сучасні програмні додатки для розсилання електронної кореспонденції зосереджуються на створенні редакторів для редагування вмісту електронної кореспонденції, простого функціоналу на зразок – календаря розсилки і засобів, власне розсилання. Деякі ПП надають можливість збору статистики, але жодна не аналізує дані для розсилки і вся ноша вирішення цієї задачі лягає на людські плечі, тому пропонується створити модульну систему для розсилання електронної кореспонденції, яка б складалась з наступних модулів: модуль для аналізу і кластеризації текстових документів, максимально пристосований для роботи з текстовими даними і максимально гнучкий і розширюваний, модуль для розсилання, графічний модуль для занесення даних. Зважаючи на відсутність рішень з нахилом на оптимізацію розсилання електронної кореспонденції, проблема створення математичного та програмного забезпечення для задачі розсилання електронної кореспонденції є ***актуальною***.

Розроблюваний засіб повинен бути у вигляді модульної і гнучкої системи, а також бути легким у освоєнні і застосуванні. Тому для реалізації було обрано мову програмування C#, що відзначається зручною можливістю розробки гнучких і розширюваних модульних систем.

Зв'язок роботи з науковими програмами, планами, темами. Напрямок виконаних досліджень безпосередньо пов'язаний з науково-дослідним

напрямком кафедри комп'ютерних наук Тернопільського національного економічного університету.

Мета і завдання досліджень.

Метою дипломної роботи є розробка математичного і програмного забезпечення для задачі розсилання електронної кореспонденції, оптимізація процесу розсилання електронної кореспонденції. Досягнення даної мети виконується за рахунок виконання наступних завдань:

- аналіз відомого ПЗ для розсилання електронної кореспонденції,
- аналіз проблеми автоматизації і оптимізації задачі розсилання електронної кореспонденції,
- попередній аналіз основних функціональних та нефункціональних особливостей, які слід реалізувати у ПЗ розсилання електронної кореспонденції,
- розробити і дослідити базові методи та алгоритми аналізу електронної кореспонденції,
- за допомогою підходів об'єктно-орієнтованого проектування на основі мови UML, спроектувати структуру програмного забезпечення для автоматизованого контекстного розсилання електронної кореспонденції,
- реалізувати системну частину спроектованого на попередньому кроці програмного забезпечення, що аналізує дану електронну кореспонденцію і розсилає її,
- реалізувати прикладну частину спроектованого програмного забезпечення,
- провести оцінку продуктивності і надійності розробленого застосунку, ілюструючи ці оцінки кількісними даними та графіками.

Об'єкт дослідження: процес оптимізації розсилання електронної кореспонденції за допомогою кластерного аналізу.

Предмет дослідження: математичне та програмне забезпечення задачі розсилання електронної кореспонденції для покращення ведення бізнесу.

Методи дослідження. Теоретичною основою магістерського дослідження виступає алгоритм кластеризації к-середніх та елементарні поняття Data Mining і Text Mining.

Наукова новизна отриманих результатів полягають у тому, що удосконалено інформаційну технологію розсилання електронної кореспонденції за допомогою кластерного аналізу, засобами програмно-лінгвістичного аналізу, що сприяло підвищенню ефективності здійснення популяризації товарів чи бізнесу.

Практичне значення отриманих у роботі результатів полягає у можливості автоматичного аналізу електронної кореспонденції перед її надсиланням.

Особистий внесок здобувача. Основні результати магістерської роботи отримані автором самостійно, на основі власних ідей та розробок.

Апробація результатів. Основні положення магістерського дослідження апробовані на VI Всеукраїнській школі-семінарі молодих вчених і студентів «Сучасні комп'ютерній інформаційні технології», що проходила 20-21 травня 2016 року у м. Тернополі на базі Тернопільського національного економічного університету, копія публікації подана у додатку Д.

Публікації. Поворозник В.С. Оптимізація задачі розсилання електронної кореспонденції. / Поворозник В.С. // Матеріали VI Всеукраїнської школи-семінару молодих вчених і студентів «Сучасні комп'ютерні інформаційні технології», АСІТ'2016 – Тернопіль: ТНЕУ, 2016. − с. 145-146.

Структура та об'єм роботи. Магістерська робота складається з трьох розділів, загальним обсягом 96 сторінок, де розміщено 16 таблиць, 25 рисунків, 5 додатків та 21 джерело в переліку посилань.

РОЗДІЛ 1
АНАЛІЗ МАТЕМАТИЧНОГО ТА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ДЛЯ РОЗСИЛАННЯ ЕЛЕКТРОННОЇ КОРИСПОНДЕНЦІЇ

1.1. Огляд існуючого ПЗ для розсилання електронної кореспонденції

Із стрімким розвитком інтернету і технологій пов'язаних з ним, участі комп'ютерів і мобільних пристроїв у практично будь-якому аспекті життя людини та дедалі більшому поширенню можливості популяризації свого бізнесу через ці пристрої – проблема оптимізації електронного маркетингу стає дедалі актуальнішою.

Електронна пошта – є секретною зброєю Internet маркетингу. Кожне відправлене компанією електронне повідомлення є елементом маркетингу, і можна змусити цю потужну силу працювати на вас чи з її допомогою заявити про свою діяльність.

Хоча Web – сайт є найважливішим елементом маркетингової діяльності у мережі, Internet також надає у користування і багато інших маркетингових інструментів, у яких використовується звичайний текст. До них належать електронна пошта, автоматичні списки для розсилки і групи для обговорення, чи новинні групи. І все-таки розсилання електронної пошти, залишається одним із найважливіших інструментів.

При такому швидкому розвитку технологій є надзвичайно важливим ефективно використовувати їх для отримання максимальної вигоди, а для того, щоб цього досягти – були створені та продовжують створюватись програмні системи, програми, ціллю яких є ведення маркетингу через розсилання електронної кореспонденції.

Можна виділити ряд основних сфер застосування та впровадження систем для розсилання електронної кореспонденції:

- популяризація товарів;
- розсилання важливої внутрішньої інформації;
- проведення кампаній;

- інформаційна розсилка.

Оскільки додатки для розсилання електронної кореспонденції є надзвичайно популярними і їх існує величезна кількість, тому розглянемо найпопулярніші за оцінками користувачів [1] [2] [3], а також визначимо їх плюси і мінуси [4]:

MailChimp

Більше 12 мільйонів чоловік з різних підприємств по всьому світу використовують Mailchimp. Властивості інтеграції дозволяють відправляти поштою маркетингові автоматизовані повідомлення, адресні кампанії. Також ПЗ дає можливість формувати докладні звіти, що допоможуть покращувати характеристики рекламних кампаній з часом.[5]

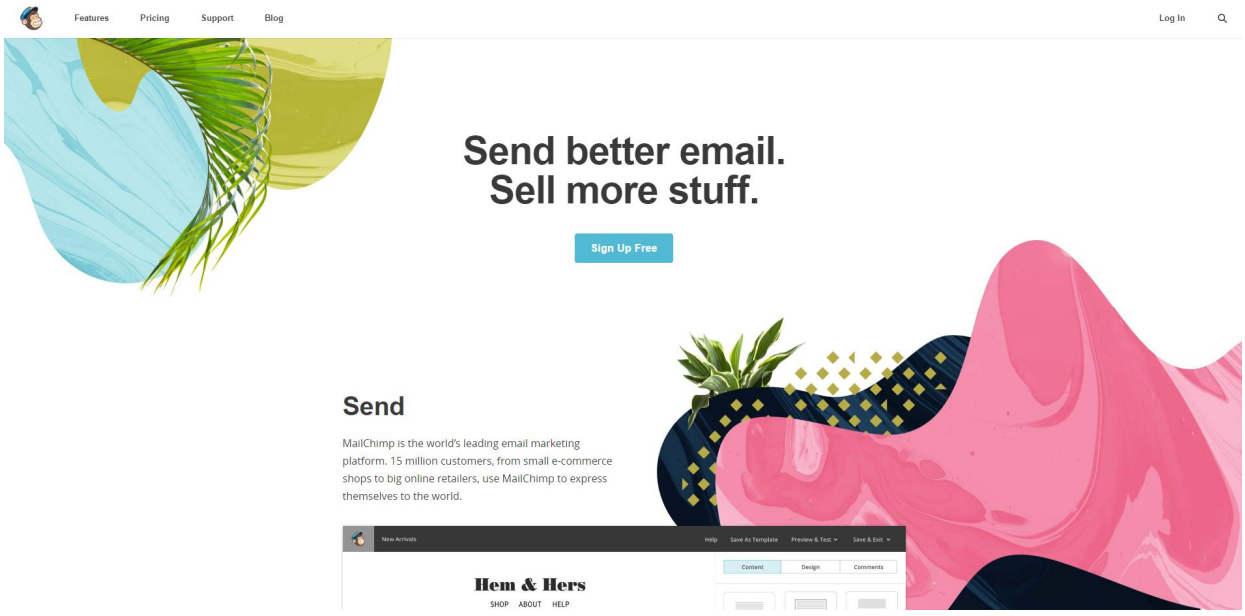

Рисунок 1.1 – Веб сайт MailChimp

MailChimp пропонують простий і приємний інтерфейс і є багато можливостей для початківців і досвідченіших користувачів. MailChimp також має Pro план для корпоративного рівня, який має деякі складні розширені функції, але прийдеться платити за ці функції $199 на місяць.

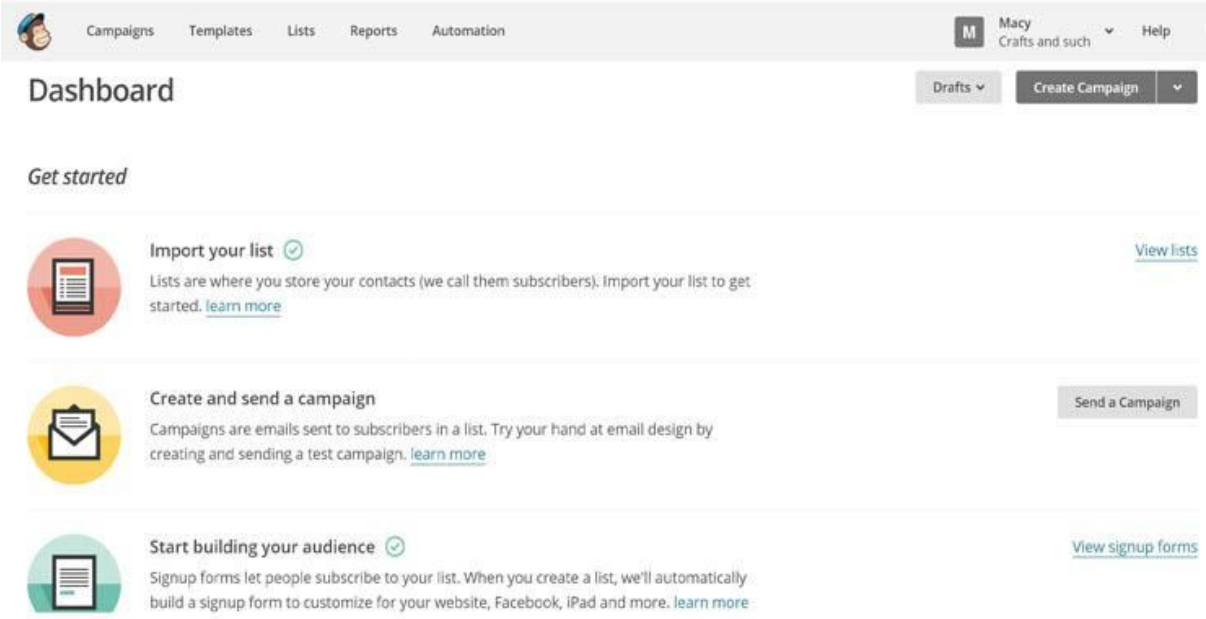

Рисунок 1.2 – Головне меню MailChimp

MailChimp має найкращий безкоштовний тарифний план серед інших постачальників послуг електронної розсилки для маркетингу. Поряд з довгим списком функцій, включаючи хостинг зображень, Google Analytics, а також інтеграції з сотнями додатків, MailChimp є відмінним вибором для початківців або тих, хто мають обмежене фінансування.

Ключові особливості

MailChimp має безліч функцій, які відносяться до розсилання електронної пошти в тому числі, нові функції автоматизації і можливість створення облікового запису на декілька користувачів. MailChimp сканер купонів також став нарешті доступний, ця функція дозволяє власникам магазинів відправити QR код купони своїм абонентам. Також MailChimp має такі можливості [5]:

- Інтеграція Електронної комерції (Shopify, Magento, WooCommerce, BigCommerce);
- Drag & drop дизайнер;

- можливості організації робочого процесу спільної роботи, такі як дозвіл декількох облікових записів користувачів і додавання коментарів в редакторі;
- автоматизація маркетингу;
- Розширені функції звітності;
- Інтеграція Google Analytics ;
- Інтеграція з Twitter і Facebook;
- Документоване API.

Плюси

- Вільний план для списків з 2000 або менше абонентів і до 12 000 листів в місяць;
- HTML шаблони Mailchimp прості у використанні і ви можете використовувати їх візуальний редактор, щоб налаштувати свої власні проекти;
- Тести усіх великих інтернет-провайдерів, щоб побачити, чи спам-фільтри спрацьовують;
- Відслідковування підписок;
- Сотні інтеграцій, в тому числі з Google Analytics;
- Підтримка мобільних платформ;
- Безпосереднє управління списком контактів і легкий імпорт контактів з допомогою CVS файлу;
- Чат.
- Веб і мобільна версія

Мінуси

- Імпорт контактних списків міг би бути простішим;
- Відсутність імпорту шаблонів з існуючих веб-сайтів;

- Багато розширених функції доступні тільки з Pro плану (починаючи з 199$ в місяць);
- Повільна швидкодія, особливо коли сайт під нагрузкою;
- Сегментація обмежена у безкоштовному плані і не автоматизована;
- Відсутність підказок;
- Немає підтримки по телефону, а підтримка через чат часто відсутня;
- Відсутність контекстної розсилки, що слід вважати основним недоліком у контексті нашого дослідження.

Campaign Monitor

Campaign Monitor це програмне забезпечення для розсилання електронної кореспонденції, створене спеціально з врахуванням вимог графічних дизайнерів, легкий і простий Drag & drop дизайн спеціально для початківців. Цей інтерфейс дуже приємний для очей і дозволяє керувати декількома обліковими записами. Campaign Monitor має більшість основних функцій для запуску успішної маркетингової кампанії через електронну пошту.

Campaign Monitor — це онлайн сервіс інтернет-маркетингу, запущений в 2004 році компанією, що знаходиться у Сіднеї, Австралія. Програмне забезпечення було створено, коли Бен Річардсон і Девід Грайнер були на останньому році навчання в своєму коледжі і багато осіб запрошували їх для управління веб-сайтами і кампаніями по розсилці електронної комерції. Після того, як вони проаналізували усі тогочасні програмні продукти по розсилці електронної комерції, вони вирішили створити свій з приємним дизайном і можливістю керувати декількома обліковими записами одночасно [6].

Campaign Monitor стоїть осторонь, коли справа доходить до розсилання електронної кореспонденції, і не відрізняється в цьому чимось кардинальним тому, що він був побудований спеціально для дизайнерів. В результаті Campaign Monitor дає дизайнерам можливість розробляти електронні листи

за допомогою своїх улюблених інструментів дизайну, створення шаблонів і перегляду скріншотів електронної пошти більш ніж в 20 поштових клієнтах. Їх унікальний професійний підхід і особливості окупилися. Деякі з найбільш відомих у світі компаній використовують Campaign Monitor, в тому числі і Apple, EBay, Twitter, Facebook, Intel, Basecamp і Всесвітній фонд дикої природи.

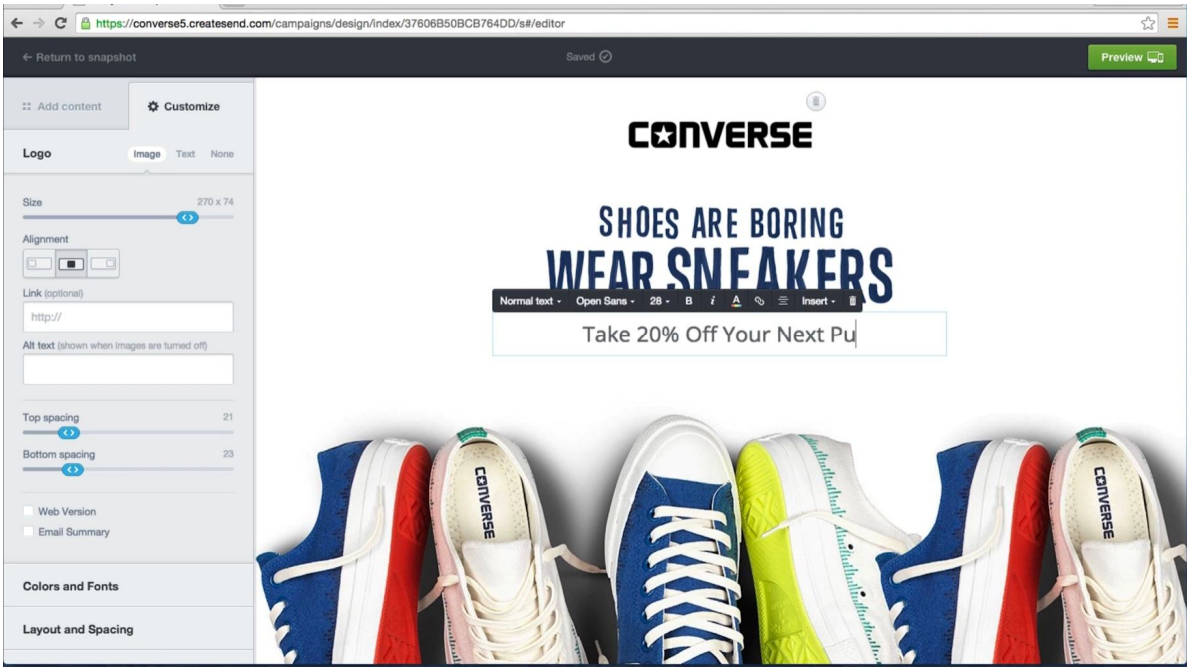

Рисунок 1.3 – Меню створення повідомлення в Campaign Monitor

Ключові особливості

- Інтеграція з Facebook і Twitter;
- Інтеграція з Salesforce, Shopify, Zapier, Podio, Enlist, Unbounce & Wufoo;
- Перетворення RSS в електронну пошту;
- Функція відображення на карті світу (в реальному часі) хто де і коли підписується на розсилку, відкриває лист, переходить по посиланню, також передаючи це їхнім друзям у соціальних мережах;
- Автовідповідач;
- Звітність ROI;
- Мобільний додаток для моніторингу;

- Спліт тестування A / B;
- Змінні інтерфейси;
- Розширювана можливість управління підписками і сегментація списків;
- Професіонали можуть перепродати послуги через унікальну можливість партнерства.

Плюси

- Персоналізований вміст електронної пошти іменем абонента, настроюється вручну;
- Можливість робити дизайн за допомогою улюблених інструментів;
- Галерея з кращими дизайнами повідомлень/кампаній з коментарями експертів;
- Відсутність логотипу, водяних знаків чи лінків Campaign Monitor в середині повідомлень;
- Інтеграція API з CMS, блогом або іншим програмним забезпеченням сторонніх розробників;
- 10% знижка для креативних рекламних агентств для партнерства з їхніми послугами;
- Підтримка клієнтів 24/7;

Мінуси

- Немає підтримки клієнтів через чат;
- Тарифний план Basic має обмежену кількість відсилань на місяць;
- Обмежені ресурси для навчання, відсутність туторіалів;
- Додаткові витрати на спам-тестування з тарифним планом Basic і опції оплати за кожну кампанію;
- Відсутність можливості створення опитувань;
- Відсутність контекстної розсилки, що вважаємо основним недоліком у контексті нашого дослідження.

GetResponse

GetResponse — це одна з найпростіших програм, щоб почати розсилання електронної кореспонденції, і у них є більш 350000 клієнтів, щоб довести це. GetResponse активно допомагає клієнтам покращити їхні кампанії, щоб достукатись і зацікавити аудиторію на кожному кроці проведення кампанії. Інші безоплатні послуги включають в себе: онлайн-опитування, попередній перегляд, автовідповідач і інтеграція соціальних мереж. Всі функції, починаючи від створення емейлів до відстеження активності кампаній і звітності, надиво прості у використанні і розумінні. І якщо цього недостатньо, GetResponse є одним з кращих постачальників послуг розсилання електронної кореспонденції за свої гроші. Багато функцій, які інші сервіси пропонують за окрему плату, включені з щомісячною оплатою GetResponse. Тому GetResponse відмінно підходить як для початківців, так і для тих хто хоче перейти з іншого сервісу [7].

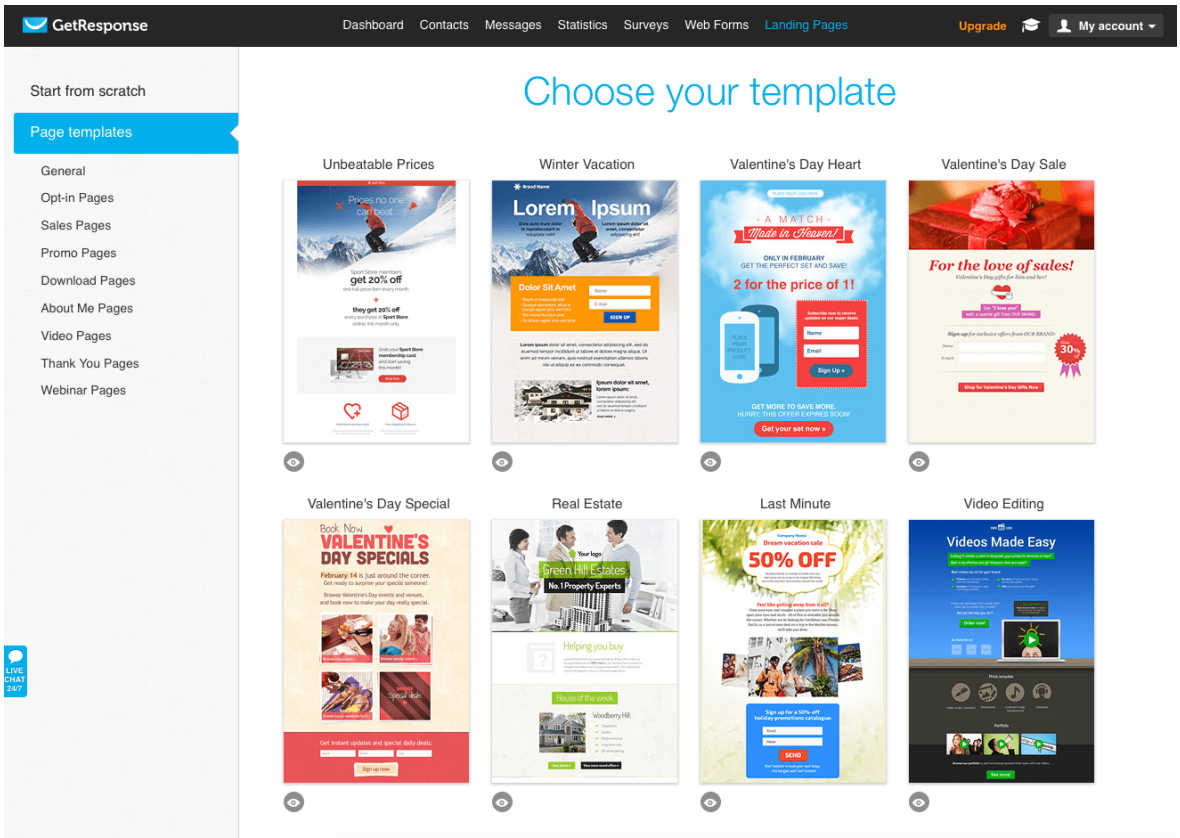

Рисунок 1.4 – Меню створення повідомлення в GetResponse

Ключові особливості

Безліч функцій, що забезпечує GetResponse надаються без додаткової оплати. Найкращими і найбільш популярними з них є:

- Можливість імпортувати контакти з Gmail, Magento, Salesforce і Highrise ;
- Інтеграція з Facebook, Twitter, Google+, LinkedIn і Pinterest;
- App Center: Ви можете приєднати більш ніж 100 програм, в тому числі PayPal, Google Analytics, Joomla, Salesforce, Shopify і багато інших ;
- Мультимедійний центр: створення аудіо та відео файлів;
- Email Intelligence: звітність за часом / днем, автоматизована доставка звітів, статистичні дані по автовідповідачу, соціальна інтеграція і підписна модель;
- Спліт Тестування А / Б;
- Розширена сегментація;
- Генератор QR кодів;
- API для розробників;
- Безкоштовна можливість створювати онлайн-опитування;
- Безкоштовна програма для створення списків, що піднімає маркетинг на новий рівень;
- Адаптивний дизайн електронних повідомлень;
- Створення цільових сторінок.

Плюси

- Легкий у використанні дизайнер повідомлень з drag and drop редактором, редактор HTML для досвідчених користувачів, а також безліч шаблонів;
- Підтримка клієнтів доступна через електронну пошту, телефон і онлайн чат;
- Вбудований сервіс для відстеження статистики;

- 1000+ Istock зображень;
- Spam Assassin інтегрований в редактор повідомлень;
- Можливість перетворити RSS в повідомлення;
- Попередній перегляд повідомлень;
- iPhone і Android додаток;
- Безкоштовна 30-денна пробна демо-версія, без потреби в кредитній карточці;
- Створення цільових сторінок;
- Вебінари;
- Безкоштовна можливість створювати онлайн-опитування;
- Генератор QR кодів;
- Мультимедійний центр: створення аудіо та відео файлів;
- Email Intelligence і розширені функції створення звітів;
- Спліт Тестування А / Б.

Мінуси

- Відсутність безкоштовного тарифного плану;
- Список для сегментації обмежений;
- Редактор повідомлення обмежений в порівнянні з іншими системами;
- Немає підтримки по телефону у вихідні дні;
- Додаткова оплата за виділення IP-адреси;
- Відсутність контекстної розсилки, що є основним у нашому дослідженні.

Проаналізувавши основні програмні продукти на ринку можна відзначити, що спільними рисами для них є: редактор повідомлень, автоматичне розсилання і створення списків користувачів. Потрібно відмітити, що жоден з продуктів не використовує засобів інтелектуального аналізу для оптимізації задачі розсилання електронної кореспонденції. Також виявлено, що деякі продукти надають можливості для збору і аналізу статистики, спліт

тестування і категоризацію клієнтів, але основна частина заходів щодо оптимізації розсилання лягає на самих користувачів. Тому є доцільним створення продукту, який би міг якимось чином оптимізувати і автоматизувати задачу, для зменшення людських зусиль і затрат.

1.2. Проблема автоматизації і оптимізації задачі розсилання електронної кореспонденції

З задачею розсилання електронної кореспонденції люди стикаються дуже часто, для популяризації свого бізнесу, для автоматизації робочого процесу і т.д. Оскільки розсилання електронної кореспонденції не є безплатним процесом, то постає питання в ефективному розподіленні отримувачів цієї кореспонденції, для того, щоб вона була надіслана тільки тим, кому це насправді потрібно.

Процес розсилання електронної кореспонденції на даний момент відбувається таким чином, що список отримувачів ніяк не фільтрується, і більшість кореспонденції йде у кошик для спаму, і для того щоб не набридати користувачам, пропонується фільтрувати повідомлення залежно від уподобань користувача.

Хоча деякі додатки надають засоби для категоризації і аналізу даних, але вони залишаються на примітивному рівні і вимагають людського втручання, розглянемо ці засоби і їх особливості:

Категоризація клієнтів

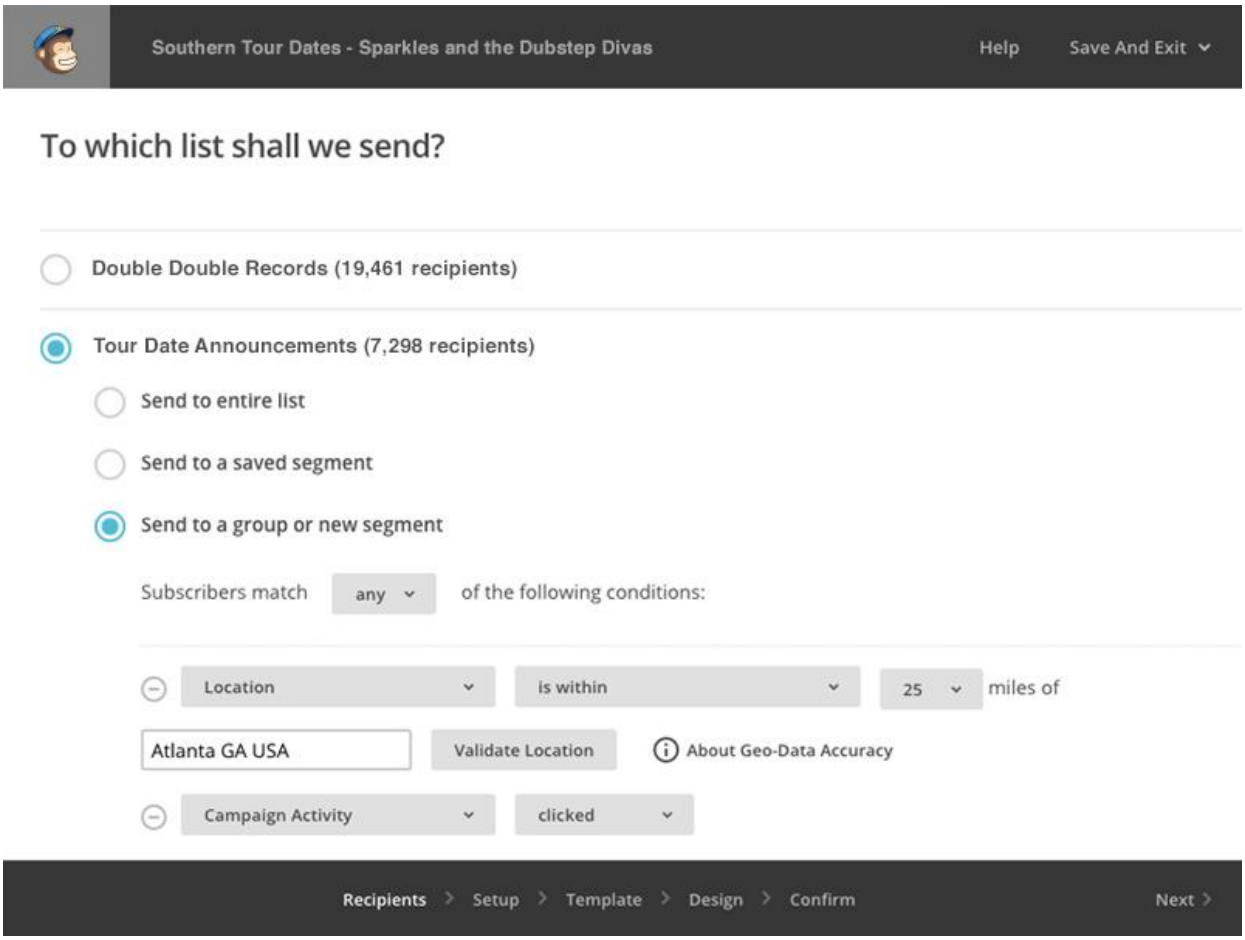

Рисунок 1.5 – Меню категоризації в MailChimp

Як видно із рисунку вище (рис. 1.5), користувачу самому приходиться добавляти категорії, добавляти клієнтів до категорій, і кожен раз перед надсиланням вибирати кому надсилати, що вимагає великих затрат, досконалого знання своїх клієнтів, і включає в себе людський фактор прийняття рішення, що не завжди призводить до найкращих результатів, також кожен раз при добавленні нового сегменту товарів прийдеться добавляти до нього групу.

Відображення статистики клієнтів

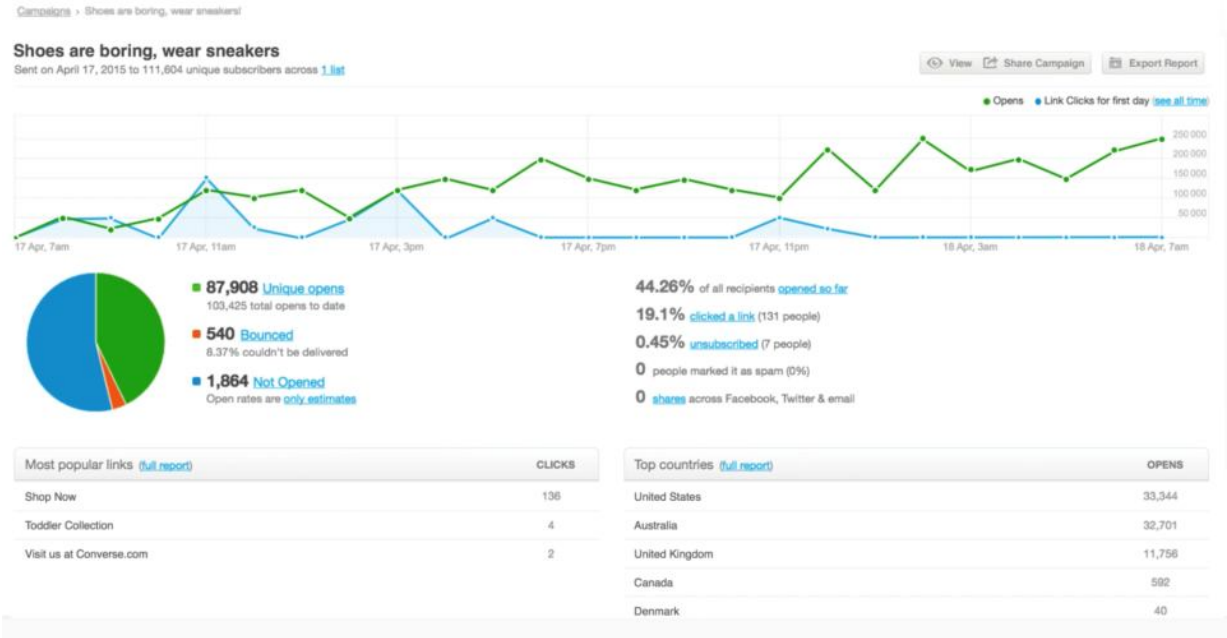

Рисунок 1.6 – Меню відображення статистики в MailChimp

Як видно із рисунку вище (рис. 1.6), користувачу надається різного роду статистика по розісланій кореспонденції, але в той же час ця статистика ніяк не обробляється програмою, наприклад, для пропозицій створення нових груп чи іншого. Користувачу пропонується самому обробляти ці дані і робити висновки, що знову ж таки включає в себе людський фактор прийняття рішення.

Розглянемо проблему автоматизації і оптимізації задачі розсилання електронної кореспонденції з точки зору маркетингу. В цьому випадку електронною кореспонденцією ми вважатимемо рекламу.

Реклама — це будь-яка платна форма неособистої пропозиції товарів чи послуг від імені певного спонсора, для впливу на аудиторію і досягнення поставленої мети. За допомогою реклами формується потрібне уявлення покупця про товар чи послуги.

Мета реклами — привернути увагу, викликати інтерес, передати споживачеві інформацію і змусити його діяти певним чином. Виробити товар ще недостатньо, важливо, щоб він знайшов свого споживача. Тому рекламне звернення має повідомити дещо важливе і цікаве для споживача, про щось виключне, особливе, чого нема в інших товарах. Звернення в рекламі повинно бути правдивим, доказовим і виголошеним доступно, вчасно, щоб покупець звернув увагу на рекламований товар чи послугу і придбав його [8].

На думку спеціалістів [8], рекламу можна розглядати як деяку форму комунікації, яка покликана перекласти параметр якості товару та послуг на мову що виражає потреби споживачів. Під рекламною комунікацією розуміється передача звернення від джерела інформації до її споживача (рис. 1.7.)

Модель рекламної комунікації

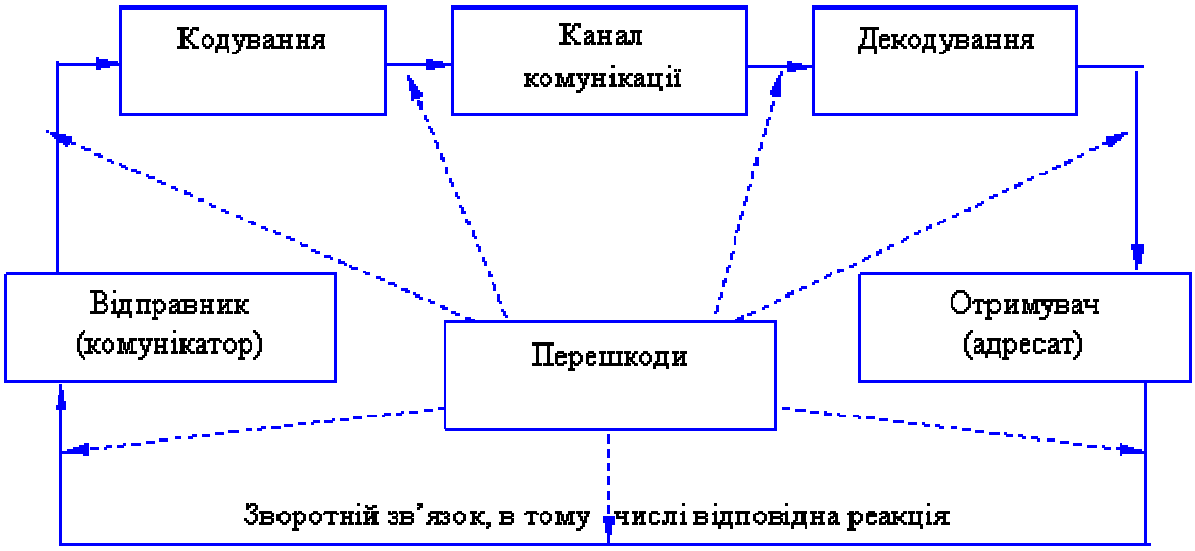

Рисунок 1.7 – Модель рекламної комунікації

Отже, якщо сприймати рекламу як комунікацію, то не варто набридати співрозмовнику нецікавими йому темами, бо це може викликати відразу у нього не тільки до предмета реклами, а й до компанії, яка її надає, тому для уникнення такої прикрої ситуації у маркетингу пропонується сегментувати ринок.

Сегментування ринку — процес знаходження однорідних груп споживачів для пропозиції товарів та послуг, які задовольняють їхні потреби. Сегмент ринку утворюють споживачі, які однаково реагують на той самий набір спонукальних (причинних) стимулів маркетингу [9].

Основою сегментації ринку є типологічне групування споживачів за певними чітко вираженими ознаками. Сегментації ринку зазвичай передує маркетингове дослідження, метою якого є виявлення типу і структури ринку конкретного товару, знаходження ознак, відповідно до яких провадитиметься сегментація споживачів. Особливого значення набуває визначення привабливості сегмента для збуту товару, його потенційна місткість, обґрунтування вибору даного сегменту. Сегментація ринку — це дуалістичний процес; з одного боку, це функція маркетинг-менеджменту, а з іншого — суто статистичний процес, функція маркетингового дослідження [9].

Отже для вирішення проблеми розсилання електронної кореспонденції можна застосувати сегментацію споживачів, і зайнятись так званим цільовим маркетингом (сегментаційна стратегія маркетингу), який потребує проведення трьох основних заходів: сегментації ринку, вибору цільових сегментів ринку та позиціонування товару на ринку. Тобто нам потрібно вибрати серед споживачів тільки тих, яким повідомлення буде насправді цікаве, наприклад, проаналізувавши їхні зацікавлення, які можуть бути представленні товарами, які вони уже переглянули на сайті продажу товарів.

1.3. Теоретична основа сегментації споживачів для автоматизації і оптимізації задачі розсилання електронної кореспонденції

Вирішити задачу сегментації споживачів допоможе одна із задач інтелектуального аналізу даних (Data Mining) – кластеризація. Кластерний аналіз — задача розбиття заданої вибірки об'єктів на підмножини, що називаються кластерами (сегментами), так, щоб кожен кластер складався з схожих об'єктів, а об'єкти різних кластерів істотно відрізнялися. Завдання кластеризації відноситься до статистичної обробки, а також до широкого класу завдань навчання без вчителя [10], тобто кластеризація зберігає свою актуальність при невідомих умовах.

Проблема кластеризації широко вивчається в базах даних і статистичній літературі в контексті широкого спектру завдань інтелектуального аналізу даних. Завдання кластеризації визначається як відшукання групи подібних об'єктів в даних. Подібність між об'єктами вимірюється за допомогою функції подібності. Формалізуємо цей підхід, нехай X — множина об'єктів, Y — множина ідентифікаторів кластерів. Задамо функцію відстані між об'єктами $p(x, x')$. Маємо кінцеву вибірку з об'єктів $X^m = \{x_1, ..., x_m\} \subset X$. Тепер потрібно розбити вибірку на підмножини, що називаються кластерами, так, щоб кожен кластер складався з об'єктів, близьких по метриці p, а об'єкти різних кластерів істотно відрізнялися і кластери не пересікалися. При цьому кожному об'єкту $x_i \in X^m$ приписується ідентифікатор кластеру y_i.

Незалежно від конкретної сфери, застосування кластерного аналізу, він передбачає наступні етапи:

- Формування вибірки об'єктів для кластеризації.
- Визначення множини характеристик, по яких будуть оцінюватися об'єкти у вибірці.

- Обчислення значень подібності, тією чи іншою мірою схожості між об'єктами і утворення вектору значень.
- Нормалізація вектору значень
- Застосування одного з методів кластерного аналізу для створення груп (кластерів) схожих об'єктів.
- Перевірка відповідності результатів кластеризації.

Якщо кластерному аналізу передує факторний аналіз, то вибірка не потребує нормалізації значень — викладені вимоги виконуються автоматично самою процедурою факторного моделювання. В іншому випадку вибірку потрібно нормалізувати, для меншого розкиду результатів.

Традиційні методи кластеризації, як правило, зосереджені на випадках з кількісними даними, в яких атрибути даних є цифровими, а також часто використовується для категоріальних даних, в яких атрибути можуть мати загальні значення, але проблема кластеризації може бути дуже корисною у випадку текстової інформації, де об'єкти кластеру можуть бути різного ступеня деталізації: документи, статті, речення або терміни. Проблема кластеризації знаходить застосування для цілого ряду завдань аналізу текстових даних:

- Організація документів і перегляд: Ієрархічна організація документів в чіткі і зрозумілі категорії може бути дуже корисною для систематичного перегляду колекції документів. Класичним прикладом цього є Scatter / Gather метод, який забезпечує систематичний метод перегляду з використанням кластерної організації колекції документів [11].
- Підсумок ключових слів/термінів у збірках документів: Методи кластеризації забезпечують чіткий список ключових слів збірки у вигляді словесних кластерів, які можуть бути використані для того, щоб представити коротке розуміння загального змісту всієї збірки.

Також методи кластеризації можна використовувати для формування ключових термінів окремого документу чи частини тексту.

- Класифікація документів: У той час як кластеризація за своєю природою є неконтрольованим методом навчання, але вона може бути використана для того, щоб поліпшити якість результатів в її підконтрольному варіанті. Зокрема, словесні кластери та методи спільного навчання можуть бути використані для того, щоб поліпшити точність класифікації з допомогою методів кластеризації.

Багато класів алгоритмів, таких як алгоритм К-середніх, або ієрархічні алгоритми є методами загального призначення, які можуть бути розширені для будь-якого типу даних, включаючи текстові дані.

Текстовий документ може бути представлений або в двійковому вигляді, коли ми використовуємо наявність або відсутність слова в документі, для того щоб створити бінарний вектор. У таких випадках можна безпосередньо використовувати різні категорійні алгоритми кластеризації даних для двійкового представлення.

Більш розширене представлення текстових даних буде включати в себе покращенні методи зважування на основі частот окремих слів у документі, а також частоти слів у цілій колекції (наприклад, TF-IDF метод). І тоді кількісні алгоритми кластеризації даних можуть бути використані в поєднанні з цими ваговими значеннями, щоб визначити найбільш близькі групи об'єктів в колекції текстових даних.

Проте, такі примітивні методи зазвичай не працюють добре для кластеризації текстових даних. Це тому, що текстові дані мають ряд унікальних властивостей, які роблять необхідним розробку спеціалізованих алгоритмів для виконання цього завдання. Відмінними характеристиками текстових даних є наступні:

- Розмірність загального текстового масиву даних є дуже великою, але текстові об'єкти масиву є мізерними у порівнянні. Іншими словами, словниковий запас, з якого створюються документи може містити мільйони слів, але даний документ може містити тільки кілька сотень слів. Ця проблема є ще більш серйозною, коли документи, які повинні бути згруповані є дуже короткими (наприклад, коли кластеризують речення або повідомлення з соціальних мереж).
- У той час як словник даної колекції документів може бути величезним, слова, як правило, корелюють одне з одним, є синонімами один одному. Це означає, що кількість понять (або основних компонентів) в даних значно менша, ніж основних ознак. Це вимагає ретельної розробки алгоритмів, які можуть усунути кореляції в процесі кластеризації.
- Кількість слів (або ненульових елементів) в різних документах, може варіюватися в широких межах. Тому, важливо, нормалізувати представлення документа відповідним чином в ході завдання кластеризації.

Програмне забезпечення, на основі застосування одного із методів кластерного аналізу повинне розподіляти користувачів та електронну кореспонденцію на групи(кластери) і в залежності від розподілу, групі в яку попала кореспонденція, усім користувачам які теж у неї попали буде надсилатися повідомлення.

Висновки до розділу 1

У першому розділі проведено аналіз основних моментів і проблем розробки оптимізованих систем розсилання електронної кореспонденції, а саме:

1. Досліджено що існуючі програмні продукти для розсилання електронної кореспонденції не використовують засобів інтелектуального аналізу для оптимізації задачі розсилання електронної кореспонденції, тому в магістерській роботі поставлено за мету створення продукту який би міг оптимізувати і автоматизувати задачу за допомогою засобів інтелектуального аналізу.
2. Проаналізувавши методи інтелектуального аналізу, визначено що для оптимізації задачі розсилання електронної кореспонденції буде оптимальним використати один із методів кластерного аналізу і кластеризувати дані для розсилки і дані що характеризують користувачів.

РОЗДІЛ 2
РОЗРОБКА МАТЕМАТИЧНОГО І ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ДЛЯ ЗАДАЧІ РОЗСИЛАННЯ ЕЛЕКТРОННОЇ КОРЕСПОНДЕНЦІЇ

2.1. Математичне забезпечення застосунка

В попередньому розділі було розглянуто основні проблеми, що виникають при інтелектуальному аналізі текстових даних який буде використовуватись для рішення задачі розсилання електронної кореспонденції. В цьому параграфі опишемо та обґрунтуємо основні методи, що необхідні для розв’язання поставлених проблем.

Одною з перших проблем що виникає при кластеризації текстових даних це кількість слів (або ненульових елементів) в різних документах, може варіюватися в широких межах. Тому, важливо, нормалізувати представлення документа відповідним чином в ході завдання кластеризації. Якість будь-якого методу видобутку даних, таких як класифікація і кластеризація сильно залежить від шумових слів - це слова, які не несуть смислового навантаження, тому їх користь та роль для пошуку не суттєва[12], які використовуються для процесу кластеризації. Тому першим кроком для кластеризації текстових даних є позбавлення від так званих стоп-слів. Для кожної мови вони є різними, наприклад для української це такі слова як:

- цифри: 1, 2, 3, 4, 5, 6, 7, 8, 9, 0 (один, два, три, чотири, п'ять, шість, сім, вісім, дев'ять, нуль).
- окремо розташовані знаки пунктуації: . , = + /! "; :%? * ()
- окремо розташовані букви алфавіту: а, б, в, г, ґ, д, е, є, ж, з, и, і, ї, й, к, л, м, н, о, п, р, с, т, у, ф, х, ц, ч, ш, щ, ь, ю,я
- займенники, дієприкметники, прийменники, вигуки, суфікси і поєднання букв: без, більш, б, був, була, були, було, бути (окрім фразеологічних зворотів, таких як «бути чи не бути»), вам, вас, адже, весь, вздовж, замість, поза, вниз, внизу, всередині, під, навколо, от, все, завжди, все, всіх, ви, де, да, давай, давати, навіть, для, до і т. д.

Для кожної мови є свої стоп-слова і передбачити їх іншим чином як вручну вибравши – неможливо, тому тут варто створити словники для кожної з мов і вичищати текст від цих слів. Але оскільки ці слова є унікальними для кожної з мов потрібно визначити мову повідомлення, оскільки це не є основною задачею дослідження, то для цього будуть використанні сторонні програмні методи описанні у §3.1, які основані на моделі триграм які детальніше описанні у [13] і [14].

Після того як було визначено мову повідомлень, усунуто стоп-слова, наступним кроком є усунення слів, які корелюють один з одним, відмінків слів. Для цього ми скористаємось процесом стемінгу - це процес скорочення слова до основи шляхом відкидання допоміжних частин, таких як закінчення чи суфікс. Результати стемінгу іноді дуже схожі на визначення кореня слова, але його алгоритми базуються на інших принципах. Тому слово після обробки алгоритмом стемінгу (стематизації) може відрізнятися від морфологічного кореня слова. Стемінг застосовується в лінгвістичній морфології та в інформаційному пошуку. Багато пошукових систем використовують стемінг для об’єднання слів у яких збігаються форми після стематизації, вони вважають такі слова синонімами. Цей процес називають злиттям. В процесі стемінгу слова "красиво", "красивий", "красиві" будуть перетворені у форму "красив". А от такі слова як "бігом", "бігаю", "бігати" взагалі скорочуються до спільного кореня слова "біг". Самим популярним алгоритмом для стемінгу є Стеммер Портера - алгоритм стемінгу, опублікований Мартіном Портером в 1980 році[15]. Оригінальна версія стеммеру була призначена для англійської мови і була написана на мові BCPL. Згодом Мартін створив проект «Snowball» використовуючи основну ідею алгоритму[16]. Алгоритм не використовує баз основ слів, а лише, застосовуючи послідовно ряд правил, відсікає закінчення і суфікси, грунтуючись на особливостях мови, в зв'язку з чим працює швидко, але не завжди безпомилково.

Останнім кроком “очистки” текстових даних є знаходження і ліквідація синонімів за значенням, для цього єдиним варіантом є створення словника синонімів і виключення слів за цим словником, деталі реалізації цього процесу розкриті у §3.1.

Після “очистки” текстового масиву від шумів можна приступати до початку кластеризації. Для того, щоб забезпечити ефективний процес кластеризації, частоту появи слів потрібно нормалізувати відносно їх відносної частоти присутності в документі і у всій колекції. Зазвичай представлення яке використовується для обробки тексту це вектор на основі TF-IDF представлення. У представленні TF-IDF, частота терміну для кожного слова нормалізована за допомогою зворотньої частоти у документі, або IDF. Нормалізація за допомогою зворотньої частоти у документі зменшує вагу термінів, які зустрічаються частіше у колекції. Це зменшує значення загальних термінів у колекції, гарантуючи, що порівняння документів більшою мірою буде залежати від більш важливих слів, які мають відносно низькі частоти в колекції. Найбільш поширена схема TF-IDF дає слову w в документі d вагу відповідно до:

$$TF\_IDF\_Weight(w,d) = TermFreq(w,d) \cdot \log(N / DocFreq(w)), \qquad (2.1)$$

де $TermFreq(w,d)$ - це частота появи слова в документі, N - кількість всіх документів і $DocFreq(w)$ це кількість документів що мають слово w.

Оскільки ми представляємо текстовий документ як вектор з TF-IDF значень, для виміру подібності документів найкраще застосувати одну з функцій визначення дистанції. Найбільш відомою функцією знаходження дистанції, яка використовується у в текстовій області є функція - коефіцієнт Отиаи (cosine similarity). Нехай $U = (f(u_1)...f(u_k))$ і $V = (f(v_1)...f(v_k))$ є очищеними і нормалізованими векторами файлів U і V. Параметри $u_1...u_k$ і $v_1...v_k$ представляють собою нормалізовані частоти термінів і функція $f(.)$

представляє функцію очистки від шумів. Тоді дана функція дистанції між двома цими документами визначається наступним чином:

$$COSINE(U,V) = \frac{\sum_{i=1}^{k} f(u_i) \cdot f(v_i)}{\sqrt{\sum_{i=1}^{k} f(u_i)^2} \cdot \sqrt{\sum_{i=1}^{k} f(v_i)^2}}, \quad (2.2)$$

Отриманим результатом буде схожість в діапазоні від -1 означає повністю протилежний, до 1 – повністю такий самий, де 0 означає декорреляцію значення у проміжку вказують на проміжну схожість або відмінність.

Для кластеризації обрано метод к-середніх, адже цей алгоритм найкраще показує себе на великих обсягах даних[17]. Цей алгоритм кластеризації був винайдений в 1950-х роках математиком Гуґо Штайнгаузом[і майже одночасно Стюартом Ллойдом, особливу популярність отримав після виходу роботи МакКвіна[18], і є одним з найпростіших і найбільш відомих алгоритмів неконтрольованого навчання, які вирішують відому проблему кластеризації. Починається кластеризація з вибору k початкових документів що будуть представляти кластери, з всіх документів і добавляння до них інших документів за функцією схожості, в основному початкові кластери обираються випадково але у нашому випадку k буде рівне двом, а саме документу кореспонденції і найдальшому документу від кореспонденції . В наступній ітерації розраховується новий центр кожного кластера, ваги якого розраховуються як середнє арифметичне усіх вагів що входять у цей кластер. Все продовжується доти поки не відбувається жодних змін у центроїдах або вийдено за ліміт кількості ітерацій. Алгоритм працює в такий спосіб:

Ініціалізація:

k документів (у нашому випадку 2) , утворюють ядро k кластерів. Усі інші вектори присвоюються кластеру найближчого з документів.

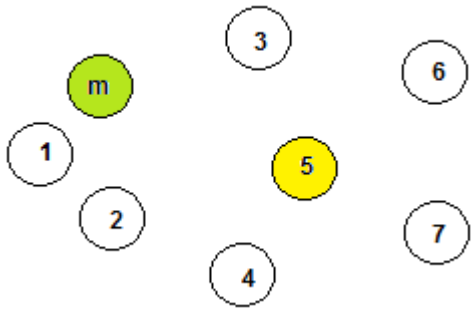

Рисунок 2.1 – Графічна демонстрація вибору початкових документів як центрів кластерів, де m це документ кореспонденції, а 5 - це найвіддаленіший документ

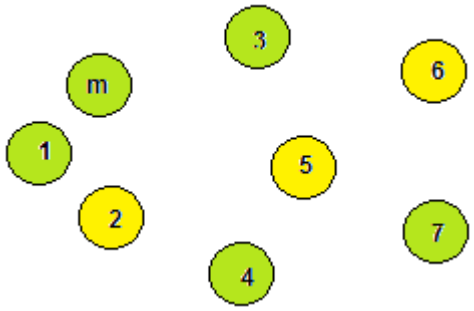

Рисунок 2.2 – Графічна демонстрація присвоєння документів найближчому кластеру

Ітерація:

обраховуються центроїди M_i поточних кластерів:

$$M_i = \left|C_i\right|^{-1} \sum_{x \in c_i} x \qquad (2.3)$$

Кожен документ переприсвоюється до найближчого центроїда

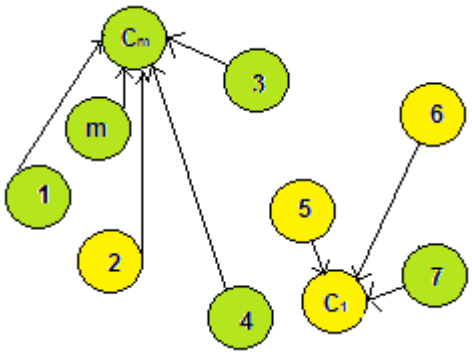

Рисунок 2.3 – Графічна демонстрація утворення центроїда, і переприсвоєння документів

Умова зупинки:

не відбувається жодних змін у центроїдах або вийдено за ліміт кількості ітерацій

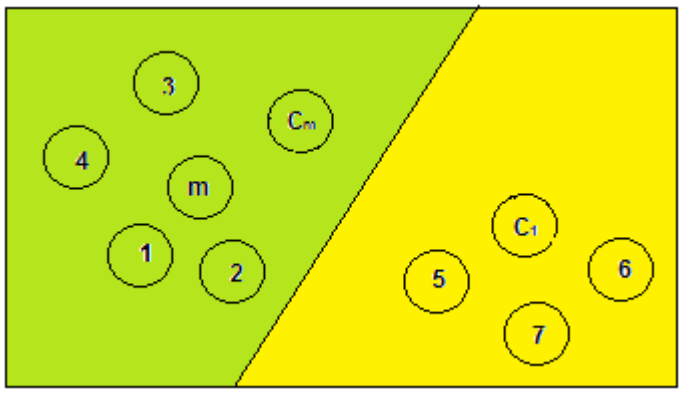

Рисунок 2.4 – Графічна демонстрація результатів після кластеризації

Загальний алгоритм виглядає так:

Алгоритм 2.1.

Алгоритм кластеризації електронної кореспонденції

Крок 1. Зібрати дані у вигляді текстових документів.

Крок 2. Визначити мову кожного документу і відсіяти документи з відмінною мовою ніж у повідомлення.

Крок 3. Очистити документи від стоп-слів.

Крок 4. Очистити документи від синонімів.

Крок 5. Застосувати стемінг до кожного з термінів у документах.

Крок 6. Визначити векторні значення термінів у документах за допомогою TF-IDF функції.

Крок 7. Запустити процес кластеризації.

2.2. Проектування системи

З задачею розсилання електронної кореспонденції люди стикаються дуже часто, для популяризації свого бізнесу, для автоматизації робочого процесу і тд. Оскільки розсилання електронної кореспонденції не є безплатним процесом, то постає питання в ефективному розподіленні отримувачів цієї кореспонденції, для того щоб вона була надіслана тільки тим кому це насправді потрібно.

Процес розсилання електронної кореспонденції на даний момент відбувається таким чином, що список отримувачів ніяк не фільтрується, і більшість кореспонденції йде у кошик для спаму, і для того щоб не набридати користувачам пропонується фільтрувати повідомлення залежно від уподобань користувача, уподобання користувача повинні бути преставленні текстовими документами.

Вирішити дану задачу допоможе одна із задач інтелектуального аналізу даних (Data Mining) – кластеризація. Програмного забезпечення, на основі застосування одного із методів кластерного аналізу розподілятиме користувачів та електронну кореспонденцію на групи і в залежності від розподілу груп буде надсилати повідомлення. Загальну схему організаційної структури системи проілюстровано на рисунку 2.5 .

Рисунок 2.5 – Організаційна структура системи розсилання електронної кореспонденції

Проаналізувавши предметну область, виділено два основні бізнес-процеси, які необхідно буде автоматизувати за допомогою розроблюваної програмної системи:

- кластеризація даних;
- розсилання повідомлення;

Кластеризація даних – процес, що включає в себе читання, фільтрування та обробка даних у системі (рисунок 2.6).

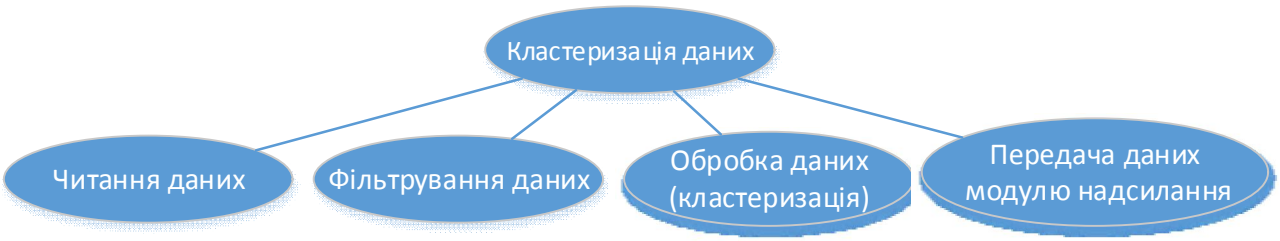

Рисунок 2.6 – Дерево функцій бізнес-процесу «Кластеризація даних»

Характеристика цього бізнес-процесу представлена в таблиці 2.1

Таблиця 2.1

Характеристика бізнес-процесу «Кластеризація даних»

Назва характеристики	Значення характеристики
Ім’я бізнес-процесу	Кластеризація даних
Основні учасники	Повідомлення, дані клієнтів
Вхідна подія	Ініціювання відправлення повідомлення адміністратором
Вхідні документи	Немає
Вихідна подія	Відправка повідомлення
Вихідні документи	Дамп кластерів
Клієнт бізнес-процесу	Система

Розсилання повідомлення є процесом, що відбувається у системі після визначення клієнтів яким можна його надіслати, проводиться передача повідомлення клієнтам через SMTP сервер (рисунок 2.7).

Рисунок 2.7 – Дерево функцій бізнес-процесу «Розсилання повідомлення»

Характеристика цього бізнес-процесу представлена в таблиці 2.2.

Таблиця 2.2

Характеристика бізнес-процесу «Розсилання повідомлення»

Назва характеристики	Значення характеристики
Ім’я бізнес-процесу	Розсилання повідомлення
Основні учасники	Система, SMTP сервер, користувач
Вхідна подія	Запит на розсилання повідомлення
Вхідні документи	Немає
Вихідна подія	Збереження даних
Вихідні документи	Немає
Клієнт бізнес-процесу	Система

Ознайомившись з основними термінами предметної області, приступаємо до опису варіантів використання програмної системи розсилання електронної кореспонденції.

Проаналізувавши всі вимоги до системи, виділено основні важливі варіанти використання, які проілюстровані на діаграмі варіантів використання системи (рисунок 2.8).

Рисунок 2.8 – Діаграма варіантів використання для адміністратора

Проведемо детальний опис варіантів використання, що зображені на діаграмах варіантів використання (рисунок 2.8). Детальний опис характеристик варіантів використання системи, які подані на рисунку 2.4, наведено у таблицях 2.3 – 2.16.

Таблиця 2.3

Варіант використання «Перегляд стану надсилання повідомлення»

Контекст використання	Перегляд стану надсилання повідомлення
Дійові особи	Адміністратор
Передумова	Вхід в систему
Тригер	Немає
Сценарій	1. Перейти у панель керування повідомленнями
Постумова	Виводяться усі повідомлення

Таблиця 2.4

Варіант використання «Відміна надсилання повідомлення»

Контекст використання	Відміна надсилання повідомлення
Дійові особи	Адміністратор
Передумова	Вхід в систему
Тригер	Немає
Сценарій	1. Перейти у панель керування повідомленнями 2. Обрати повідомлення і натиснути кнопку видалити 3. Підтвердити
Постумова	Виводиться повідомлення про стан виконання операції

Таблиця 2.5

Варіант використання «Надсилання повідомлення»

Контекст використання	Надсилання повідомлення
Дійові особи	Адміністратор
Передумова	Вхід в систему
Тригер	Немає
Сценарій	1. Перейти у панель керування повідомленнями 2. Натиснути кнопку додавання нового повідомлення 3. Ввести текст повідомлення 4. Підтвердити
Постумова	Виводиться повідомлення про стан виконання операції

Таблиця 2.6

Варіант використання «Видалення контактів»

Контекст використання	Видалення контактів
Дійові особи	Адміністратор
Передумова	Вхід в систему
Тригер	Немає
Сценарій	1. Перейти у панель керування контактами 2. Обрати контакт і натиснути кнопку видалити 3. Підтвердити
Постумова	Виводиться повідомлення про стан виконання операції

Таблиця 2.7

Варіант використання «Редагування контактів»

Контекст використання	Редагування контактів
Дійові особи	Адміністратор
Передумова	Вхід в систему
Тригер	Немає
Сценарій	1. Перейти у панель керування контактами 2. Обрати контакт і натиснути кнопку редагувати 3. Ввести нові дані 4. Підтвердити
Постумова	Виводиться повідомлення про стан виконання операції

Таблиця 2.8

Варіант використання «Добавлення контактів»

Контекст використання	Добавлення контактів
Дійові особи	Адміністратор
Передумова	Вхід в систему
Тригер	Немає
Сценарій	1. Перейти у панель керування контактами 2. Натиснути кнопку додавання контакту 3. Ввести дані 4. Підтвердити
Постумова	Виводиться повідомлення про стан виконання операції

Таблиця 2.9

Варіант використання «Вхід в систему»

Контекст використання	Вхід в систему
Дійові особи	Адміністратор
Передумова	Немає
Тригер	Немає
Сценарій	1. Запустити систему 2. Ввести свої дані 3. Натиснути кнопку "Увійти" 4. Підтвердити
Постумова	Авторизація

Таблиця 2.10

Варіант використання «Особистий кабінет»

Контекст використання	Перегляд особистого кабінету
Дійові особи	Адміністратор
Передумова	Вхід в систему
Тригер	Немає
Сценарій	1. Перейти на панель особистого кабінету
Постумова	Відображення особистого кабінету

Таблиця 2.11

Варіант використання «Зміна паролю»

Контекст використання	Зміна паролю
Дійові особи	Адміністратор
Передумова	Вхід в систему
Тригер	Немає
Сценарій	1. Перейти на панель особистого кабінету 2. В особистому кабінеті натиснути кнопку зміни паролю 3. Ввести необхідні дані 4. Підтвердити
Постумова	Виводиться повідомлення про стан виконання операції

Таблиця 2.12

Варіант використання «Перегляд результатів кластеризацій»

Контекст використання	Перегляд результатів кластеризацій
Дійові особи	Адміністратор
Передумова	Вхід в систему
Тригер	Немає
Сценарій	1. Перейти у панель звітів 2. Вибрати пункт "Кластеризації" 3. Натиснути кнопку перегляду
Постумова	Відображення результатів кластеризацій

Таблиця 2.13

Варіант використання «Перегляд стану рахунку SMTP серверу»

Контекст використання	Перегляд стану рахунку SMTP серверу
Дійові особи	Адміністратор
Передумова	Вхід в систему
Тригер	Немає
Сценарій	1. Перейти у вкладку SMTP серверу
Постумова	Відображення стану рахунку SMTP серверу

Таблиця 2.14

Варіант використання «Вихід»

Контекст використання	Вихід з системи
Дійові особи	Адміністратор
Передумова	Вхід в систему
Тригер	Немає
Сценарій	1. Натиснути кнопку виходу
Постумова	Вихід з системи

Таблиця 2.15

Варіант використання «Перегляд звітів надсилань»

Контекст використання	Перегляд звітів надсилань
Дійові особи	Адміністратор
Передумова	Вхід в систему
Тригер	Немає
Сценарій	1. Перейти у панель звітів 2. Вибрати пункт "Надсилання" 3. Обрати звіт і натиснути кнопку перегляду
Постумова	Відображення результатів обраного надсилання

Специфікація не функціональних вимог наведена у таблиці 2.16

Таблиця 2.16

Специфікація не функціональних вимог

№. вимоги	Назва вимоги	Атрибути вимоги		
		Пріоритет	Складність	Характеристика
1.	Застосовність			
1.1	Час, необхідний для навчання звичайних і досвідчених користувачів	Рекомендоване	Середня	Час , необхідний для навчання користувачів не повинен перевищувати одного дня
1.2	Вимірний час відгуку для типових завдань	Обов’язкова	Висока	Час відгуку не повинен перевищувати 2секунд

Продовження таблиці 2.16

1.3	Основні вимоги застосовності нової системи відносно інших систем, які знають користувачі	Опціоне	Низька	Оскільки це новий продукт і користувачі раніше не користувались іншими системами то ця вимога є неважливою
2.	Надійність			
2.1	Доступність	Рекомендов ане	Середня	Час що витрачається на обслуговування не має перевищувати 3% від часу експлуатації системи
2.2	Середній час безвідмовної роботи	Обов’язкова	Висока	Середній час безвідмовної роботи не має бути менше 7 днів
2.3	Середнє напрацювання до ремонту	Обов’язкова	Висока	Система повинна працювати не менше одного місяця до того, коли повинне бути проведене її обслуговування
2.4	Максимальна норма помилок або дефектів	Обов’язкова	Висока	Повинно бути не більше 1ї помилок на тисячу рядків коду

Продовження таблиці 2.16

3.	Робочі характеристики			
3.1	Швидкодія для транзакції	Обов’язкова	Висока	1-2 сек
3.2	Використання ресурсів	Рекомендоване	Середня	• OS: Windows • SQL Server 2008 •.NET 4.5 • RAM 1 GB • CPU 1.5 MHz • 100 MB Free Space • Internet speed 32kb/sec
4.	Інтерфейси			
4.1	Комунікаційні інтерфейси	Обов’язкова	Висока	У сервера повинен бути статичний ір адрес, також має бути вільний 80й порт
4.2	Програмні інтерфейси	Обов’язкова	Середня	На сервері має стояти Windows OS
5	Гнучкість	Обов’язкова	Висока	Розроблена архітектура має бути гнучкою до внесення змін і до додавання нових елементів

Продовження таблиці 2.16

6	Ідентифікація користувачів системи	Обов’язкова	Висока	Розроблювана ПС має містити у собі опис ролей користувачів (Користувач, Адміністратор) та довідник по роботі даних користувачів. У системі має бути забезпечена можливість реєстрації користувачів і визначення їх типу.
7	Адаптованість	Необов’язкова	Висока	Додаток повинен працювати під різними версіями Windows (починаючи від Windows XP)

Висновки до розділу 2

У другому розділі проведено проектування основних елементів системи для розсилання електронної кореспонденції а саме:

1. Визначено алгоритм для оптимізації і автоматизації задачі розсилання електронної кореспонденції.
2. Проаналізовано предметну область, спроектовано діаграму варіантів використання і розписано кожен варіант, що дозволяє приступити до реалізації системи .

РОЗДІЛ 3
ПРОГРАМНА РЕАЛІЗАЦІЯ І ДОСЛІДЖЕННЯ ЗАСТОСУНКА

3.1. Реалізація системної частини системи

Розроблювана система у магістерській роботі є комбінацією системної та прикладної частин. Роль системної частини полягає у кластеризації текстових даних що прийшли ззовні. Також роль системної частини включає в себе розсилання кореспонденції залежно від результатів кластеризації. Роль прикладної частини полягає у наданні користувачу інтерфейсу для роботи з системною частиною додатку і візуалізації результатів роботи додатку.

Для реалізації обрано платформу .NET та мову програмування C#, графічна (прикладна) підсистема Windows Forms а також середовище розробки Microsoft Visual Studio 2012.

C# - високорівнева мультипарадигменна об'єктно-орієнтована багатоплатформова мова програмування загального призначення яка швидко розвивається та дозволяє максимально зекономити час при розробці програмного продукту. Швидкодія програм написаних на C# є досить висока. Перевагою даної мови є те, що розмір результуючих програмних продуктів є дуже малий і не потребує додаткових бібліотек. Для виконання програм написаних на мові C# необхідно щоб на комп’ютері був встановлений .NET Framework відповідної версії. Також перевагою C# програм є виконання на різних платформах, для яких розроблений .NET Framework.

Microsoft Visual Studio — серія програмних продуктів компанії Майкрософт, які включають у себе - інтегроване середовище для розробки програмного забезпечення та ряд інших інструментальних і програмних засобів. Ці продукти дозволяють розробляти як прості консольні програми, так і програми з продвинутим графічним інтерфейсом, в тому числі з підтримкою технології Windows Forms, а також веб-сайти, веб-додатки, веб-служби як у рідному, так і у керованому середовищі для всіх платформ, що підтримуються у Microsoft Windows, Windows Mobile, Windows CE, .NET

Framework, .NET Compact Framework та Microsoft Silverlight, а також з останнього часу у Mono.

Windows Forms – інтерфейс(API) для програмування прикладних додатків, що відповідає за графічний інтерфейс користувача і є частиною Microsoft .NET Framework. Даний інтерфейс спрощує доступ до елементів інтерфейсу Microsoft Windows за рахунок створення обгортки навколо існуючого Win32 API у керованому середовищі. Причому кероване середовище включає класи, що реалізують API для Windows Forms і не залежать від програмної мови розробки. Тобто розробник однаково може використовувати інтерфейс Windows Forms як для написання ПЗ на C#, C++, так і на VB.Net, J# та інших.

З одного боку, Windows Forms розглядається як заміна старішої і складнішої бібліотеки MFC, спочатку написаної на мові C++. З іншого боку, Windows Forms не пропонує парадигму схожу з MVC. Для виправлення цієї ситуації і реалізації даного функціоналу в Windows Forms існують сторонні бібліотеки і модулі. Однією з найбільш використовуваних подібних бібліотек є User Interface Process Application Block, випущена спеціальною групою Microsoft, що займається прикладами і рекомендаціями, для безкоштовного розповсюдження. Ця бібліотека також містить початковий код і навчальні приклади для прискорення навчання.

Додатки на Windows Forms орієнтуються на події, що підтримуються у Microsoft .NET Framework. На відміну від пакетних програм що виконуються в одному руслі, основною задачею прикладних програм на Windows Forms є очікування від користувача будь-яких дій, як, наприклад, введення даних у відповідне поле або кліку мишкою по елементу. Усередині .NET Framework, Windows Forms розміщена в рамках простору імен System.Windows.Forms.

Для програмної реалізації системи обрано модульно-інтерфейсну архітектуру, для надання можливості розширення в подальшому як окремих модулів так і самої системи. Модульна архітектура - це організація коду програми як сукупності окремих, незалежних блоків(модулів), структура і

поведінка яких підпорядковується заздалегідь визначеним правилам. Використання модульної архітектури програмування дозволяє спростити задачу тестування ПЗ і виявлення помилок, адже апаратно-залежні підзадачі(наприклад розсилання повідомлення через SMTP) можуть бути чітко відділені від інших підзадач, що покращує мобільність створюваних систем. Принцип модульності є засобом спрощення задачі проектування ПС і розподілу процесу розробки ПС між групами розробників. При розбитті ПС на модулі для кожного модуля вказується реалізована їм функціональність, а також зв'язки з іншими модулями. Зручність у використанні модульної архітектури полягає в можливості зміни чи розширення модуля, без необхідності внесення змін у інші частини системи.

Роль програмних модулів можуть виконувати окремі структури даних, бібліотеки, класи, сервіси, програмні елементи, що реалізують певну функціональність і надають програмний інтерфейс доступу до неї.

Програмний код зачасту розбивається на декілька окремих файлів, кожен з яких компілюється окремо від інших. Така модульність програмних елементів дає досить значуще зменшення часу для внесення змін у систему, зміни вносяться лише у файли/модулі які цього потребують, а система зразу їх підхвачує, також такий підхід спрощує командну розробку, і надає можливість для заміни окремих компонентів системи (таких як jar або dll бібліотеки) кінцевого програмного продукту, без необхідності повторної збірки всього проекту (наприклад, розробка плагінів для уже готової системи).

Одним з методів написання модульних програм є об'єктно-орієнтоване програмування. ООП забезпечує високу ступінь модульності завдяки таким властивостям, як інкапсуляція, поліморфізм і пізнє зв'язування[19] Інтерфейси дозволяють реалізувати множинне наслідування об'єктів і в той же час вирішити проблему ромбовидного успадкування.

Інтерфейс - це програмна/синтаксична структура, що визначає відношення між об'єктами, які поділяють певну поведінкову модель і не

пов'язані ніяк інакше. При проектуванні класів, розробка інтерфейсу тотожна розробці специфікації (безлічі методів, який кожен клас, який використовує інтерфейс повинен реалізовувати). Інтерфейс надаючи певну абстракцію, окреслює чітку межу взаємодії між класами або компонентами яка здійснюється і реалізується стороною наслідування.

Відповідно до рисунку 2.5 у системі потрібно реалізувати 2 модуля:

- Модуль кластеризації
- Модуль надсилання

А також прикладну частину для взаємодії користувача з системою.

Розпочнем з реалізації модулю кластеризації, адже основною задачею дослідження є оптимізація процесу розсилання електронної кореспонденції. Основним класом даного модулю є клас MessageClassificator через який буде відбуватись вся робота пов’язана з кластеризацією, для підтримки розширюваності модулю і для можливості перевикористання модулю у подальшому клас MessageClassificator у своєму конструкторі приймає перелік інтерфейсів що можуть замінити його стандартні елементи, а саме:

- IDataProvider
- ILanguageDetector
- ISynonymsDetector
- IStopWordsProvider
- IStemmersProvider

Сам MessageClassificator реалізовує інтерфейс IDisposable, який змушує його реалізовувати функцію Dispose, потрібно вона для того щоб звільнити контрольовані ресурси, в нашому випадку це елемент модулю для пошуку синонімів, який детальніше розкриється пізніше. Також MessageClassificator має метод ClusterToUserData, який приймає 2 параметра:

- string message – основне повідомлення(кореспонденція) відносно якої буде проводитись кластеризація
- int clastersCount=2 – кількість кластерів для утворення, по замовчуванню стоїть 2, тобто утвориться в результаті виконання

кластеризації 2 кластера, один з усіма елементами що схожі до основного повідомлення і усі інші, при потребі кількість утворених кластерів можна збільшити, але не зменшити.

Повертає функція список з кластерів(центроїдів).

Відповідно до алгоритму 2.1 розглянемо елементи для його виконання, реалізовані у модулі MessageClassificator.

Для збору інформації про користувачів використовується інтерфейс IDataProvider, він є обов'язковим атрибутом при створенні класу MessageClassificator, у інтерфейсі є всього лиш одне поле – функція GetUserData(), що повертає колекцію інтерфейсів IUserData, що представляють собою дані про клієнта компанії. Інтерфейс IUserData складається з 2 полів, а саме:

- string Id – унікальний ідентифікатор клієнта для того щоб ідентифікувати дані після кластеризації за допомогою цього параметру
- List<string> Data – колекція текстових даних, що собою представляють інтереси клієнта, і будуть використані для кластеризації

Інтерфейси IDataProvider і IUserData повинні бути реалізовані на стороні прикладної частини.

Наступним елементом є інтерфейс ILanguageDetector, що відповідає за визначення мови повідомлення, його можна перевизначити передавши з прикладної частини у конструктор MessageClassificator, якщо цього не зроблено – буде використано стандартний елемент. Інтерфейс ILanguageDetector має тільки одне поле-функцію DetectLanguage, яка приймає на вхід текстове повідомлення і повертає ключ відповідної мови.

Розглянемо стандартну реалізацію цього елементу. Стандартною реалізацією цього інтерфейсу є клас LanguageDetector, він за допомогою моделі триграм, реалізація є відкритим кодом, який можна знайти тут [20], реалізовує визначення мови повідомлення, для цього є клас LanguageStatistics який зберігає в собі всі таблиці розподілу триграм залежно від мови, наприклад для української: " на", " за", "ння", "ня ", "на ", " пр", "ого", і тд. Це основні закінчення, суфікси і вставні слова що притаманні мові, при створенні цього класу в середині себе він створює унікальні колекції моделей триграм (класи TrigramModel і Trigram), після чого в функції DetectLanguage відбувається визначення мови.

Спочатку нормалізується текст(очищується від цифр і декількох пробілів підряд):

```
//для визначення мови достатньо перших 4000 знаків
if (text.Length > MAX_LENGTH) text =
text.Substring(0,MAX_LENGTH);
// очистити від цифр і знаків пунктуації
text = Regex.Replace(text, "[\u0021-\u0040„”|\\-]", "");
//очистити від зайвих пробілів
text = text.Replace("\t", " ").Replace("\n", " ")
.Replace("\r","");
text = text.Replace("  ", " ").Replace("  ", " ").Replace("  ",
" ").Replace("  ", " ");
return Identify(text.Trim());}
```

далі текст перевіряється на принадлежність до мови за допомогою спец символів відповідних видів алфавітів (кирилиця, хірагана і тд.), за допомогою регулярних виразів ("Basic Latin "[\u0000-\u007F]"):

```
Hashtable relevant_runs = new Hashtable();

for (int x = 0; x < unicodeBlockTests.Length; x+=2)
{
string name = unicodeBlockTests[x];
string regex = unicodeBlockTests[x + 1];
if (RegexCache[name] == null) RegexCache[name] =
new Regex(regex);
int charCount =
((Regex)RegexCache[name]).Matches(text).Count;
```

```
double pct = (double)charCount /(double)text.Length;
relevant_runs[name] = pct;
}
return relevant_runs;
```

Тоді будується модель триграму по тексту і вимірюючи дистанцію між моделлю тексту і моделлю для кожної мови вибирається найближча мова і повертається її ключ:

```
TrigramModel model = CreateOrderedModel(sample);
            int lowestScore = Int32.MaxValue;
            string lowestScoreLanCode = null;
            for (int i = 0; i < langs.Length; i++)
            {
                string lkey = langs[i].ToLower();
                if (langStat.Models[lkey] == null)
continue;//next please, no known model for this
                TrigramModel known_model =
(TrigramModel)langStat.Models[lkey];
                int dist = Distance(model, known_model);
                scores[lkey] = dist;
                if (dist< lowestScore)
                {
                    lowestScore = dist;
                    lowestScoreLanCode = lkey;
                }
            }
            return lowestScoreLanCode;
```

Наступним елементом є інтерфейс IStopWordsProvider, що відповідає за визначення стоп слів для мови за її ключем, його можна перевизначити передавши з прикладної частини у конструктор MessageClassificator, якщо цього не зроблено – буде використано стандартний елемент. Інтерфейс IStopWordsProvider має тільки одне поле-значення колекцію стоп-слів LanguagesStopWords, яка являє собою словник з ключем-ідентифікатором мови і значенням – колекцією стоп-слів.

Розглянемо стандартну реалізацію цього елементу. Стандартною реалізацією цього інтерфейсу є клас StopWordsProvider, він володіє стандартним словником стоп-слів для 3х мов – української, російської і англійської які читаються з файлів наступним чином:

```
var eng = Regex.Split(Properties.Resources.StopWordsEng,
Environment.NewLine);
var rus = Regex.Split(Properties.Resources.StopWordsRus,
Environment.NewLine);
var ukr = Regex.Split(Properties.Resources.StopWordsUkr,
Environment.NewLine);
LanguagesStopWords = new Dictionary<string, List<string>>();
LanguagesStopWords.Add("en", eng.ToList());
LanguagesStopWords.Add("ru", rus.ToList());
LanguagesStopWords.Add("uk", ukr.ToList());
```

Стандартні стоп-слова знаходяться у файлі ресурсів.

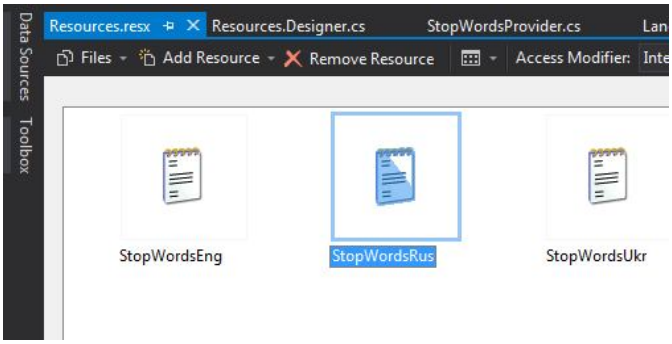

Рисунок 3.1 – Файли стандартних стоп-слів

Наступним елементом є інтерфейс ISynonymsDetector, що відповідає за визначення синонімів для слова, його можна перевизначити передавши з прикладної частини у конструктор MessageClassificator, якщо цього не зроблено – буде використано стандартний елемент. Інтерфейс ISynonymsDetector має тільки одне поле-функцію GetSynonyms, яка приймає на вхід ключ-ідентифікатор мови і слово, функція повертає колекцію з синонімів для вхідного слова.

Розглянемо стандартну реалізацію цього елементу. Стандартною реалізацією цього інтерфейсу є клас SynonymsDetector, він використовує COM об’єкт Word.Application. COM (Component Object Model) — платформа компонентно-орієнтованого програмування розроблена в 1993 році компанією Microsoft; дозволяє використання міжпроцесної взаємодії (inter-process communication) та динамічного створення об'єктів у будь-якій мові програмування, що підтримує технологію[21]. Використовується зазвичай у ОС Windows, хоча була реалізована і на інших платформах. Об'єкт Application – являє собою сам додаток Microsoft Word. Всі інші об'єкти Word'у знаходяться у ньому. Створити цей об'єкт - означає, запустити додаток Word на вашій машині. Для запуску Word з іншої програми необхідно додати у проект посилання на бібліотеку Microsoft Word 11.0 Object Library. Запустивши Word.Application з програмного коду, вікно Word'у не з’явиться тому, що за замовчуванням додаток Word запускається в прихованому вікні і якщо в ньому не відкрито жоден документ, він зразу же закривається (після то-го, як завершується процедура його створення). Оскільки об’єкт Word.Application є управляємим ресурсом, його потрібно звільнити після того як закінчили використання, тому SynonymsDetector наслідує інтерфейс IDisposable і в методі Dispose звільняє об’єкт Word.Application. Для пошуку синонімів використовується функція Word.Application. get_SynonymInfo:

```
Word.SynonymInfo synInfo = wordApp
.get_SynonymInfo(word,languageID);
 Array objSynInfo = synInfo.MeaningList as Array;
                if (objSynInfo != null)
                {
                   for (int synCount = 1;
                    synCount <= objSynInfo.Length; synCount++)
                    {
      searchList.Add(objSynInfo.GetValue(synCount).ToString());
                    }
                }
```

Наступним елементом є інтерфейс IStemmersProvider, що відповідає за відсікання суфіксів і префіксів у словах, добуття кореню слова, його можна перевизначити передавши з прикладної частини у конструктор MessageClassificator, якщо цього не зроблено – буде використано стандартний елемент. Інтерфейс IStemmersProvider має тільки одне поле-значення колекцію стеммерів LanguagesStemmers, яка являє собою словник з ключем-ідентифікатором мови і значенням – інтерфейсом IStemmer для відповідної мови. Інтерфейс IStemmer є інтерфейсом для реалізації стеммера для певної мови, інтерфейс має тільки одне поле-функцію Stem, яка приймає на вхід слово для стемінгу, і повертає слово що залишилось після процесу стемінгу.

Розглянемо стандартну реалізацію цього елементу. Стандартною реалізацією цього інтерфейсу є клас StemmersProvider, він володіє стандартним словником стеммерів для 3х мов – української, російської і англійської які відповідним чином реалізують інтерфейс IStemmer.

Для прикладу розглянем реалізацію інтерфейсу IStemmer для української мови. Реалізація основана на алгоритмі Стеммера Портера, що описаний у [16]. Спочатку визначаються усі закінчення, суфікси префікси які потрібно усунути у вигляді регулярного виразу, наприклад закінчення з голосних - `private const string NOUN = "(а|ев|ов|е|ями|ами|еи|и|ей|ой|ий|й|иям|ям|ием|ем|ам|ом|о|у|ах|иях|ях|і|ь|ию|ью|ю|ия|ья|я|і|ові|ї|ею|єю|ою|є|еві|ем|єм|ів|їв|\'ю)$";` далі починається процес стемінгу, крок за кроком слово очищається від закінчень, і інших неважливих морфологічних ознак:

```
wordToStem = wordToStem.ToLower();
            wordToStem = wordToStem.Replace("ё", "е");
            if (IsMatch(wordToStem, RVRE))
            {
                // Step 1
              if (!Replace(ref wordToStem,PERFECTIVEGROUND,"")){
                        Replace(ref wordToStem, REFLEXIVE, "");
```

```
                if (Replace(ref wordToStem, ADJECTIVE, ""))
                {
                    Replace(ref wordToStem, PARTICIPLE, "");
                }
                else
                {
                    if (!Replace(ref wordToStem, VERB, ""))
                    {
                        Replace(ref wordToStem, NOUN, "");
                    }
                }
            }
            // Step 2
            Replace(ref wordToStem, "и$", "");
            // Step 3
            if (IsMatch(wordToStem, DERIVATIONAL))
            {
                Replace(ref wordToStem, "ость?$", "");
            }
            // Step 4
            if (!Replace(ref wordToStem, "ь$", ""))
            {
                Replace(ref wordToStem, SUPERLATIVE, "");
                Replace(ref wordToStem, "нн$", "н");
            }
        }

        return wordToStem;
```

Для перевірки співпадінь і очищення використовуються регулярні вирази і їх клас парсер Regex:

```
private bool IsMatch(string word, string matchingPattern)
        {
            return new Regex(matchingPattern).IsMatch(word);
        }

        private bool Replace(ref string replace, string
cleaningPattern, string by)
        {
            string original = replace;
            replace = new Regex(cleaningPattern,
                        RegexOptions.ExplicitCapture |
                        RegexOptions.Singleline
                        ).Replace(replace, by);
            return original != replace;
        }
```

Регулярні вирази - це формальна мова пошуку і здійснення маніпуляцій з текстовими рядками, заснований на використанні метасимволів. Для пошуку використовується шаблони, що складаються з символів і метасимволів і задають правила пошуку. Для маніпуляцій з текстом додатково задається рядок заміни, який також може містити в собі спеціальні символи.

Після того як визначенні усі допоміжні елементи, можна приступити до кластеризації. Процес починається з класу DocumentInfoConstructor який приймає у своєму конструкторі усі допоміжні елементи і проводить очистку текстових даних і їхню нормалізацію через виклик функції ProcessDocumentCollection яка повертає колекцію документів класу DocumentInfo що містять такі дані:

- IUserData UserData об’єкт користувача для подальшої ідентифікації, якщо документ являє собою повідомлення то тут буде нульове значення
- string Content неочищений текст
- string LanguageKey ключ ідентифікатор мови
- List<string> Terms список очищених термінів
- float[] TermsVector вектор нормалізованих значень частоти термінів у документі

Процес отримання очищених даних відбувається наступним чином:

- Визначення мови повідомлення:

```
var messageLangKey = _languageDetector.DetectLanguage(_message);
```

- Отримання даних з дата провайдеру

```
var userData = _dataProvider.GetUserData();
```

- Відсіювання даних за мовою:

```
foreach (var data in userData)
            {
                foreach (var txt in data.Data)
                {
                    var txtLangKey =
                    _languageDetector.DetectLanguage(txt);
                    if (txtLangKey==messageLangKey)
                    {
                       _documentCollection.Add(newDocumentInfo()
                         { Content = txt, UserData = data,
                         LanguageKey = messageLangKey });
```

- Поділ документу на терміни за допомогою регулярного виразу:

```
Regex _wordSplitRegex = new Regex("([ \\t{}()\",:;.!?+-– \n])");
_wordSplitRegex.Split(document.Content)
```

- Відсіювання стоп-слів

```
if (!_stopWordsProvider.LanguagesStopWords.ContainsKey
(messageLangKey) ||
!_stopWordsProvider.LanguagesStopWords[messageLangKey]
.Contains(term.ToLower()))
```

- Відсіювання розділових знаків і цифр

```
  var removeList = new List<string>() { "\"", "\r", "\n", "(",
")", "[", "]", "{", "}", "", ".", " ", ",", "?", "!", "+", ":",
"*", "-", "–", "—" };
int n; bool isNumeric = int.TryParse(term, out n);
           if (!isNumeric && !removeList.Contains(term))
```

- Визначення синонімів і заповнення словника термінів

```
var synonyms =
_synonymsDetector.GetSynonyms(messageLangKey, term.ToLower());
synonyms.Add(term);
for (int i = 0; i < synonyms.Count; i++)
{
     synonyms[i] = stemmer != null ?
     stemmer.Stem(synonyms[i].ToLower()) :
     synonyms[i].ToLower();
}
if (!_distinctTerms.ContainsKey(term.ToLower()))
_distinctTerms.Add(term.ToLower(), synonyms);

document.Terms.Add(stemmer != null ?stemmer.Stem(term.ToLower())
: term.ToLower());
```

- Визначення вагових коефіцієнтів для кожного терміну:

```
foreach (var document in _documentCollection)
            {
                int count = 0;
                var space = new float[_distinctTerms.Count];
                foreach (var term in _distinctTerms.Values)
                {
                    space[count] = GetTF_IDF(document, term);
                    count++;
                }

                document.TermsVector = space;
            }
```

Функція для визначення ваги терміну виглядає так:

```
       private float GetTF_IDF(DocumentInfo document,
                                            List<string> term)
        {
            float tf = GetTermFrequency(document, term);
            float idf = GetInverseDocumentFrequency(term);
            return tf*idf;
        }

        private float GetTermFrequency(DocumentInfo document,
List<string> termAndSynonyms)
        {
            return
document.Terms.Count(termAndSynonyms.Contains);
        }

        private float GetInverseDocumentFrequency(List<string>
termAndSynonyms)
        {
            int count = _documentCollection.ToArray().Count(s =>
s.Terms.Any(termAndSynonyms.Contains));

           return
(float) Math.Log(_documentCollection.Count()/(float) count);
        }
```

Далі приступаємо до самої кластеризації, для цього є статичний клас DocumentClusterer, який містить функцію PrepareDocumentCluster яка приймає на вхід кількість кластерів що потрібно утворити і колекцію нормалізованих даних класу DocumentInfo, функція повертає колекцію

кластерів(центроїдів) класу Centroid, що містить усі документи що до нього входять, відповідно до заданої кількості по завершенню кластеризації. Спочатку обираємо початкові документи що утворять початкові центри кластерів, обов'язково сюди мають входити документ повідомлення і найвіддаленіший від повідомлення документ, для цього використовується функція GetCentroidsIndexesSet, що повертає індекси документів що стануть центроїдами:

```
            var result = new HashSet<int>();
            var r = new Random();
            result.Add(0); //message
            result.Add(FindDistanceIndex(docs.GetRange(1,
docs.Count - 1), docs[0], true) + 1);//farthermost
            var max = k > docs.Count ? docs.Count : k;
            while (result.Count < max)
            {
                int pos = r.Next(0, docs.Count);
                result.Add(pos);

            }
            return result;
```

Наступним кроком за допомогою функції FindClosestClusterCenter, що приймає колекцію центроїдів і об'єкт документу, шукаєм найближчий центроїд до кожного документу і додаєм той документ до знайденого центроїда, близькість визначається за допомогою функції відстані, у нашому випадку це коефіцієнт Отиаи (cosine similarity):

```
//1=close 0=far
public static float GetCosineSimilarity(float[] vecA, float[]
vecB)        {
            var dotProduct = DotProduct(vecA, vecB);
            var magnitudeOfA = Magnitude(vecA);
            var magnitudeOfB = Magnitude(vecB);
            float result =
                        dotProduct/(magnitudeOfA*magnitudeOfB);

            if (float.IsNaN(result))
                return 0;
            else
                return (float) result;
        }
```

```
public static float DotProduct(float[] vecA, float[] vecB)
        {
            return vecA.Select((t, i) => (t*vecB[i])).Sum();
        }

        public static float Magnitude(float[] vector)
        {
            return (float)Math.Sqrt(DotProduct(vector, vector));
        }
```

Слідуючим кроком за допомогою функції CalculateMeans, що приймає колекцію центроїдів, перекалькульовуєм ваги центроїдів:

```
            for (int i = 0; i < center.Count(); i++)
            {
                if (center[i].ClusterDocuments.Count > 0)
                {
                    for (int j = 0; j <
      center[i].ClusterDocuments[0].TermsVector.Count(); j++)
                    {
                        var total = center[i].ClusterDocuments
                              .Sum(doc => doc.TermsVector[j]);

                        center[i].ClusterDocuments[0]
                        .TermsVector[j] =
                        total /center[i]
                        .ClusterDocuments.Count();
                    }
                }
            }

            return center;
```

Останнім кроком за допомогою функції CheckStopCriteria, що приймає останню колекцію центроїдів і нову колекцію центроїдів перевіряє чи відбулись якісь зміни у центроїдах, якщо відбулись то ми продовжуємо, якщо ні то це наша остаточна колекція центроїдів(кластерів) і ми її повертаєм на зовні як результат. Повний код для перевірки критерії зупинки відображено у додатку В.

Загальну структуру проекту зображено на рисунку 3.2 , а діаграму класів усього проекту зображено у додатку А.

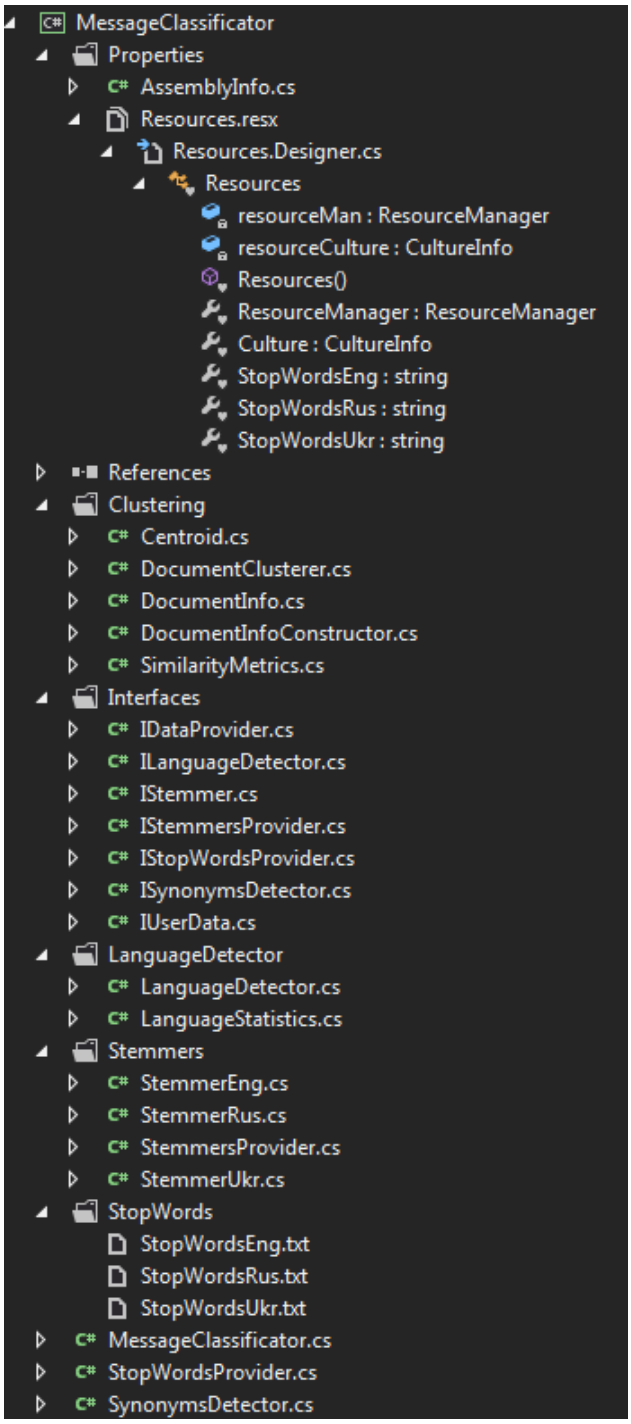

Рисунок 3.2 – Загальна структура модулю кластеризації

В результаті ми отримаєм кластери, що будуть репрезентувати результат кластеризації таким, що якщо документи входить у кластер з повідомленням то його користувачу буде цікаво отримати це повідомлення.

Наступним реалізуємо модуль надсилання. Основним компонентом даного модулю є інтерфейс IMessageSendingPlugin через який буде

відбуватись вся робота пов'язана з надсиланням. Інтерфейс IMessageSendingPlugin має поле-функцію GetPluginInfo яка повертає інформацію про плагін у вигляді об'єкту класу SendingModule і поле-функцію SendMessage, яка приймає на вхід повідомлення яке потрібно надіслати, інформацію про користувача, і тип тексту. Клас SendingModule має наступні поля : ім'я, тип модулю (емейл чи смс) і тип тексту що він підтримує.

Розглянемо для прикладу модуль для надсилання емейлів. Спочатку його потрібно зконфігурувати і задати параметри:

```
_server = config.AppSettings.Settings["Server"].Value;
_port = int.Parse(config.AppSettings.Settings["TcpPort"].Value);
_emailAddress = config.AppSettings.Settings["EmailAddress"]
                    .Value;
_password = config.AppSettings.Settings["Password"].Value;
_enableSsl = bool.Parse(
               config.AppSettings.Settings["EnableSsl"].Value);
```

Далі використовуючи простори імен System.Net і System.Net.Mail можна розсилати повідомлення:

```
public OperationResult SendMessage(string recipientInfo, string
subject, string message, TextType typeOfText)
        {
            try
            {
                var mail = new MailMessage
                {
                    From = new MailAddress(_emailAddress),
                    Subject = subject,
                    Body = message
                };
                if (typeOfText == TextType.Html)
                {
                    mail.IsBodyHtml = true;
                }
                mail.To.Add(new MailAddress(recipientInfo));
```

```
            var client = new SmtpClient(_server, _port)
            {
                EnableSsl = _enableSsl,
                Credentials = new
                NetworkCredential(_emailAddress, _password),
                DeliveryMethod = SmtpDeliveryMethod.Network
            };

            client.Send(mail);
        }
        catch (Exception exception)
        {
            return new
                OperationResult(false, exception.Message);
        }

        return new OperationResult(true, string.Empty);
    }
```

System.Net.Mail це простір імен який містить класи, що використовуються для відправки електронної пошти на сервер Simple Mail Transfer Protocol (SMTP) для доставки.

Повний код модулю SMS розсилки відображено у додатку Г.

Для загрузки модулів, для подальшого використання, буде використовуватись динамічний загружчик модулів, клас GenericPluginLoader в якому реалізована динамічна загрузка модуля залежно від його типу, код класу відображено у додатку Б.

3.2. Реалізація прикладної частини застосунка

Windows Forms дозволяє розробляти інтелектуальні клієнти. Інтелектуальний клієнт - це програма з повнофункціональним графічним інтерфейсом, проста в встановленні і оновленні, здатна працювати при наявності або відсутності підключення до Інтернету і використовує більш безпечний доступ до ресурсів на локальному комп'ютері в порівнянні з традиційними додатками Windows.

Windows Forms - це технологія інтелектуальних клієнтів для .NET Framework. Вона являє собою набір керованих бібліотек, що спрощують виконання стандартних завдань, таких як читання з файлової системи і запис в неї. За допомогою середовища розробки типу Visual Studio можна створювати інтелектуальні клієнтські програми Windows Forms, які відображають інформацію, дають можливість введення від користувачів і обмінюються даними з віддаленими комп'ютерами по мережі.

У Windows Forms форма - це видима поверхня, на якій виводиться інформація для користувача. Зазвичай додаток Windows Forms будується шляхом переміщення елементів управління на форму і написання коду для реагування на дії користувача, такі як правий і лівий клік миші або натискання клавіш на клавіатурі. Елемент управління - це окремий елемент користувацького інтерфейсу, який призначений для відображення або введення даних. При виконанні користувачем якої-небудь дії над формою або одним з її елементів управління створюється подія. Додаток реагує на ці події за допомогою коду і обробляє події при їх виникненні.

Windows Forms включає широкий набір елементів управління, які можна додавати на форми: текстові поля, кнопки, списки, що розкриваються, перемикачі та навіть веб-сторінки. Якщо існуючий елемент управління не задовільняє потреби, в Windows Forms можна створювати власні елементи управління за допомогою класу UserControl.

До складу Windows Forms входять багатофункціональні елементи призначені для користувацького інтерфейсу, що дозволяють відтворювати можливості таких складних додатків, як Microsoft Office. Використовуючи елементи управління ToolStrip і MenuStrip, можна створювати панелі інструментів і меню, що містять текст і малюнки, підменю і інші елементи управління, такі як текстові поля і поля зі списками.

За допомогою конструктора Windows Forms що підтримує перетягування, в Visual Studio можна легко створювати додатки Windows Forms. Досить виділити елемент керування курсором і помістити його в

потрібне місце на формі. Для подолання труднощів, пов'язаних з вирівнюванням елементів управління, конструктор надає такі елементи, як лінії сітки і лінії прив'язки, також ви можете використовувати елементи керування FlowLayoutPanel, TableLayoutPanel і SplitContainer для створення складних макетів форм за менший час.

В крайньому випадку, якщо потрібно створити свої власні елементи призначені для користувацького інтерфейсу, простір імен System.Drawing містить широкий набір класів, необхідних для відмальовування ліній, кіл та інших фігур безпосередньо на формі.

Для побудови графічного інтерфейсу користувача системи ми будемо використовувати більшість найчастіше вживаних елементів Windows Forms.

Опрацювавши усі сценарії із розділу 2.2 ми приступаємо до реалізації їх у прикладній частині системи.

Першим вікном що потрібно імплементувати це вікно входу у систему:

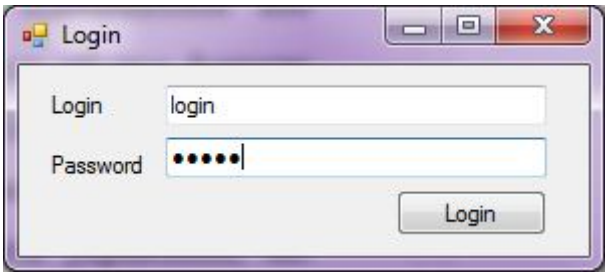

Рисунок 3.3 – Вікно авторизації

У вікні авторизації поле паролю спеціально закривається спец символами для безпеки, також сам пароль ніде не зберігається а зразу відправляється на перевірку. Щодо особливостей користувацького інтерфейсу то дане вікно і усі наступні появляються посередині екрану, для цього встановлено значення параметру StartPosition у значення перечислення FormStartPosition.CenterScreen.

Для редагування паролю розроблено вікно профілю:

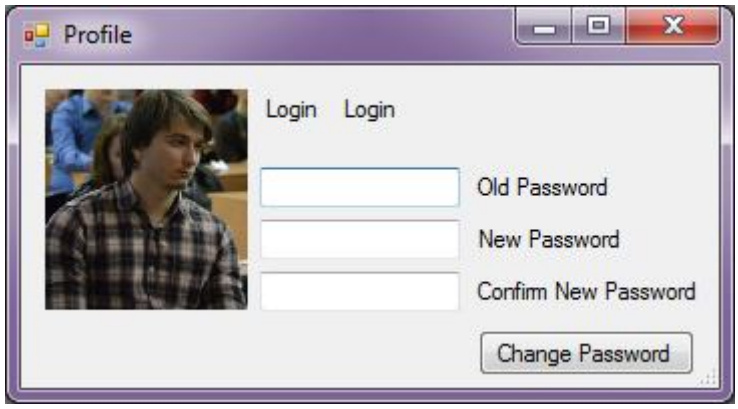

Рисунок 3.4 – Вікно профілю

На даному етапі вікно дозволяє лише змінити пароль, але у перспективі це вікно можна розширити і добавити нові функції.

Для перегляду стану модулів надсилань було розроблене наступне вікно:

Рисунок 3.5 – Вікно стану модулів надсилання

Контакти добавляти, редагувати і видаляти можна через наступне вікно:

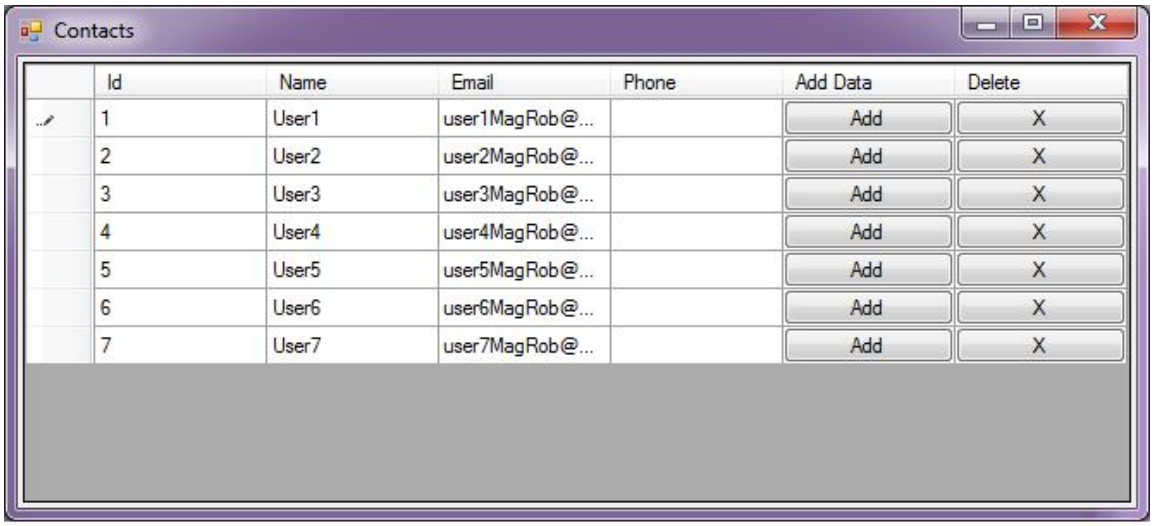

Рисунок 3.6 – Вікно редагування контактів

Список контактів виконаний через елемент Windows Forms – DataGridView, який прив’язаний до колекції контактів, редагування відбувається зразу таблиці, завдяки чудовій системі подій у Windows Forms усі зміни зберігаються зразу після редагування.

Надсилання відбувається у відповідному вікні:

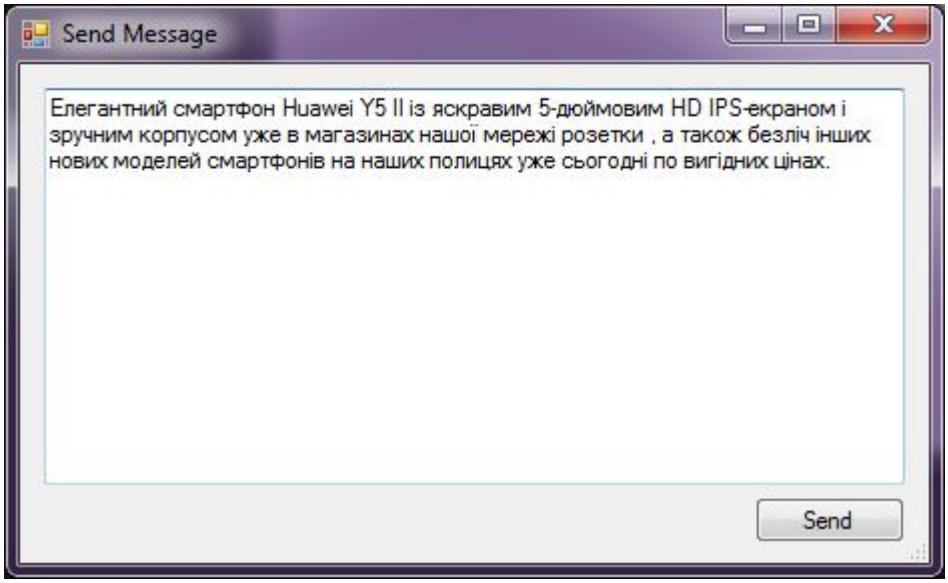

Рисунок 3.7 – Вікно надсилання нового повідомлення

Основним вікном програми у якому можна зразу переглянути стан надсилання, відмінити надсилання, переглянути звіти по надсиланнях є наступне:

Рисунок 3.8 – Головне вікно програми

З цього вікна можна попасти у попередньо описані модулі через кнопку меню – File. Тут також використано елемент Windows Forms – DataGridView для відображення даних, тому що він є дуже зручним і дозволяє добавляти колонки з кнопками як елементи відображення даних, що дозволяє реалізувати перегляд звітів і відміну надсилання прямо з таблиці.

Наступним вікном є відображення звіту по надсиланню:

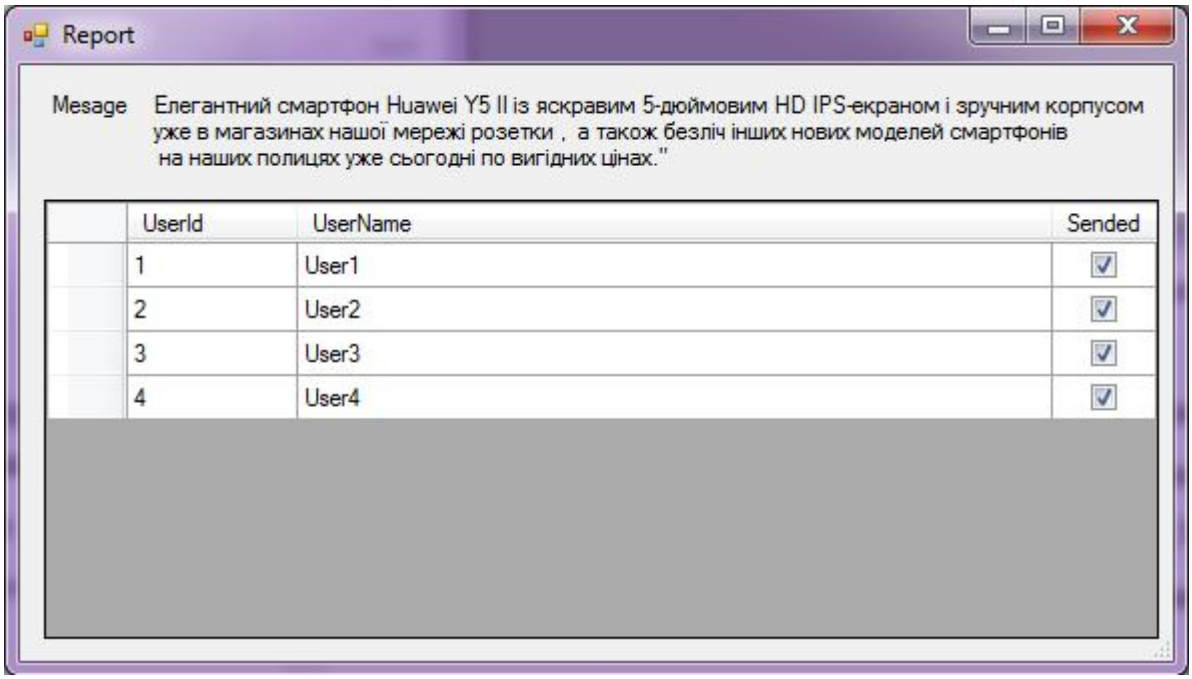

Рисунок 3.9 – Вікно звіту по надсиланню

Останнім, але не останнім по важливості, вікном є відображення звіту по кластеризації:

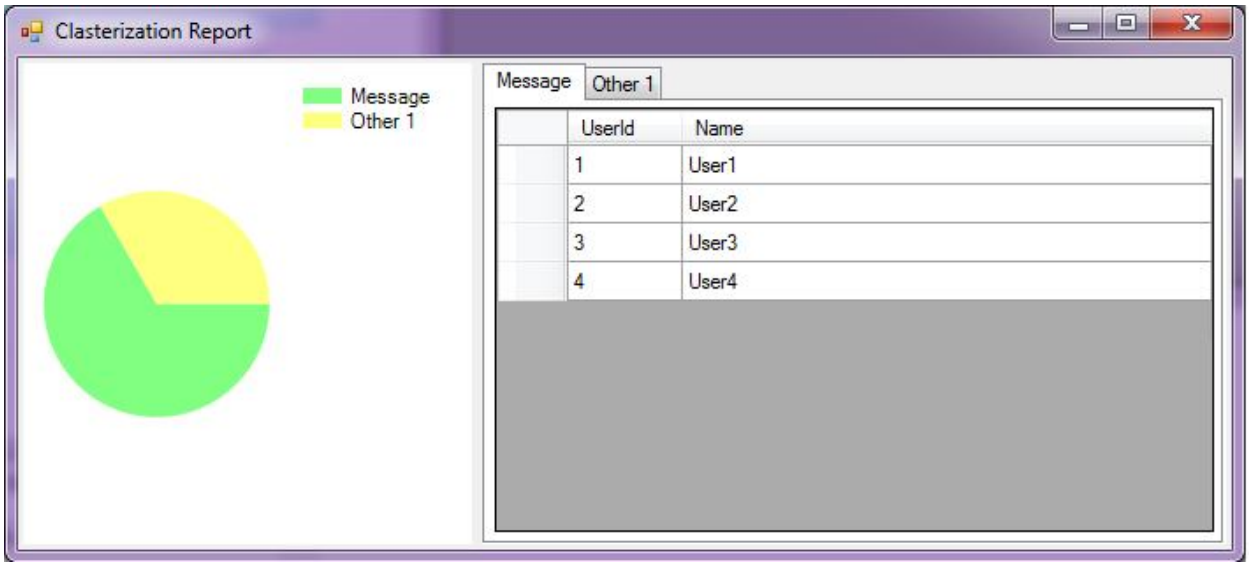

Рисунок 3.10 – Вікно звіту по надсиланню

Це вікно відображає кластери що утворились в результаті кластеризації, їх співвідношення між собою у вигляді кругової діаграми, що дає змогу наочно побачити як саме відбулась кластеризація, а також відображає користувачів які попали у відповідні кластери.

Висновки до розділу 3

У третьому розділі здійснено реалізацію основних частин системи розсилання електронної кореспонденції, а саме:

1. Проаналізувавши основні взаємодіючі елементи, побудовано модульно-інтерфейсну архітектуру, описано усі модулі системи і їх функціонал, складено діаграму класів основного модуля кластеризації і детально описано сам процес кластеризації з програмної точки зору.
2. Реалізацію прикладної частини системи для розсилання електронної кореспонденції реалізовано на мові програмування C# із використанням фреймворку .NET 4.0. Для з візуального представлення додатку було обрано технологію Windows Forms.

ВИСНОВКИ

У магістерській роботі вирішено проблему відсутності оптимізації і автоматизації процесу розсилання електронної кореспонденції. У процесі роботи було отримано наступні результати:

1. Проведено аналіз існуючих додатків для розсилання електронної кореспонденції.
2. Детально розглянуто та проаналізовано функціональні можливості існуючих додатків для розсилання електронної кореспонденції та виявлено недоліки, підхід до вирішення яких ми знайшли у даній роботі.
3. Проаналізовано основні проблеми, що виникають при інтелектуальному аналізі текстових даних.
4. На основі теоретичних міркувань розроблено алгоритмічне забезпечення для оптимізації процесу розсилання електронної кореспонденції.
5. Спроектовано структуру програмної системи для вирішення поставленої задачі.
6. Застосунок, розроблений із використанням C#/WinForms, було успішно впроваджено у робочий процес розробки програмного забезпечення.

ПЕРЕЛІК ПОСИЛАНЬ

1. **Softwareinsider.** Compare Email Marketing Services. *Softwareinsider.* [Електронний ресурс] http://email-marketing.softwareinsider.com/?utm_source=google&utm_medium=search.ad&utm_campaign=ao.sa.go.vse.softwareinsider.email-marke.

2. **Capterra.** Top Email Marketing Software Products. *Capterra* [Електронний ресурс] http://www.capterra.com/email-marketing-software/.

3. **MOLLY K. MCLAUGHLIN.** The Best Email Marketing Software of 2017. *Pcmag.*
[Електронний ресурс] http://www.pcmag.com/article2/0,2817,2453354,00.asp.

4. **werockyourweb.** Email Marketing Battle. *Werockyourwebs* [Електронний ресурс] https://www.werockyourweb.com/getresponse-vs-aweber-vs-mailchimp-vs-constant-contact-vs-vertical-response-vs-icontact/.

5. **Sally Jones.** MailChimp Reviews. *Werockyourwebs* [Електронний ресурс] https://www.werockyourweb.com/mailchimp-reviews.

6. **Sally Jones.** Campaign Monitor Review. *Werockyourwebs* [Електронний ресурс] https://www.werockyourweb.com/getresponse-vs-aweber-vs-mailchimp-vs-constant-contact-vs-vertical-response-vs-icontact.

7. **Sally Jones.** GetResponse Review. *Werockyourwebs* [Електронний ресурс] https://www.werockyourweb.com/get-response-review.

8. **Миронов Ю.Б., Крамар Р.М.** *Основи рекламної діяльності.* Дрогобич: Посвіт, 2007. – 108 с.

9. **Wikipedia.** Маркетинг. *Wikipedia* [Електронний ресурс] https://uk.wikipedia.org/wiki/Маркетинг.

10. **Wikipedia.** Кластерний аналіз. *Wikipedia* [Електронний ресурс] https://uk.wikipedia.org/wiki/Кластерний_аналіз.

11. **D. Cutting, D. Karger, J. Pedersen, J. Tukey.** A Cluster-based Approach to Browsing Large Document Collections. *semanticscholar* [Електронний ресурс]

https://pdfs.semanticscholar.org/1134/3448f8a817fa391e3a7897a95f975ad2873a.pdf

12. **Wikipedia.** Шумові слова. *Wikipedia* [Електронний ресурс] https://uk.wikipedia.org/wiki/Шумові_слова.

13. **Salim Roukos.** Trigram Language Model. *Survey of the State of the Art in Human Language Technology* [Електронний ресурс] http://www.cslu.ogi.edu/HLTsurvey/ch1node41.html.

14. **Vladlen.** Використання триграм для корекції результатів розпізнавання. *it-ua.info* [Електронний ресурс] http://it-ua.info/news/2016/08/25/vikoristannya-trigram-dlya-korekc-rezultatv-rozpznavannya.html.

15. **Wikipedia.** Стеммер Портера. *Wikipedia* [Електронний ресурс] https://ru.wikipedia.org/wiki/Стеммер_Портера.

16. **snowball.tartarus.** Snowball. *snowball.tartarus* [Електронний ресурс] http://snowball.tartarus.org/.

17. Charu C. Aggarwal, ChengXiang Zhai A SURVEY OF TEXT CLUSTERING ALGORITHMS *MINING TEXT DATA.* Kluwer Academic Publishers Boston/Dordrecht/London.

18. **Wikipedia.** Кластеризація методом к–середніх. *Wikipedia* [Електронний ресурс] https://uk.wikipedia.org/wiki/Кластеризація_методом_к–середніх.

19. **Wikipedia.** Модульное программирование.*Wikipedia*[Електронний ресурс] https://ru.wikipedia.org/ wiki /Модульное_программирование.

20. **Sasvári Tamás**. LanguageDetector class (C#). *idsyst.hu* . [Електронний ресурс] http://idsyst.hu/development/language_detector.html.

21. **Wikipedia.** Component Object Model. *Wikipedia* [Електронний ресурс] https://uk.wikipedia.org/wiki/Component_Object_Model.