НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»
Факультет інформатики та обчислювальної техніки
кафедра обчислювальної техніки

До захисту допущено:
Завідувач кафедри
_____ Сергій, СТІРЕНКО
«___»_____20__ р.

Дипломний проєкт

на здобуття ступеня бакалавра
за освітньо-професійною програмою «Інженерія програмного забезпечення комп'ютерних систем»
спеціальності 121 «Інженерія програмного забезпечення»
на тему: «Система аналізу емоційної оцінки мультимедійного контенту»

Виконав (-ла):
студент (-ка) IV курсу, групи ІП-64
Євтушенко Ілля Ігорович _____
Керівник:
Професор, доктор фізико-математичних наук
Гордієнко Юрій Григорович _____
Консультант з нормконтролю:
Професор, доктор технічних наук
Сімоненко Валерій Павлович _____
Рецензент:
Доцент, кандидат технічних наук
Писаренко Андрій Володимирович _____

Засвідчую, що у цьому дипломному проєкті немає запозичень з праць інших авторів без відповідних посилань.
Студент (-ка) _____

Київ – 2020 року

ВІДОМІСТЬ ДИПЛОМНОГО ПРОЄКТУ

№ з/п	Формат	Позначення	Найменування	Кількість листів	Примітка
1	А4		Завдання на дипломний проєкт	2	
2	А4	ДП 6404. 00.001 ВП	Відомість дипломного проєкту	1	
3	А4	ДП 6404. 00.002 ТЗ	Технічне завдання	3	
4	А4	ДП 6404. 00.003 ПЗ	Пояснювальна записка	61	
5	А4	ДП 6404. 00.004 Д1	Принципова схема системи	1	
6	А4	ДП 6404. 00.005 Д2	Структурна схема системи	1	
7	А4	ДП 6404. 00.006 Д3	Функціональна схема системи	1	

Змн.	Арк.	№ докум.	Підпис	Дата	ДП 6406. 00.001 ТЗ		
Розроб.		Євтушенко І. І.			Система аналізу емоційної оцінки мультимедійного контенту	Літ.	Арк. 1 / Аркушів 4
Перевір.		Гордієнко Ю.Г.					
Реценз.						КПІ ім. Ігоря Сікорського, Каф. ОТ, ІП-	
Н. Контр.		Сімоненко В. П.			Відомість дипломного проєкту		
Затверд.		Стіренко С. Г.					

Національний технічний університет України
«Київський політехнічний інститут»

Інститут (факультет) ___інформатики та обчислювальної техніки_____
(повна назва)

Кафедра _____обчислювальної техніки_____
(повна назва)

Рівень вищої освіти – перший (бакалаврський)

Спеціальність _____121 «Інженерія програмного забезпечення»___
(повна назва)

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____Сергій, СТІРЕНКО
"___" _____ 2020 року

ЗАВДАННЯ
на дипломний проєкт студента

_____Ілля ЄВТУШЕНКО_____
(ім'я, прізвище)

1. Тема проєкту *«Система аналізу емоційної оцінки мультимедійного контенту»*
керівник проєкту _____*д.ф.-м.н., проф., Юрій ГОРДІЄНКО*_____,
(ім'я, прізвище, науковий ступінь, вчене звання)
затверджені наказом вищого навчального закладу від "___" _____ 20__ року N ___

2. Термін подання студентом проєкту
_____

3. Вихідні дані до проєкту технічна документація, теоретичні дані, програмний код

4. Зміст пояснювальної записки
– опис предметної області, опис алгоритмів емоційного аналізу, реалізація запропонованого рішення

5. Перелік графічного матеріалу (із зазначенням обов'язкових креслеників, плакатів, презентацій тощо)

6. Консультанти розділів проєкту

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
1	*д.ф.-м.н., проф., Гордієнко Ю.Г.*	*10.03.2020*	*31.05.2020*
2	*д.ф.-м.н., проф., Гордієнко Ю.Г.*	*10.03.2020*	*31.05.2020*
3	*д.ф.-м.н., проф., Гордієнко Ю.Г.*	*10.03.2020*	*31.05.2020*
4	*д.ф.-м.н., проф., Гордієнко Ю.Г.*	*10.03.2020*	*31.05.2020*

7. Дата видачі завдання *15.12.2019*_____

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломного проекту	Строк виконання етапів проекту	Примітка
1	*Затвердження теми роботи*	*15.12.2019*	
2	*Вивчення та аналіз завдання*	*20.12.2019-02.02.2020*	
3	*Проведення огляду засобів та методів збору даних ЕЕГ*	*02.02.2020-01.03.2020*	
4	*Проведення аналізу засобів для розробки програмного забезпечення*	*01.03.2020-15.04.2020*	
5	*Розробка програмної реалізації*	*15.04.2020-13.05.2020*	
6	*Проведення моделювання та аналізу розробленої системи*	*13.05.2020-19.05.2020*	
7	*Оформлення матеріалів роботи*	*19.05.2020-31.05.2020*	
8	*Передзахист*		
9	*Захист*		

Студент _____ Ілля ЄВТУШЕНКО
(підпис)

Керівник проекту (роботи) _____ Юрій ГОРДІЄНКО
(підпис)

Анотація

В бакалаврській *дипломній* роботі реалізована система емоційної оцінки мультимедійного контенту на основі бібліотеки Stanford Core NLP.

За допомогою цього Web-додатку користувач може проводити емоційний аналіз тексту, щоб визначити позитивний він чи негативний, а також знаходити в тексті ключові слова такі як Person (ім'я або прізвище), City (місто), Country (країна) тощо. Продукт створений на мові програмування Java у середовищі розробки IntelliJ Idea, з використанням Spring. Для візуалізації проекту використовується набір інструментів BootStrap.

Аннотация

В бакалаврской дипломной работе реализована система эмоциональной оценки мультимедийного контента на основе библиотеки Stanford Core NLP.

С помощью этого Web-приложения пользователь может проводить эмоциональный анализ текста, чтобы определить положительный он или отрицательный, а также находить в тексте ключевые слова такие как Person (имя или фамилия), City (город), Country (страна) и др. Продукт создан на языке программирования Java в среде разработки IntelliJ Idea, с использованием Spring. Для визуализации проекта используется набор инструментов BootStrap.

Annotation

In the bachelor's thesis, a system of emotional evaluation of multimedia content based on the Stanford Core NLP library is implemented.

Using this Web application, the user can conduct an emotional analysis of the text to determine whether it is positive or negative, as well as find keywords such as Person (name or surname), City (city), Country (country), etc. The product was created. in the Java programming language in the IntelliJ Idea development environment using Spring. To visualize the project, the BootStrap Toolbox is used.

Технічне завдання
до дипломного проекту

на тему: «Система аналізу емоційної оцінки мультимедійного контенту»

Київ – 2020

ЗМІСТ

1.НАЙМЕНУВАННЯ ТА ОБЛАСТЬ ЗАСТОСУВАННЯ

Найменування – система аналізу емоційної оцінки мультимедійного контенту.

Застосування – для визначення позитивний чи негативний відгук до фільму.

2. ПІДСТАВИ ДЛЯ РОЗРОБКИ

Підставою для розробки є завдання на виконання бакалаврської дипломної роботи, затверджене кафедрою обчислювальної техніки Національного технічного університету України «Київський політехнічний інститут» імені Ігоря Сікорського.

3. МЕТА ТА ПРИЗНАЧЕННЯ РОЗРОБКИ

Метою і основним призначенням розробки є розробка веб-додатку для аналізу емоційної оцінки мультимедійного контенту, в якій можна визначити позитивний чи негативний вігдук до фільму.

4. ДЖЕРЕЛА РОЗРОБКИ

Джерелами розробки є науково-технічна література, різноманітні публікації в інтернеті, бакалаврські роботи інших студентів, приклади, вимоги та методичні рекомендації щодо дипломної роботи**.**

5. ТЕХНІЧНІ ВИМОГИ

5.1. Вимоги до програмного продукту, що розробляється

Додаток, що розробляється, повинен:

1) Використовувати NLP - Natural Language Processing (Обробка Природних Мов) бібліотеку для аналізу тональності тексту.

2) Можливість ввести текст, щоб дізнатися позитивний він або негативний

3) Можливість Розпізнавання іменованих сутностей (NER - Named Entity Recognition)

5.2 Вимоги до мов програмування

Мова програмування – Java. Середовище розробки – IntelliJ Idea. Відображення – BootStrap. З використанням Spring фреймворку, а також бібліотеки Stnford Core NLP.

5.3 Вимоги до програмного забезпечення

Використовуватися може будь-який комп'ютер, який підтримує web-браузер.

6. ЕТАПИ РОЗРОБКИ

Назва етапів виконання	Термін виконання
Затвердження теми роботи	*15.12.2019*
Огляд існуючої методології	*20.12.2019-01.03.2020*
Розробка теоретичної основи	*01.03.2020-15.04.2020*
Програмна реалізація	*15.04.2020-13.05.2020*
Оформлення документації	*13.05.2020-31.05.2020*

Пояснювальна записка
до дипломного проекту

на тему: «Система аналізу емоційної оцінки мультимедійного контенту»

Київ – 2020

Зміст

ВСТУП

В епоху науково-технічного прогресу інформація оточує нас усюди. Щосекунди виходить в середньому близько 6000 твітів, що відповідає понад 350 000 твітів, що надсилаються за хвилину, 500 мільйонів твітів на день і близько 200 мільярдів твітів на рік [9].

Аналіз тональності тексту - це процес визначення того, чи є написання позитивним, негативним чи нейтральним. Система аналізу настроїв для аналізу тексту поєднує в собі NLP(Natural Language Processing), обробку природних мов та методи Machine Learning, машинного навчання, щоб присвоїти зваженим балам настроїв сутність, теми та категорії в реченні чи фразі [2].

По суті, Sentiment Analysis, аналіз тональності тексту - це аналіз почуттів (тобто емоцій, поглядів, думок тощо) за словами, використовуючи інструменти обробки природних мов (NLP). Обробка природних мов по суті має на меті зрозуміти та створити природну мову за допомогою необхідних інструментів та прийомів [1].

Аналіз тональності тексту, також відомий як майнінг думок, відноситься до методів і процесів, які допомагають організаціям отримувати інформацію про те, як їх клієнтська база реагує на певний продукт чи послугу [3].

Ще аналіз тональності тексту використовує обробку природних мов та машинне навчання, щоб допомогти організаціям виглядати далеко за межі кількості лайків/акцій/коментарів, які вони отримують за рекламну кампанію, допис у блозі, випущений продукт чи щось подібне.

Аналіз тональності тексту допомагає аналітикам даних на великих підприємствах оцінювати громадську думку, проводити нюансовані дослідження ринку, контролювати репутацію бренду та товару та розуміти досвід клієнтів. Крім того, компанії з аналізу даних часто інтегрують сторонні API (Application Programming Interface), Прикладний програмний інтерфейс

аналізу почуттів у власне управління досвідом роботи з клієнтами, моніторинг соціальних медіа чи платформу аналітики робочої сили, щоб надавати корисну інформацію власним клієнтам.

Актуальність дипломного проекту полягає у необхідності вдосконалення та спрощення сучасних систем аналізу емоційної оцінки мультимедійного контенту з метою використання наданого результату для подальших висновків.

Метою виконання проекту є розробка системи аналізу емоційної оцінки мультимедійного контенту, яка об’єднає в собі зручність використання та простоту сприйняття інформації. А також надасть можливість проводити аналіз тональності тексту.

РОЗДІЛ 1. АНАЛІЗ ТОНАЛЬНОСТІ ТЕКСТУ

1.1 Базові поняття

Людська мова складна. Навчити машину аналізувати різні граматичні нюанси, культурні варіації, сленг та неправильні написання, які трапляються в онлайнових згадках, є складним процесом. Навчити машину розуміти, як контекст може впливати на тон, ще складніше.

Люди інтуїтивно розуміють, коли мова йде про інтерпретацію тони письмового тексту. Просто прочитавши публікацію, ви зможете визначити, чи автор позитивно чи негативно ставився до теми, але це при умові, якщо ви добре розбираєтесь у мові. Однак на комп'ютері немає поняття природньо розмовної мови - значить, нам потрібно розбити цю проблему на математику (мову комп'ютера). Він не може просто вивести, чи щось містить радість, розчарування, гнів чи інше - без будь-якого контексту того, що означають ці слова.

Аналіз почуттів вирішує цю проблему за допомогою обробки природних мов. В основному, він розпізнає необхідні ключові слова та фрази в документі, які в підсумку допомагають алгоритму класифікувати емоційний стан документа.

Основний аналіз настроїв текстових документів проходить просто [10]:

1. Розбиття кожного текстового документу на його складові частини (речення, фрази, лексеми та частини мови)
2. Визначення кожної фрази як компоненту, що несуть настрої
3. Призначення оцінки настрою кожній фразі та компоненту (від -1 до +1)
4. Необов'язково: комбінування балів для багатошарового аналізу настроїв

Вчені та програмісти даних пишуть програми, які подають документи в алгоритм і зберігають результати таким чином, щоб клієнти були корисними для використання та розуміння [14].

Вивчення ключових слів є однією з найпростіших методик і широко використовується алгоритмами Sentiment Analysis. Поданий вхідний документ ретельно сканується на предмет очевидних позитивних та негативних слів, таких як “sad - сумно", " happy - щасливо", " disappoint - розчарувати", " great - чудово", " satisfied - задовільно" тощо [13].

Існує ряд алгоритмів аналізу тональності тексту, і кожен має різні бібліотеки слів і фраз, які вони оцінюють як позитивні, негативні та нейтральні. Ці бібліотеки часто називають bag of words (мішком слів), багатьма алгоритмами.

Багато в тому, як ваш мозок запам'ятовує описові слова, з якими ви стикаєтесь протягом свого життя, та їх відносну «вагу настроїв», основна система аналізу настроїв спирається на бібліотеку настроїв, щоб зрозуміти фрази, що несуть почуття.

Бібліотеки емоційного тону - це дуже великі колекції прикметників (good - хороший, wonderful - чудовий, awful - жахливий, horrible - страшний) та фрази (good game - хороша гра,wonderful story - чудова історія, awful performance - жахливе виконання, horrible show - жахливе шоу), які були вручну забиті людськими програмістами. Цей підрахунок ручних настроїв є складним процесом, оскільки кожен, хто бере участь, повинен досягти певної згоди щодо того, наскільки сильний чи слабкий кожний бал повинен бути відносно інших балів. Якщо одна людина дає «поганий» бал настрою -0,5, а інша людина дає «жахливо» однаковий бал, ваша система аналізу настроїв зробить висновок, що обидва слова однаково негативні [5].

Більше того, багатомовний механізм аналізу настроїв повинен підтримувати унікальні бібліотеки для кожної мови, яку він підтримує. І кожну

з цих бібліотек потрібно постійно підтримувати: підганяти результати, додавати нові фрази, видаляти неактуальні фрази [11].

1.2 Типи аналізу тональності тексту

Щоб зрозуміти, як застосувати аналіз тональності тексту у контексті вашої бізнес-операції - потрібно зрозуміти різні його типи [12].

1-й тип. Fine-grained Sentiment Analysis (Дрібнозернистий аналіз) передбачає визначення полярності думки. Це може бути проста диференціація позитивних/негативних настроїв бінарних сфер. Цей тип також може входити до більш високої специфікації (наприклад, дуже позитивна, позитивна, нейтральна, негативна, дуже негативна), залежно від випадку використання (наприклад, як у п'ятизіркових відгуках про Amazon).

2-й тип. Emotion detection (Виявлення емоцій) використовується для виявлення ознак конкретних емоційних станів, представлених у тексті. Зазвичай існує комбінація лексиконів і алгоритмів машинного навчання, які визначають, що є що і навіщо.

3-й тип. Aspect-based sentiment analysis (Аналіз настроїв на основі аспектів) іде глибше. Його мета - виявлення думки щодо конкретного елемента виробу. Наприклад, яскравість ліхтарика в смартфоні. Аналіз на основі аспектів зазвичай використовується в аналітиці продуктів, щоб слідкувати за тим, як продукт сприймається та які сильні та слабкі сторони з точки зору клієнта.

4-й тип. Intent Analysis (Аналіз намірів) - це все про дію. Його мета - визначити, який намір виражається в повідомленні. Він зазвичай використовується в системах підтримки клієнтів для впорядкування робочого процесу.

1.3 Алгоритми аналізу тональності тексту

Є два основні методи аналізу тональності тексту [8].

1.3.1 Rule-based approach (Підхід на основі правил)

Аналіз настроїв на основі правил ґрунтується на алгоритмі з чітко визначеним описом думки для визначення. Включає виявлення суб'єктивності, полярності чи предмета думки.

Підхід, заснований на правилах, передбачає основну процедуру обробки природного мови. Він включає наступні операції з текстовим корпусом [9]:

- Language Identification (Ідентифікація мови)
- Tokenizaton (Токенізація)
- Sentence Breaking (Розрив речення)
- Stemming and Lemmatization (Стемінг та Лемматизація)
- Chunking (Розщеплення речення)
- Part of speech tagging (Розпізнавання частин мови)
- Syntax Parsing (Синтаксичний розбір)
- Sentence Chaining (Ланцюгування речення)
- Lexicon analysis (depending on the relevant context) (Аналіз лексикону (залежно від відповідного контексту))

Як це працює? Є два списки слів. Один з них включає лише позитивні, інший - негативні. Алгоритм проходить через текст, знаходить слова, які відповідають критеріям. Після цього алгоритм розраховує, який тип слів є більш поширеним у тексті. Якщо більше позитивних слів, то текст вважається позитивним.

Справа з алгоритмами, заснованими на правилах, полягає в тому, що хоча він дає якісь результати - йому не вистачає гнучкості та точності, які б зробили їх справді корисними. Наприклад, підхід на основі правил не враховує контекст. Однак його можна використовувати для загальних цілей визначення тону повідомлень, що може стати в нагоді для підтримки клієнтів [10].

У наші дні аналіз настроїв на основі правил зазвичай використовується

для створення основи для подальшої реалізації та навчання машинного рішення.

1.3.2 Automatic Sentiment Analysis (Автоматичний аналіз почуттів)

Хоча підхід, заснований на правилах, більше є іграшкою, ніж реальним інструментом, автоматизований аналіз настроїв - справжня угода. Це єдиний підхід, який справді копається в тексті та доставляє товар. Замість чітко визначених правил - цей тип аналізу настроїв використовує машинне навчання для з'ясування суті повідомлення.

Через це точність та точність операції різко зростають, і ви можете обробляти інформацію за численними критеріями, не надто складно.

По суті, автоматичний підхід передбачає керовані алгоритми класифікації машинного навчання. Насправді, аналіз настрою є одним із найскладніших прикладів того, як використовувати класифікацію для досягнення максимального ефекту. Крім цього, для дослідження даних використовуються непідтримувані алгоритми машинного навчання.

В цілому аналіз сентиментації може включати такі типи алгоритмів класифікації [6]:

- Linear Regression (Лінійна регресія)
- Support Vector Machines (Підтримка векторних машин
- RNN (Recurrent Neural Network), Рекурентна нейронна мережа включає в себе LSTM (Long Short-Term Memory), довга короткочасна пам'ять та GRU(Gated Recurrent Units) , Вентильний рекурентний вузол (Похідні RNN LSTM та GRU)
- Naive Bayes (Наївний Байєс)

1.4 Language Identification (Ідентифікація мови)

Першим кроком у аналітиці тексту є визначення мови, на якій написано

текст. Іспанською? Англійською? Арабською? Кожна мова має свої ідіосинкразії, тому важливо знати, з чим ми маємо справу.

Як основна, як може здатися, ідентифікація мови визначає весь процес для кожної іншої функції аналізу тексту. Тож дуже важливо правильно підібрати цю підфункцію.

1.5 Tokenizaton (Токенізація)

Знаючи якою мовою написаний текст, ми можемо розбити його на частини. Токени - це окремі одиниці значення, над якими ви працюєте. Це можуть бути слова, фонеми чи навіть повні речення. Токенізація - це процес розбиття текстового документа на частини [7].

У текстовій аналітиці лексеми - це найчастіше просто слова. Тоді речення з 10 слів міститиме 10 лексем. Однак для глибшої аналітики часто корисно розширити визначення свого маркера. Для лексалітиків токенами можуть бути:

- Слова
- Знаки пунктуації (знаки оклику посилюють настрої)
- Гіперпосилання (https: //…)
- Позитивні маркери (апострофи)

Токенізація залежить від мови, і кожна мова має свої вимоги до токенізації. Наприклад, англійська мова використовує пробіл та розділові знаки для позначення лексем, і токенізація порівняно проста.

Насправді, більшість алфавітних мов дотримуються відносно прямолінійних умов, щоб розділити слова, фрази та речення. Отже, для більшості алфавітних мов ми можемо покладатися на токенізацію на основі правил. Але не кожна мова використовує алфавіт.

У багатьох логографічних (на основі символів) мовах, таких як китайська, немає пробілів між словами. Токенізація цих мов вимагає

використання машинного навчання.

1.6 Sentence Breaking and Chunking (Розрив та розщеплення речення)

Після визначення мови текстового документа, токенізування та розбиття на речення, треба поставити теги.

Розпізнавання частин мови (або позначення PoS) - це процес визначення частини мови кожної лексеми в документі, а потім позначення її як такої.

Функції аналітики тексту, відома як Chunking відноситься до цілого ряду систем, що порушують речення, які розщеплюють речення на складові фрази (іменні фрази, дієслівні фрази тощо).

Позначення PoS означає віднесення частин мови до лексем.

Chunking означає присвоєння маркерів PoS до фраз.

1.7 Stemming and lemmatization (Стемінг та Лемматизація)

З граматичних причин в документах збираються використовувати різні форми слова, такі як organize, organizes, та organizing. Крім того, є сім'ї похідних споріднених слів із подібними значеннями, такі як democracy(демократія), democratic (демократичність), та democratization (демократизація). У багатьох ситуаціях здається, що для пошуку одного з цих слів було б корисно повернути документи, які містять інше слово в наборі [16].

Мета як стемінгу, так і лематизації - зменшити флективні форми.

am, are, is be

car, cars, car's, cars' car

Результат стемінгу або лематизації має бути таким:

the boy's cars are different colors

the boy car be differ color

Однак два слова відрізняються за своїм відтінком. Здебільшого йдеться про грубий евристичний процес, який рубає кінці слів з надією досягти цієї мети правильно більшу частину часу і часто включає видалення похідних

афіксів. Під лематизацією, як правило, йдеться про те, щоб правильно робити справи із використанням словника та морфологічного аналізу слів, як правило, метою є лише видалення флективних закінчень та повернення основної чи словникової форми слова, яка відома як лема. Якщо зіткнутися з лексемою saw, стемінг може повернутися лише s, тоді як лематизація намагатиметься повернути або see, або saw в залежності від того, чи використовується маркер як дієслово чи іменник. Вони також відрізняються тим, що найчастіше буває згортання похідних споріднених слів, тоді як лематизація зазвичай руйнує лише різні флективні форми лемми. Лінгвістична обробка для визначення або лематизації часто виконується додатковим компонентом плагіну до процесу індексації, і існує ряд таких компонентів, як комерційних, так і з відкритим кодом [9].

Найпоширеніший алгоритмом стеммінгу в англійській мові, який неодноразово виявлявся емпірично дуже ефективним, - це алгоритм Портера.

Алгоритм Портера складається з 5 фаз скорочення слів, що застосовуються послідовно. У межах кожної фази існують різні умови для вибору правил, наприклад, вибір правила з кожної групи правил, що застосовується до найдовшого суфіксу [4].

1.8 Part of speech tagging (розпізнавання частин мови)

Перед тим, як ви аналізувати речення та фразу на почуття, потрібно зрозуміти фрагменти, які його утворюють. Процес розбиття документа на його складові частини включає в себе кілька підфункцій, включаючи тегування частини мови (PoS)

Тегування частин мови - це процес ідентифікації структурних елементів текстового документа, таких як дієслова, іменники, прикметники та прислівники.

Більшість мов дотримуються деяких основних правил та зразків, які можна записати в комп'ютерну програму, щоб використовувати основний

розпізнавальник частин мови. Наприклад, англійською мовою цифра, за якою йде власне іменник, і слово “Street” найчастіше позначає адресу вулиці. Серія символів, перервана знаком @ і закінчується на ".com", ".net" або ".org", зазвичай представляє адресу електронної пошти. Навіть імена людей часто йдуть на узагальнені дво- або трисловні зразки іменників.

Іменники та займенники найчастіше представляють названі утворення, тоді як прикметники та прислівники зазвичай описують ці утворення в емоційному вираженні. Визначаючи поєднання прикметниково-іменникових, система аналізу настроїв набуває першої підказки, що вона дивиться на фразу, що несе настрої.

Точна частина мовлення тегів є критично важливою для надійного аналізу тональності тексту, тому важливо, щоб ці зміни змінювались на основі правил.

1.9 Syntax Parsing (Синтаксичний розбір)

Підфункція синтаксичного розбору є способом визначення структури речення. По правді, синтаксичний розбір - це просто фантазія для діаграми речень. Але це важливий підготовчий крок у аналізі настроїв та інших особливостях обробки природних мов.

Синтаксичний аналіз - це один з найбільш обчислювально-інтенсивних кроків у аналізі тексту [6].

1.10 Sentence Chaining (Ланцюгування речення)

Останнім кроком підготовки неструктурованого тексту для більш глибокого аналізу є ланцюгування речень, що іноді перекладають як відношення речення. Лексичний ланцюг пов'язує окремі речення за силою зв'язку кожного речення із загальною темою. Навіть якщо у реченнях з’являється багато абзаців у документі, лексичний ланцюг протікатиме через документ і допоможе машині виявити надмірні архівні теми та кількісно

оцінити загальне «відчуття».

Висновки до розділу 1

Аналіз тональності тексту - це надзвичайно цінна технологія для бізнесу, оскільки дозволяє отримувати реальні відгуки від ваших клієнтів неупереджено (або менш упереджено). При правильному підході, це може бути гарним придатком для ваших систем, додатків або веб-проектів.

Більш-менш кожен великий бренд в наші дні значною мірою покладається на прослуховування соціальних медіа, щоб покращити загальний досвід клієнтів.

РОЗДІЛ 2.
АНАЛІЗ ДОСТУПНИХ ТЕХНОЛОГІЙ ДЛЯ РЕАЛІЗАЦІЇ ТА РОЗРОБКИ СИСТЕМИ АНАЛІЗУ ЕМОЦІЙНОЇ ОЦІНКИ МУЛЬТИМЕДІЙНОГО КОНТЕНТУ

У сучасному аналізі текстових даних інструменти та бібліотеки NLP незамінні. Дослідники та підприємства використовують засоби обробки природних мов для отримання інформації з аналізу текстових даних. Цей аналіз включає аналіз відгуків клієнтів, автоматизацію систем підтримки, вдосконалення алгоритмів пошуку та рекомендацій та моніторинг соціальних медіа.

Доступний широкий спектр інструментів та служб NLP, і знання їх особливостей є ключовим для хороших результатів. Хоча деякі інструменти ідеально підходять для малих проектів, інші краще для експертів, які працюють над великими даними. Все залежить від проекту.

У пошуках ідеального рішення для свого дипломного проекту, я розібрав та проаналізував найкращі інструменти, бібліотеки та служби NLP.

2.1 NLTK

Інструментарій природних мов - це платформа для побудови програм Python для роботи з даними людської мови. Сюди входить лексичний аналіз, назва розпізнавання сутності, токенізація, маркування PoS, аналіз та семантичне міркування. Він також пропонує великі початкові ресурси. Однак, оскільки NLTK важкий для ресурсів при роботі з великими даними, він рекомендується для простих проектів.

Завдяки практичному посібнику, що представляє основи програмування разом із темами обчислювальної лінгвістики, а також вичерпною документацією API, NLTK підходить як для лінгвістів, інженерів, студентів, викладачів, дослідників, так і для користувачів галузі. NLTK доступний для

Windows, Mac OS X та Linux. Найкраще, що NLTK - це безкоштовний проект з відкритим кодом, керований громадою.

NLTK назвали «чудовим інструментом для навчання та роботи в обчислювальній лінгвістиці за допомогою Python» та «дивовижною бібліотекою для гри з природною мовою».

2.1.1 Що може NLTK?

- Токенізація та помітка (тег) тексту (рис. 2.1)

```
>>> import nltk
>>> sentence = """At eight o'clock on Thursday morning
... Arthur didn't feel very good."""
>>> tokens = nltk.word_tokenize(sentence)
>>> tokens
['At', 'eight', "o'clock", 'on', 'Thursday', 'morning',
'Arthur', 'did', "n't", 'feel', 'very', 'good', '.']
>>> tagged = nltk.pos_tag(tokens)
>>> tagged[0:6]
[('At', 'IN'), ('eight', 'CD'), ("o'clock", 'JJ'), ('on', 'IN'),
('Thursday', 'NNP'), ('morning', 'NN')]
```

Рисунок 2.1 Приклад токенізації

- Визначати назви утворень (рис 2.2)

```
>>> entities = nltk.chunk.ne_chunk(tagged)
>>> entities
Tree('S', [('At', 'IN'), ('eight', 'CD'), ("o'clock", 'JJ'),
           ('on', 'IN'), ('Thursday', 'NNP'), ('morning', 'NN'),
      Tree('PERSON', [('Arthur', 'NNP')]),
           ('did', 'VBD'), ("n't", 'RB'), ('feel', 'VB'),
           ('very', 'RB'), ('good', 'JJ'), ('.', '.')])
```

Рисунок 2.2 Приклад визначення назв утворень

- Показати абстрактне синтаксичне дерево

```
>>> from nltk.corpus import treebank
>>> t = treebank.parsed_sents('wsj_0001.mrg')[0]
>>> t.draw()
```

Рисунок 2.3 Приклад звернення до синтаксичного дерева

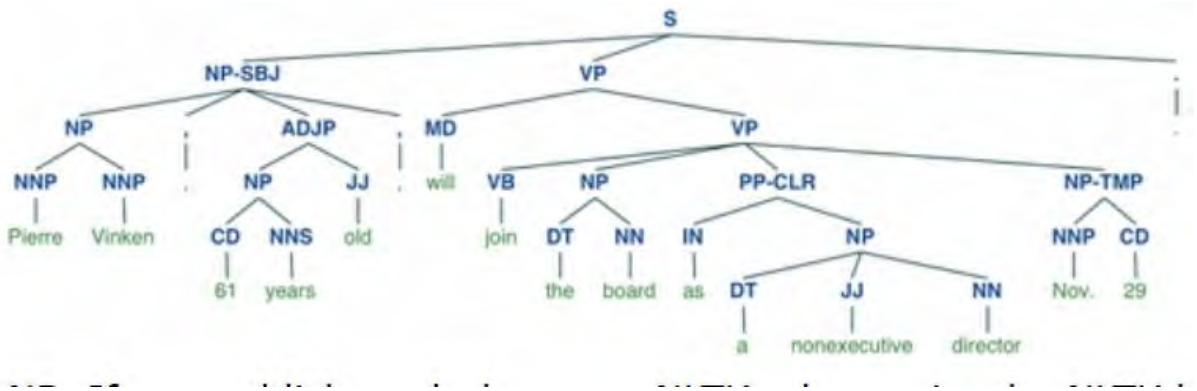


Рисунок 2.4 Абстрактне синтаксичне дерево

2.2 PyTorch-Transformers

Ця бібліотека містить заздалегідь підготовлені моделі для NLP. У ньому представлені реалізації PyTorch, попередньо підготовлені ваги моделей, сценарії використання та утиліти перетворення для моделей, включаючи BERT, GPT-2, Transformer-XL та RoBERTa.

В даний час бібліотека містить реалізацію PyTorch, попередньо підготовлені моделі, сценарії використання та утиліти перетворення для наступних моделей:

BERT (від Google), опублікований разом із документом BERT: попередня підготовка глибоких двонаправлених трансформаторів для розуміння мови Джейкобом Девліном, Мінг-Вей Чангом, Кентоном Лі та Крістіною Тутановою.

GPT (від OpenAI), опублікований разом із документом «Покращення

розуміння мови» шляхом генеральної попередньої підготовки Алека Радфорда, Картика Нарасимхана, Тіма Саліманса та Іллі Суцкевера.

GPT-2 (від OpenAI), випущений разом з мовою «Мовні статті», - це непідконтрольні багатозадачні студенти Алек Радфорд, Джефрі Ву, Ревон Чайлд, Девід Луан, Даріо Амодей та Ілля Суцкевер.

"Transformer-XL" (від Google / CMU), випущений разом із документом "Трансформер-XL: моделі уважного мовлення поза контекстом фіксованої довжини" Цзихан Дай, Жилін Ян, Йімін Янг, Хайме Карбонелл, Квок В. Ле, Руслан Салахутдінов.

XLNet (від Google / CMU), опублікований разом із документом XLNet: Узагальнений автоматичний пошук мови для розуміння мови Жилін Ян, Зіханг Дай, Імінг Ян, Хайме Карбонелл, Руслан Салахутдінов, Квок В. Ле.

XLM (від Facebook) випущений разом з паперовою мовною багатомовною системою пошуку Гійом Лампл та Алексісом Конно.

RoBERTa (від Facebook) випустив разом із документом надійно оптимізований підхід до пошуку BERT від Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, Veselin Stoyanov.

DistilBERT (від HuggingFace), випущений разом із поштовим блогом Менший, швидший, дешевший, легший: Представляємо DistilBERT, дистильовану версію BERT Віктора Санха, Лісандра Дебюта та Томаса Вольфа.

Доступні методи:

- config: повертає елемент конфігурації, що відповідає заданій моделі або pth.
- tokenizer: повертає токенізатор, що відповідає заданій моделі або шляху
- model: повертає модель, відповідну заданій моделі або шляху

- modelWithLMHead: повертає модель з даними для моделювання мови, що відповідає заданій моделі або шляху
- modelForSequenceClassification: повертає модель з класифікатором послідовностей, що відповідає заданій моделі або шляху
- modelForQuestionAnswering: повертає модель з головою для відповіді на питання, що відповідає зазначеній моделі або шляху

2.3 TextBlob

Побудований на плечах NLTK, TextBlob - це як розширення, яке спрощує багато функцій NLTK. Він пропонує простий для розуміння інтерфейс для завдань, включаючи аналіз настроїв, маркування PoS та вилучення фрази іменника. TextBlob - рекомендований інструмент для обробки природних мов для початківців, який також масштабується.

Властивості:

- Вилучення іменних словосполучень
- Визначення частин мови
- Аналіз тональності тексту
- Класифікація (Naive Bayes, дерево рішень)
- Токенізація (розбиття тексту на слова та речення)
- Частота слова та фрази
- Синтаксичний аналіз
- Словознавство (плюралізація та синуляризація) та лематизація
- Корекція орфографії
- Можливість додати нові моделі чи мови через розширення
- Інтеграція WordNet

2.4 SpaCy

SpaCy - це швидка та ефективна бібліотека з відкритим кодом, написана на Cython. Вона має простий API, заздалегідь підготовлені слова векторів, 23

статистичні моделі для 11 мов, вбудовані візуалізатори для синтаксису та NER та підтримку понад 53 мов. Її графік= оновлення також дуже послідовний.

Властивості:

- Неруйнівна токенізація
- Розпізнавання іменованих сутності
- Підтримка 55+ мов
- 17 статистичних моделей для 11 мов
- Дослідження вкладання слів
- Найсучасніша швидкість
- Легка інтеграція глибокого навчання
- Визначення частин мови
- Аналіз залежностей
- Синтаксична сегментація речень
- Вбудовані візуалізатори для синтаксису та NER
- Зручне відображення між рядками та хешами
- Експорт до масивів даних numpy
- Ефективна бінарна серіалізація
- Простота упаковки та розгортання моделі

2.5 Stanford CoreNLP

CoreNLP використовується для застосування лінгвістичного аналізу до фрагментів тексту. Він пропонує підтримку на 7 мовах, а його масштабованість робить його хорошим засобом обробки природних мов для скреблінгу інформації, навчання чат-ботів, а також обробки та створення тексту. При цьому він ліцензується за загальною публічною ліцензією GNU v3, тому комерційна ліцензія необхідна при створенні будь-якого власного програмного забезпечення.

Stanford CoreNLP має такі властивості:

- Інтегрований набір інструментів NLP з широким спектром інструментів граматичного аналізу
- Швидкий, надійний аннотатор для довільних текстів, широко використовуваний у виробництві
- Сучасний, регулярно оновлюваний пакет, із загальною якісною аналітикою тексту
- Підтримка ряду основних (людських) мов
- Доступні API для більшості основних сучасних мов програмування
- Можливість роботи як простого веб-сервісу

Мета Stanford CoreNLP полягає в тому, щоб дуже просто застосувати купу інструментів мовного аналізу тексту. Інструмент конвеєр може бути виконаний на простому тексті з лише двома рядками коду. CoreNLP розроблений таким чином, щоб бути дуже гнучким і розширюваним. За допомогою однієї опції ви можете змінити, які інструменти потрібно включити та відключити. Stanford CoreNLP інтегрує багато інструментів, включаючи POS (Part of Speech) (розпізнавання частин мови), іменований розпізнавальник іменованих сутностей (NER – Named Entity Recognition), аналізатор, систему роздільної здатності ядра, аналіз тональності тексту, вивчення завантаженого шаблону та відкриті засоби вилучення інформації. . Більше того, конвеєр анотаторів може включати додаткові спеціальні або сторонні анотатори. Аналізи CoreNLP надають основні будівельні блоки для програм для розуміння тексту вищого рівня та домену (рис. 2.5)

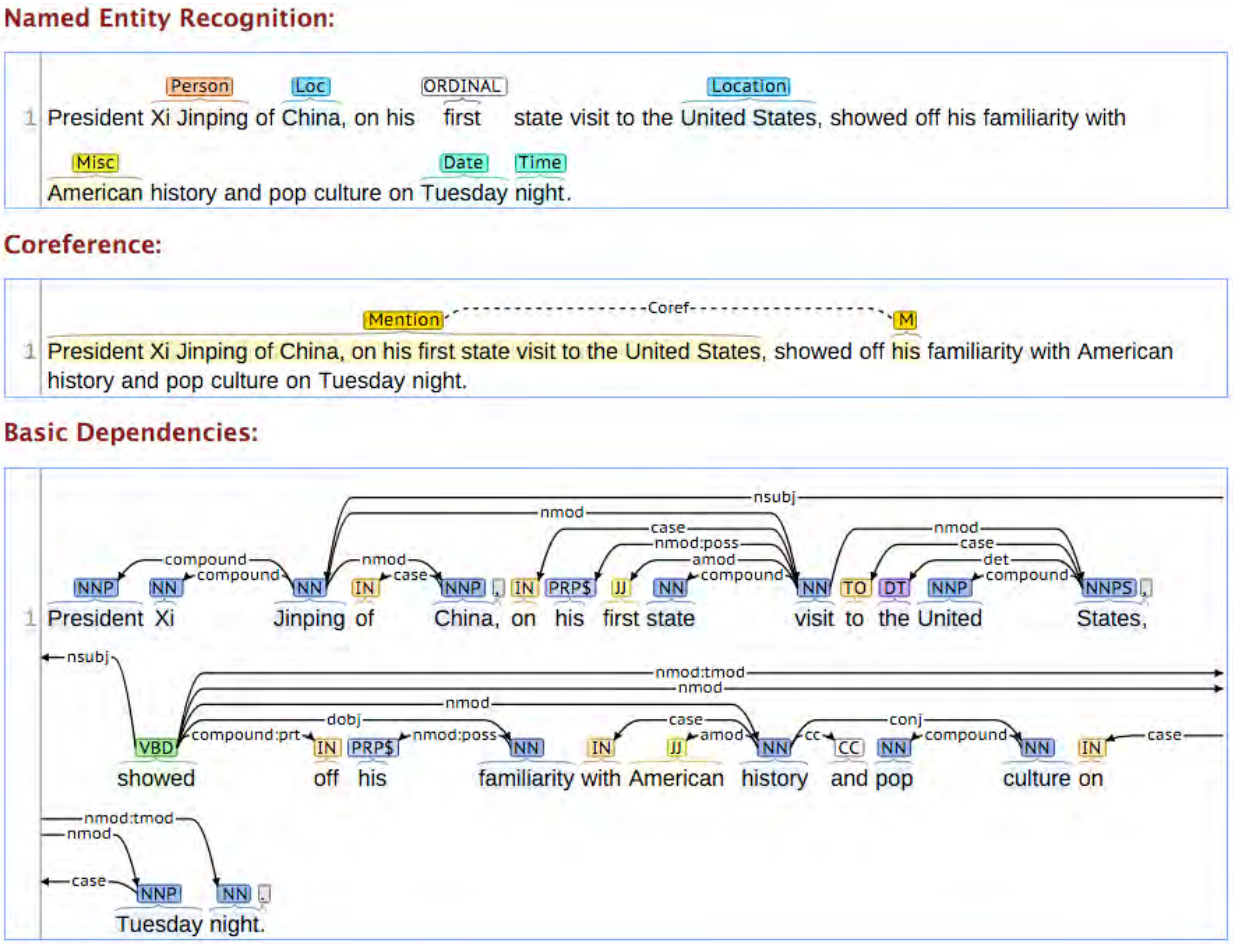


Рисунок 2.5 Аналіз бібліотеки Stanford CoreNLP

2.6 Apache OpenNLP

Ця бібліотека NLP, написана на Java, пишається своєю простотою. Вона включає токенізацію, сегментацію речень, розпізнавання частин мови, відбивання, аналіз та машинне навчання на основі перцептрона. Однак Apache - це проект, розроблений волонтерами, тому оновлення виходять не дуже часто.

Бібліотека Apache OpenNLP містить кілька компонентів, що дозволяє створити повний конвеєр обробки природних мов. До таких компонентів належать: детектор речень, токенізатор, пошук імен, категоризатор документів, розпізнавання частин мови, чункер, аналізатор, роздільна здатність основної мови. Компоненти містять частини, які дозволяють

виконувати відповідне завдання з обробки природних мов, тренувати модель, а також часто оцінювати модель. Кожен із цих засобів доступний через інтерфейс прикладної програми (API). Крім того, для зручності експериментів та занять передбачений інтерфейс командного рядка (CLI).

2.7 AllenNLP

Дослідницька бібліотека Apache 2.0, побудована на PyTorch, Allen NLP призначена для дослідників, які хочуть швидко та просто будувати моделі аналізу мови. Завдяки широкому спектру варіантів аналізу тексту, AllenNLP - це простий інструмент NLP, який також масштабується.

2.8 GenSim

Безкоштовна бібліотека Python для обробки природних мов, GenSim є рекомендованим варіантом для моделювання тем та порівняння подібності документів. Крім того, він також пропонує масштабовану статистичну семантику та аналіз семантичної структури. GenSim може похвалитися високою швидкістю обробки та здатністю обробляти велику кількість тексту.

Gensim включає потокові паралелізовані реалізації алгоритмів fastText, word2vec та doc2vec, а також прихований семантичний аналіз (LSA, LSI, SVD), невід'ємну матричну факторизацію (NMF), латентне розподілення Діріхле (LDA), tf -idf та випадкові прогнози.

Деякі з нових алгоритмів в Інтернеті в Gensim також були опубліковані в докторській дисертації 2011 р. «Масштабованість семантичного аналізу в обробці природними мовами» творця Gensim Радима Шерека

Gensim використовувався та цитується у понад 1400 комерційних та академічних заяв станом на 2018 рік, у різноманітному спектрі дисциплін від медицини до аналізу страхових претензій до пошуку патентів.

Код з відкритим кодом розробляється та розміщується на GitHub, а громадський форум підтримки підтримується в групах Google та Gitter.

Gensim комерційно підтримується компанією rare-technologies.com, яка також надає студентські інструктажі та проекти академічних дисертацій для Gensim через свою програму Student Incubator.

2.9 Architector NLP

Розроблений лабораторією Intel AI, архітектор NLP є бібліотекою Python з відкритим кодом для оптимізації NLP та вивчення глибоких топологій навчання. Він призначений для того, щоб зробити навчальні та бігові моделі простим процесом.

NLP Architect розроблений таким чином, щоб бути гнучким для додавання нових моделей, компонентів нейронної мережі, методів обробки даних та для легкого навчання та запуску моделей.

Особливості:

- Основні моделі NLP, які використовуються у багатьох завданнях NLP та корисні у багатьох програмах NLP
- Нові моделі NLU демонструють нові топології та методи
- Оптимізовані моделі NLP / NLU демонструють різні алгоритми оптимізації на нейронних моделях NLP / NLU
- Дизайн, орієнтований на модель
- Можливість тренувати та запускати моделі з командного рядка.
- API для використання моделей для висновку в python.
- Процедури для визначення спеціальних процесів для навчання, умовиводу чи будь-чого, що стосується обробки.
- Підсистема CLI для запущених процедур

Основні утиліти для роботи з моделями NLP - попередня обробка тексту рядків, введення-виведення даних, маніпуляція даними, метрики, вбудовування.

NLP Architect - бібліотека, орієнтована на модель, розроблена для

демонстрації нових та різних оптимізацій нейронної мережі. Бібліотека містить моделі, пов’язані з NLP / NLU за завданням, різні топології нейронної мережі (які використовуються в моделях), процедури спрощення робочих процесів у бібліотеці, заздалегідь визначені процесори даних та завантажувачі наборів даних та інші програми. Бібліотека розроблена як інструмент для розробки моделі: попередній процес обробки даних, побудова моделі, навчання, перевірка, висновок, збереження або завантаження моделі.

Основні рекомендації щодо дизайну:

- Агностичний фреймворк глибокого навчання
- Моделі NLP / NLU за завданням
- Різні реалізовані топології (модулі), які можна використовувати з моделями
- Цілісність програми (рішення), що використовують одну або кілька моделей NLP Architect
- Загальні завантажувачі наборів даних, утиліти для обробки текстових даних та різні утиліти, що підтримують розробку моделі NLP (завантажувачі, текстові процесори, іо, метри ки тощо)
- Процедури визначення процесів навчання, умовиводу, оптимізації або будь-якого складного сценарію.
- Pythonic API для використання моделей для висновку
- Обширна модельна документація та навчальні посібники

Висновок до розділу 2

В даному розділі було проаналізовано доступні NLP-бібліотеки з можливістю аналізу тональності тексту. Було детально розглянуто можливі підходи виконання поставленої задачі. Було зроблено висновок, що аналогічних бібліотек з потрібним функціоналом досить багато. Було прийняте рішення використовувати бібліотеку Stanford CoreNLP, написану мовою Java. Вона включає в себе весь потрібний функціонал та має покращені алгоритми аналізу тональності тексту.

РОЗДІЛ 3. ПРОЕКТУВАННЯ І РОЗРОБКА ДОДАТКУ

3.1 Використані інструменти розробки

Реалізація веб-додатку для аналізу емоційної оцінки мультимедійного контенту використовує наступні інструменти:

Мова програмування: Java. Java – це об’єктно-орієнтована мова програмування. Програмний код написаний мовою Java компілюється в байт-код, який інтерпретує віртуальна машина для потрібної платформи. Являється однією з найпоширеніших мов програмування у світі.

Spring Framework – це прикладна основа та інверсія контейнера управління для платформи Java. Основні функції рамки можуть використовуватися будь-яким додатком Java, але є розширення для побудови веб-додатків на платформі Java EE (Enterprise Edition). Хоча рамка не нав'язує жодної конкретної моделі програмування, вона стала популярною у спільноті Java як доповнення або навіть заміна моделі Enterprise JavaBeans (EJB). Spring Framework є відкритим кодом.

Бібліотека Stanford CoreNLP. По суті це набір інструментів та технологій людської мови. Він може визначити основні форми слів, їх частини мови, будь то назви компаній, людей тощо, нормалізувати дати, час та числові величини, визначити структуру речень з точки зору словосполучень та синтаксичних залежностей, вказати які іменникові фрази стосуються одних і тих же утворень, визначити настрій тексту, витягнути конкретні або відкриті відносини між згадками суб'єкта, отримувати цитати, про які говорять люди тощо.

Bootstrap – інструментарій з відкритим кодом, що допоможе швидко розробляти та налаштовувати веб-сайти для мобільних пристроїв. Містить змінні міксини та Sass, чутлива система сітки, широкі попередньо вбудовані компоненти та потужні додатки JavaScript.

3.2 Опис інструментів розробки

Spring Framework

Основна мета фреймворку Spring – полегшити створення Java Enterprise додатків та надати все необхідне для взаємодії мови програмування Java та enterprise середовищем. Поточна версія фреймворку Spring 5.1+ вимагає встановлену Java 1.8 або вище. Фреймворк Spring розділений на модулі. Програма сама визначаю та підключає модулі, які їй потрібні. Я використовую деякі з цих модулів.

Spring Boot – модуль бібліотеки Spring, що використовується для спрощення процесу створення автономних Spring програм на Java.

Основні функції:

- Створення автономних програм на Spring
- Автоматичне налаштування можливих бібліотек
- Спрощення конфігурації збірки за рахунок забезпечення впевнених залежностей від «стартера»
- Безпосереднє використання Tomcat, Undertow або Jetty (не розгортаючи файли WAR)
- Відсутня генерація коду. Не вимагає XML конфігурації.

Spring Web MVC – веб фреймворк побудований на Servlet API. Забезпечує архітектуру патерна MVC (Model – View – Controller (Модель-Відобаження-Контроллер).

- Model (Модель) – інкапсулює дані програми, які будуть у виді Java об’єктів або бінів.
- View (Відображення) – генерує HTML код, що відповідає за відображення даних Моделі.
- Controller (Контроллер) – створює відповідну модель, оброблючи запрос користувача, щоб передати її для виду у Відображенні.

Stanford Core NLP

Stanford CoreNLP – бібліотека інструментарій, заснований на використанні обробки природньої мови.

Основні переваги бібліотеки Stanford CoreNLP:

- Широкий спектр інструментів граматичного аналізу з Інтегрованим набором інструментів NLP.
- Підтримка ряду людських мов
- Надійний, швидкий аннотатор для довільних текстів, використовуваний широко у виробництві
- Можливість роботи як простого веб-сервісу
- Сучасний пакет, із загальною якісною аналітикою тексту. Регулярно оновлюється
- Доступні API для більшості основних сучасних мов програмування

Бібліотека Stanford CoreNLP має багато цікавих інструментів для обробки природньої мови такі як зчитування частин мови, розпізнавання іменованих сутностей, парсер, визначення кореферентності, аналіз емоційності тексту, завантажувальні шаблони для вивчення та відкриті засоби вилучення інформації. А також анотаторний конвеєр, який може включати в себе додаткові анотатори.

Базовий дистрибутив забезпечує покращений переклад с англійської мови, але двигун сумісний також з іншими мовами та наразі підтримує арабську, французьку, китайську, іспанську та німецьку мови.

Stanford CoreNLP написаний на об’єктно орієнтованій мові програмування Java. Поточна версія вимагає встановити Java 1.8+ для взаємодії. Хоча можна використовувати CoreNLP через консоль, тому цю бібліотеку використовують програмісти на Javascript та Python або інших мовах програмування.

Bootstrap

Bootstrap – це найрозповсюдженіший фреймворк для CSS для створення веб-додатків, що будуть адаптувати відображення на мобільні версії або версії для планшету. Основні функції:

- Контейнери. Надзвичайно важливий клас. Забезпечує нові можливости для елементів HTML, а саме поля, підкладки, вирівнювання та багато чого іншого.
- Колір фону. Дозволяють змінити колір фона (рис. 3.1).

Рисунок 3.1 Варіанти кольорів фону у бібліотеці Bootstrap

- Колір тексту. Дозволяє змінювати колір тексту (рис. 3.2).

Рисунок 3.2 Варіанти кольору тексту у бібліотеці Bootstrap

- Колонки. Однакова ширина на всіх пристроях та при будь-якій ширині екрана.
- Таблиці. Дозволяють відображення даних у таблицях (рис. 3.3).

Firstname	Lastname
John	Doe
Mary	Moe
July	Dooley

Рисунок 3.3 Вигляд таблиці у бібліотеці Bootstrap

- Попереджувальні секції (рис. 3.4)

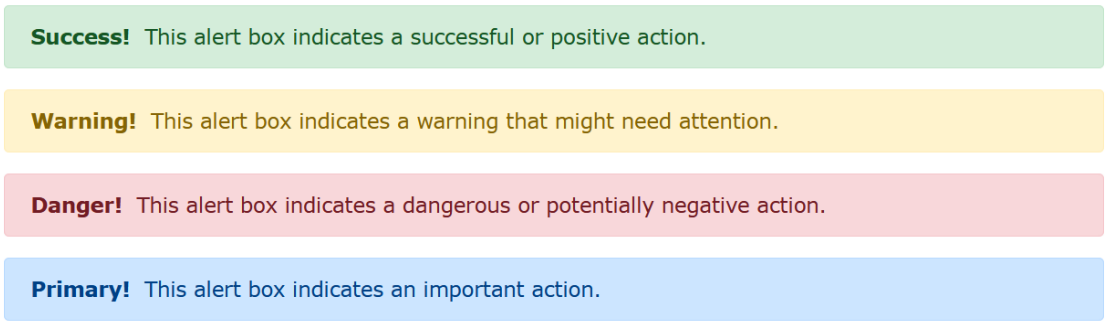


Рисунок 3.4 Вигляд попереджувальних секцій у бібліотеці Bootstrap

- Кнопки (рис. 3.5)

Рисунок 3.5 Вигляд кнопок у бібліотеці Bootstrap

3.3 Реалізація веб-додатку для аналізу емоційної оцінки мультимедійного контенту

Реалізація дипломної роботи мала на увазі створення веб-додатку, що

мав би змогу розпізнавати «емоційний відтінок» тексту на прикладі відгуків на кінофільми використовуючи обрані мною технології. Написаний програмний код можна зобразити за допомогою блок-схеми (рис. 3.6):

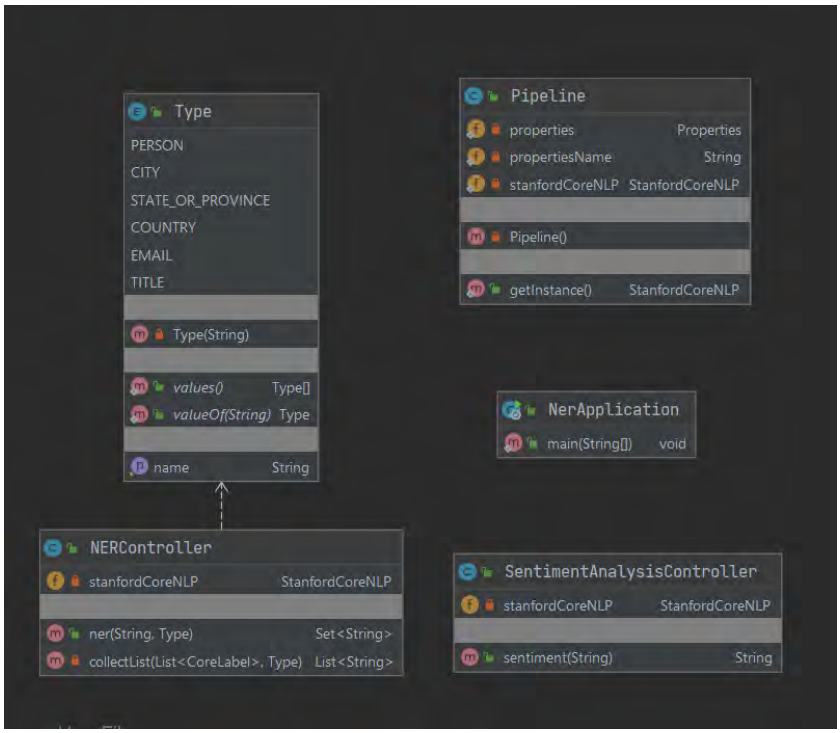


Рисунок 3.6 Блок-схема програмного коду

Розбити його можна на такі блоки:

- Pipeline
- SentimentAnalysis
- NER
- Інтерфейс користувача

3.3.1 Pipeline

Core NLP реалізує конвеєр анотацій, що також зветься “Pipeline”. Об’єкт Annotation використовується для зберігання результатів аналізу тексту і являється імплементацією Map. Спочатку текст документа додається до Annotation. Після цього Annotation Pipeline запускається через Annotation. AnnotationPipeline представляє собою лист аннотаторів (List<Annotator>).

Кожен аннотатор зберігає свої результати через один або більше ключів з аннотації (Annotation), робить аналіз природньої мови та записує результат назад у Annotation (рис. 3.7, рис. 3.8).

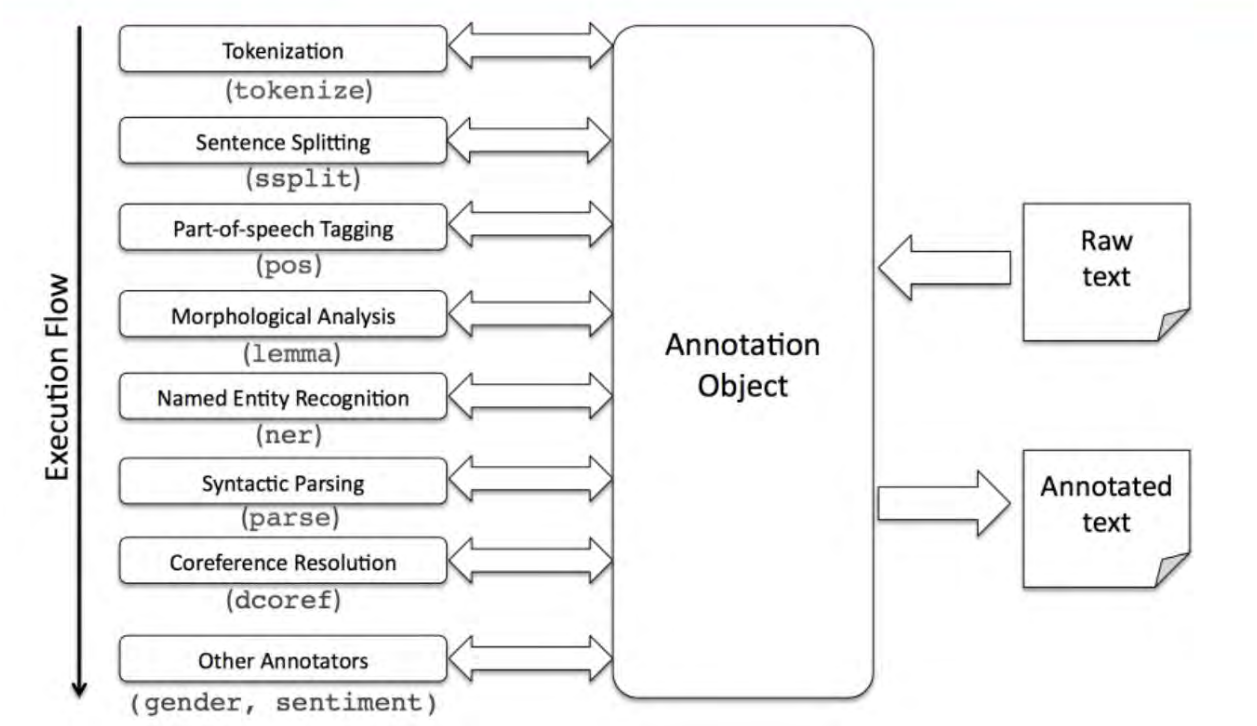


Рисунок 3.7 Схема роботи Pipeline

```
AnnotationPipeline pipeline = buildPipeline();
Annotation annotation = new Annotation("It's like a topography that is made f
f me.");
pipeline.annotate(annotation);
```

Рисунок 3.8 Приклад використання Pipeline

У бакалаврській дипломній роботі клас Pipeline представляє собою реалізацію паттерна Singletone. Повертаючи кожен раз одну і ту саму сутність об’єкта з потрібними доданими аннотаторами (рис 3.9, рис 3.10)

```
public static StanfordCoreNLP getInstance(){
    if (stanfordCoreNLP == null){
        stanfordCoreNLP = new StanfordCoreNLP(properties);
    }
    return stanfordCoreNLP;
}
```

Рисунок 3.9 Реалізація паттерна Singleton в контексті програмного коду бакалаврської роботи

```
private static Properties properties;
private static String propertiesName = "tokenize, ssplit, pos, lemma, ner, parse, sentiment";
private static StanfordCoreNLP stanfordCoreNLP;
```

Рисунок 3.10 Приклад використання Pipeline

3.3.2 Sentiment Analysis

Stanford CoreNLP включає в себе інструмент для визначення емоційної оцінки тексту. Цю модель можна додати до інструментарію та використовувати як частину бібліотеки додавши до списку аннотатор “sentiment”. Sentiment Analysis – процес встановлення емоційного забарвлення тексту: чи є він позитивним чи негативним чи взагалі нейтральним. Для того, щоб краще розуміти сам процес що ділиться на такі етапи:

1. Розбиває текст на частини (токени). Це можуть бути фрази, речення, слова, частини мови тощо. Цей процес також часто можуть називати «токенізація» (Tokenization). Токенізація – процес розбивання визначеного документу на лексеми (токени, частини) враховуючи послідовність символів та відкидаючи додаткові символи такі як пунктуація (рис. 3.11).

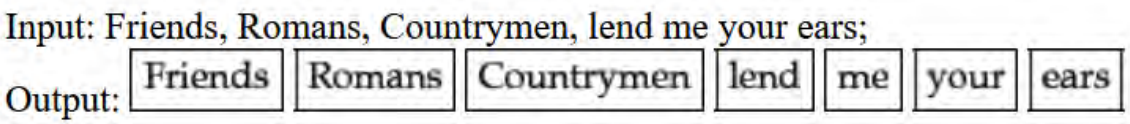
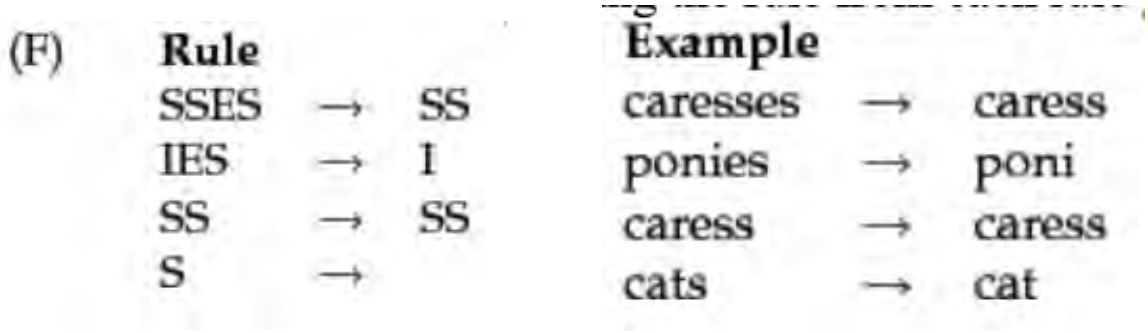


Рисунок 3.11 Приклад токенізації Pipeline

2. Спрощує слова в цих частинах до інфінітиву. З граматичних причин документ буде використовати різні форми одного й того ж слова. Щоб працювати з токенами далі необхідно спростити слова до їх «початкової» форми. Цей процес також вбирає в себе два схожі між собою терміни – стемінг та лемматизація. Цей процес представляє собою набір правил для отримання потрібної форми слова(рис 3.12).



(F)	Rule			Example		
	SSES	→	SS	caresses	→	caress
	IES	→	I	ponies	→	poni
	SS	→	SS	caress	→	caress
	S	→		cats	→	cat

Рисунок 3.12 Правила стемінгу/лемматизації

3. Визначає кожну фразу та компонент, що мають емоційний відтінок. Після розбиття слова на токени та спрощення граматичних закінчень програма проходить через кожне слово перевіряючи його емоційний відтінок та надає йому результат від -10 до +10
4. Призначає оцінку настрою кожній фразі або компоненту. Результати додаються між собою даючи потрібний результат.

В програмному коді дипломної роботи було реалізовано цей алоритм за допомогою хешмапи (HashMap), яка підраховувала кількість позитивних, негативних та нейтральних речень. Після чого я знаходив максимальний елемент в хешмапі (рис. 3.13, рис. 3.14)

```
Map<String, Integer> hashMap = new HashMap<>();

hashMap.put("Positive", 0);
hashMap.put("Negative", 0);
hashMap.put("Neutral", 0);
```

Рисунок 3.13 Ініціалізація хешмапи

```
int maxValueInMap=(Collections.max(hashMap.values()));
for (Map.Entry<String, Integer> entry : hashMap.entrySet()) {
    if (entry.getValue()==maxValueInMap) {
        return entry.getKey();
    }
}
```

Рисунок 3.14 Реалізація пошуку найбільшого значення в хешмапі

3.3.3 NER

Stanford NER імплементація Java для розпізнавання іменованих сутностей. Розпізнавання іменованих сутностей (NER – Named Entity Recognition) вішає «ярлички» на слова в тексті, що допомагає визначити – чи це ім'я людини, чи назва країна або назва професії. Використання розпізнавання іменованих сутностей являється покращенням програмного коду та робить веб-додаток більш зручним, інформативним ті корисним у використанні в порівнянні з аналогами. В своїй програмі я зробив можливість пошуку слів, що відповідають деяким категоріям і виніс все в окремий перелічувальний тип даних під назвою Type для подальшого визначення потрібної категорії та посилання відповідних запросів (рис. 3.15).

```
public enum Type {

    PERSON("Person"),
    CITY("City"),
    STATE_OR_PROVINCE("State_Or_Province"),
    COUNTRY("Country"),
    EMAIL("Email"),
    TITLE("Title");
```

Рисунок 3.15 Використання enum в контексті програмного коду бакалаврської роботи

Так як сутності часто бувають багатослівні, зазвичай задача NER обумовлюється класифікації на рівні токенов за допомогою заданих схем. Найзагальніша з них є BIES (рис. 3.16):

B – від слова beginning – перший токен в спані сущності що являється більш ніж одним словом.

I – від слова inside – те, що знаходиться всередині.

E – від слова ending, останній токен сутності.

S – від слова single. Якщо сутність лише з одного слова.

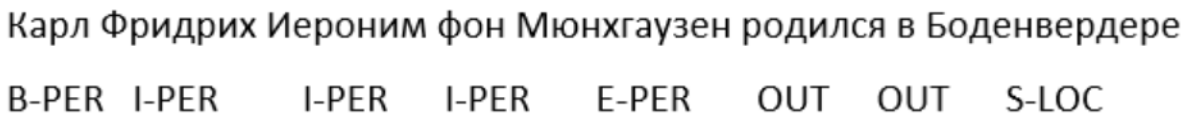


Рисунок 3.16 Приклад роботи схеми BIES

В програмному коді дипломного проекту всі слова визначені як потрібна сутність загортуються в лист строк (List<String>), що передається далі у вигляді Хешсету (HashSet<>) (рис. 3.17)

```
@PostMapping
@RequestMapping(value = "/ner")
public Set<String> ner(@RequestBody final String input, @RequestParam final Type type){
    CoreDocument coreDocument = new CoreDocument(input);
    stanfordCoreNLP.annotate(coreDocument);
    List<CoreLabel> coreLabels = coreDocument.tokens();
    return new HashSet<>(collectList(coreLabels, type));
}

private List<String> collectList (List<CoreLabel> coreLabels, final Type type){
    return coreLabels
            .stream()
            .filter(coreLabel -> type.getName().equalsIgnoreCase(coreLabel.get(CoreAnnotations.Named
            .map(CoreLabel::originalText)
            .collect(Collectors.toList());

}
```

Рисунок 3.17 Реалізація NER у програмному коді бакалаврської роботи

3.3.4 Інтерфейс користувача

По суті цей файл представляє собою відображення сторінки, «зовнішній вигляд» програми написане на HTML. На початку я підключив до відображення шаблони Bootstrap (рис. 3.18)

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Sentiment Analysis Application</title>
    <link rel="stylesheet" href="https://stackpath.bootstrapcdn.com/bootstrap/4.1.3/css/bootstrap.min.css">
    <script type="text/javascript" src="https://cdnjs.cloudflare.com/ajax/libs/jquery/3.3.1/jquery.min.js"></script>
</head>
```

Рисунок 3.18 Підключення Bootstrap

Після цього було додано декілька полей та кнопок для візуалізації програми після взаємодії з якими визивалася потрібна користувачу функція яка вже безпосередньо взаємодіяла з самою програмою (рис. 3.19).

```
<div class="form-group">
    <button type="button" class="btn btn-outline-light" onclick="sentiment()">Sentiment Analysis</button>
</div>

    <div class="form-group">
        <label for="type">Select type of Named-entity Recognition</label>
        <select class="form-control" id="type" onchange="ner(this.value)">
            <option value="">Select type</option>
            <option value="PERSON">Person</option>
            <option value="CITY">City</option>
            <option value="STATE_OR_PROVINCE">State or Province</option>
            <option value="COUNTRY">Country</option>
            <option value="TITLE">(Job) Title</option>
            <option value="EMAIL">Email</option>

        </select>
    </div>
<h3>Output:</h3>
<div class="form-group" id="result"></div>
```

Рисунок 3.19 Візуальне оформлення стартової сторінки веб-додадку

Ці функції являються частинами кода, що написані на мові JavaScript. Їх робота полягає в тому, щоб відправити POST-запрос з потрібними додатковими даними до програми, а потім отримати від неї результат у вигляди списку слів заданої сутності або ж емоційної оцінки використаного тексту (рис. 3.20).

```
<script type="text/javascript">
        function ner (type) {
            var input = $("#input").val();
            $.ajax({
                type: 'POST',
                url: "http://localhost:8080/api/v1/ner?type="+type,
                contentType: 'tet/plain; charset=utf-8',
                data: input,
                success:
                    function(response){
                    console.log(response)
                    var result = "";
                    $.each(response, function (index, value) {
                        result = result + "<span class='badge badge-success'>" + value + "</span> ";
                    })
                    $("#result").html(result);

                }
            });
        }
</script>
```

Рисунок 3.20 Функція ner

Висновки до розділу 3

В даному розділі було розглянуто обраний інструментарій. Він мав все необхідне для виконання бакалаврської роботи. Було розроблено веб-додаток, що відповідає умовам технічного завдання:

- Використання NLP-бібліотеки
- Реалізація методу аналізу емоційної оцінки мультимедійного контенту
- Реалізація методу розпізнавання іменованих сутностей
- Інтерфейс користувача

РОЗДІЛ 4. МОДЕЛЮВАННЯ І АНАЛІЗ РОБОТИ СИСТЕМИ

4.1 Кількісні характеристики моделі щодо її точності, швидкості

4.1.1 Швидкість алгоритмів моделі

Швидкість та об’єм потрібної пам’яті при користуванні бібліотеки Stanford CoreNLP залежить від обраних вами анотаторів. При якісному виборі конфігураційних налаштувань антотаторів CoreNLP не потрібно багато часу. Об’єм потрібної пам’яті в цій бібліотеці дійсно великий. Але це обумовлюється великими таблицями токенізатора кінцевих автоматів – моделі анотаторів, що займають близько половини потрібної пам’яті. Офіційний веб-сайт рекомендує обробляти великі файли частинами. Якщо ви використовуєте багато аннотаторів CoreNLP може витратити 10-40 секунд тільки завантажужаючи конвеєр аннотаторів. Отже при частому використанні конвеєру аннотаторів CoreNLP буде повільним. Тому потрібно використовувати один і той самий конвеєр за допомогою паттерна Singletone в мові програмування Java. Взятий з офіційного сайта графік (рис. 4.1) , що стосується старшої версії але дає уявлення про те, як залежить швидкість роботи системи від запису аннотацій.

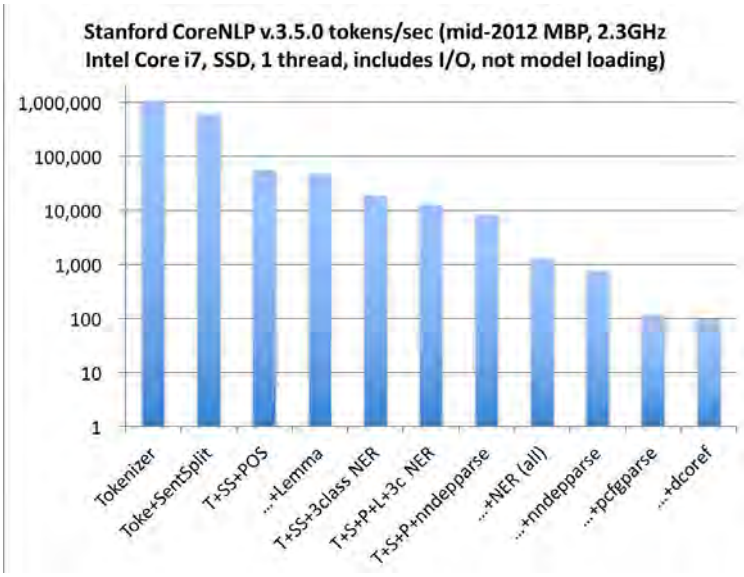


Рисунок 4.1 Залежність швидкості роботи CoreNLP від запуску аннотацій

За допомогою програмного коду дипломної бакалаврської роботи було зроблено заміри швидкості виконання аналізу емоційності тексту (табл. 4.1) та розпізнавання іменованих сутностей (табл. 4.2) на основі підбірок з 10, 50, 100, 500 та 1000 слів.

Таблиця 4.1

Кількість слів	10 слів	50 слів	100 слів	500 слів	1000 слів
Час виконання аналізу емоційної оцінки	0,236096 секунд	1,164522 секунд	2,61276 секунд	10,2687 секунд	29,4793 секунд

Таблиця 4.2

Кількість слів	10 слів	50 слів	100 слів	500 слів	1000 слів
Час виконання аналізу емоційної оцінки	0.149755 секунд	1.2789 секунд	2.8013 секунд	14.0594 секунд	24.8774 секунд

і. Точність алгоритмів моделі

Розроблений в даній бакалаврській роботі веб-додаток демонструє точні результати щодо прогнозування емоційної тональності тексту на відгуках на фільми. Більшість систем прогнозування працює просто, дивлячись на слова іозольовано. Вони дають за позитивні слова позитивні бали а за негативні

слова негативні бали, а потім підсумовують. Таким чином ігнорується порядок слів і втрачається важлива інформація. Алгоритми бібліотеки Stanford CoreNLP більш глибокі. Вони обчислюють емоційний відтінок беручі значення довших фраз. Таким чином, модель складніше обдурити, в порівнянні з більшістю систем. Наприклад, наша модель дізналася, що funny (смішний) та witty (дотепний) є позитивними словами, але наступне речення все ще залишається негативним:

This movie was actually neither that funny, nor super witty (Цей фільм насправді не був ні дійсно смішним, ні супер дотепним.).

Для підрахунку точності було взято датасет відгуків з найпопулярнішого у світі сайті для кіно imdb.com,що знаходився у відкритому доступі. Точність прогнозування емоційної оцінки відгуків для мультимедійного контенту досягає 98.5-98.8. До того ж, це єдина модель, яка може знаходити емоційну оцінку тексту а також шукати іменовані сутності.

4.2 Опис системи і її порівняння із існуючим аналогами

4.2.1 Опис системи

Інтерфейс користувача програми розроблявся простим у використанні та мінімалістичним, не маючи в собі нічого зайвого.

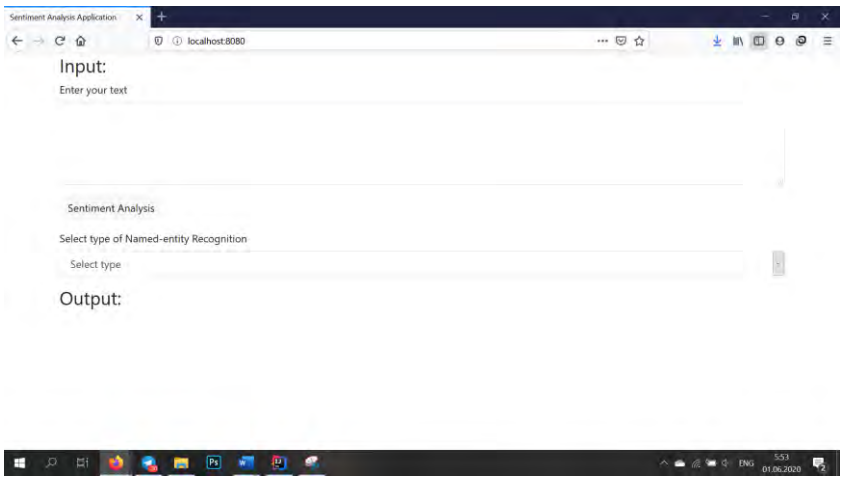


Рисунок 4.1 Інтерфейс користувача

З самого верху можна побачити пусте поле с написом «Вставте свій текст», що дає змогу інтуїтивно зрозуміти як користуватися даним додатком. Після введення тексту у користувача є два варіанти – зробити аналіз емоційної оцінки тексту або визначити сутності серед запропонованих категорій (рис. 4.2)

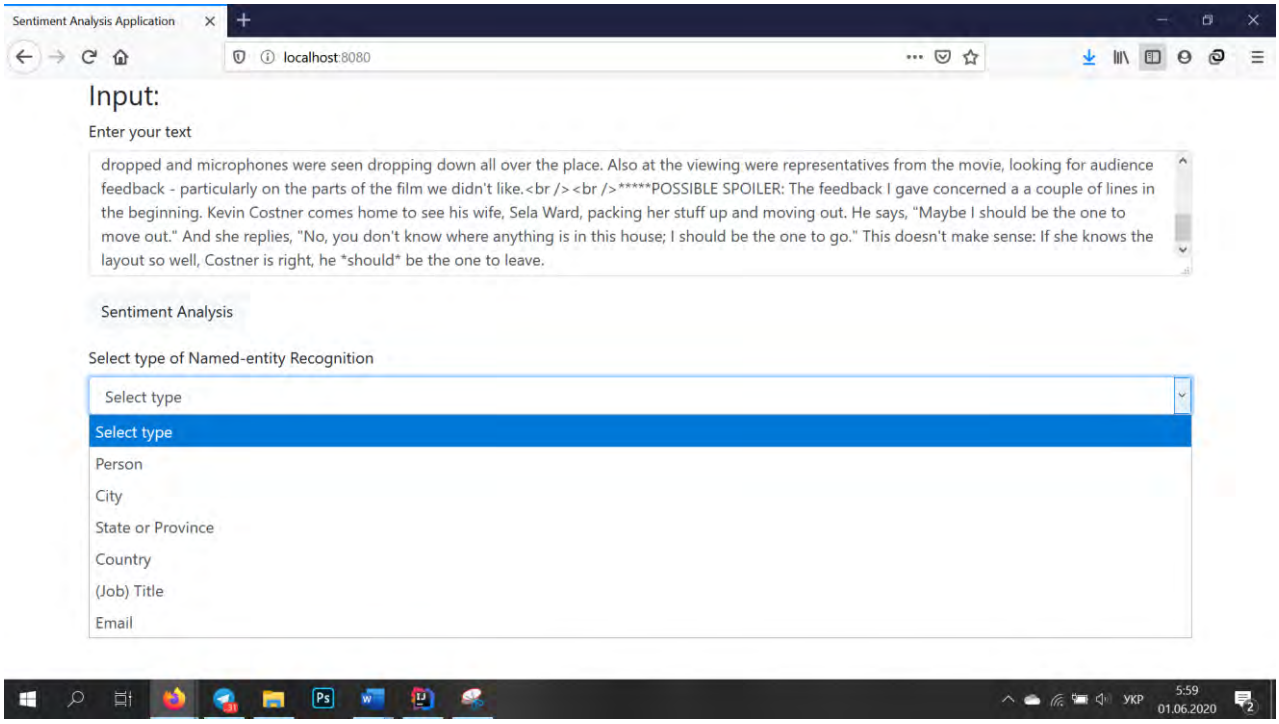


Рисунок 4.2 Меню вибора категорій

Якщо користувач обере аналіз емоційного відтінку він має натиснути на кнопку за написом «Sentiment Analysis». Після чого він побачить результат емоційної оцінки в полі Output (рис 4.3).

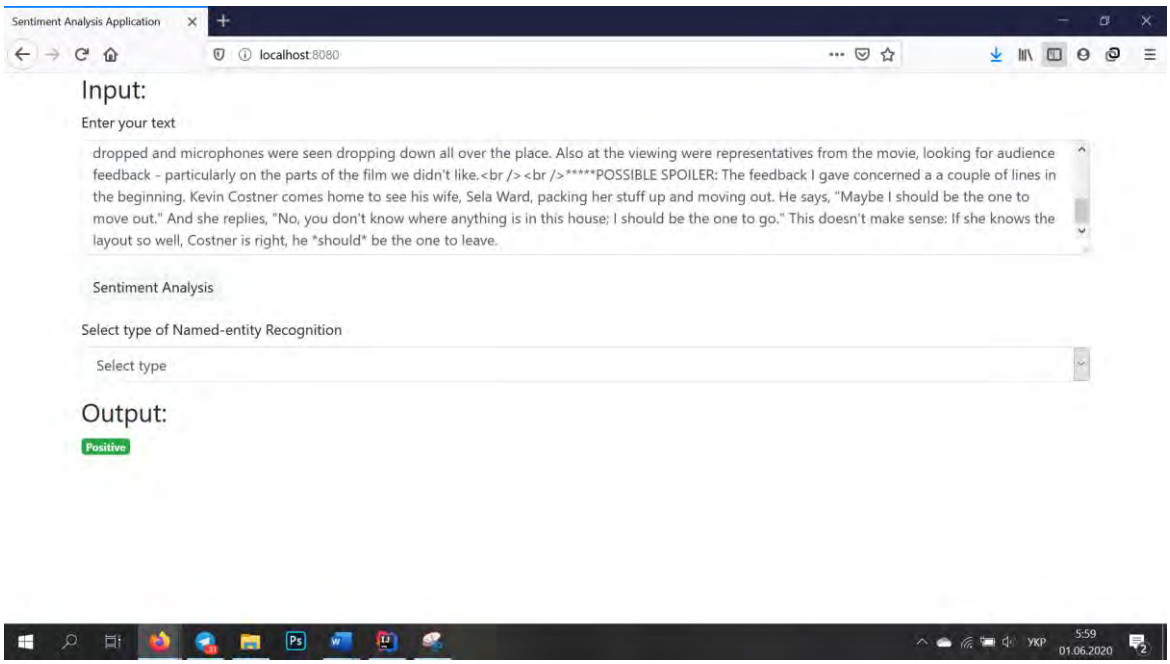


Рисунок 4.3 Аналіз емоційної оцінки мультимедійного контенту

Якщо користувач обирає розпізнавання сутностей, він має обрати по які з заданих сутностей він хоче визначити. Після натискання на відповідний варіант програма надає результат (рис. 4.4).

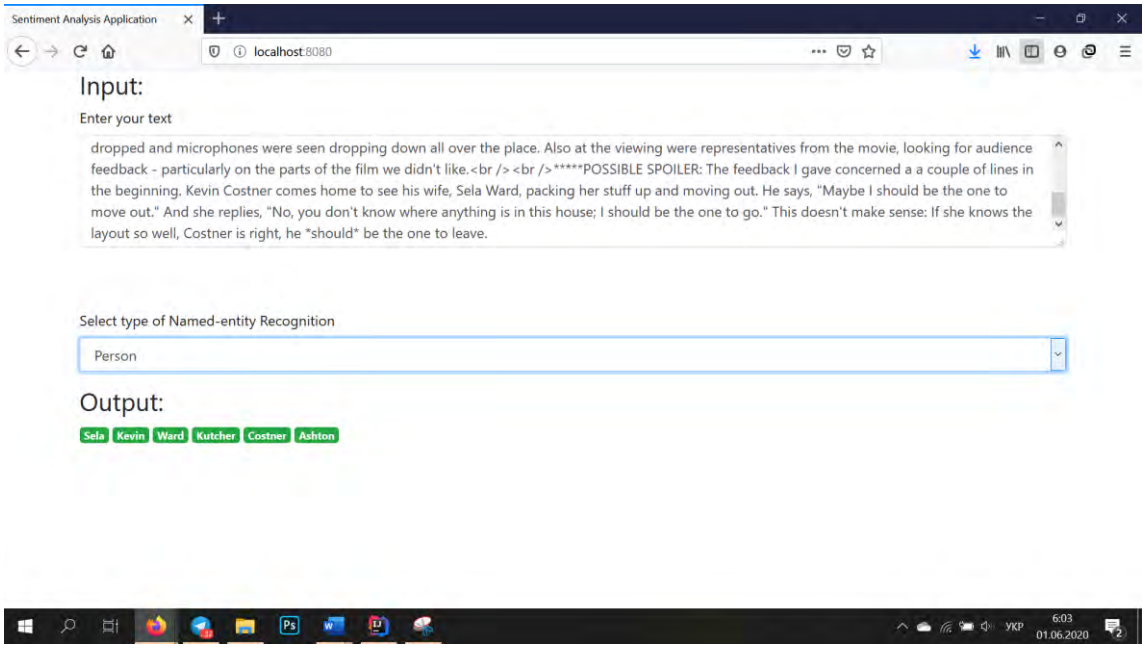


Рисунок 4.4 Розпізнавання іменованих сутностей

4.2.2 Порівняння системи з існуючими аналогами

Під час виконання бакалаврської дипломної роботи було переглянуто численну кількість аналогічних програм та веб-додатків. Більшість з них використовує інші, менш розвинуті алгоритми аналізу емоційної оцінки тексту. Ті роботи, що використовують бібліотеку Stanford CoreNLP також мають деякі недоліки в порівнянні з моєю системою. Головні недоліки що було помічено:

- Аналіз емоційної оцінки тексту виконується не відносно всього тексту, а лише відносно кожного речення в тексті. Дана Бакалаврська робота виправляє цей недолік та дає змогу користувачи проаналізувати потрібний текст повністю та видати одну оцінку.
- Відсутнє розпізнавання іменованих сутностей.

Висновок до розділу 4

В заключному розділі було описано кінцевий вигляд та функціонал системи аналізу емоційної оцінки мультимедійного контенту. Було замірено швидкість роботи методів системи на основі тексту різного об’єму. Було визначено точність алгоритмів в моделі, яка показала високі результати на визначенні емоційної тональності відгуків на фільми. Також систему було порівняно з існуючими аналогами, в результаті чого дана система мала суттєві переваги.

Загальний висновок

У даній бакалаврській роботі було виконано огляд інструментів для обробки природних мов, а саме аналізу емоційності тексту, та розпізнаванню іменованих сутностей. В процесі аналізу було вирішено обрати бібліотеку з найбільш сучасними алгоритмами виконання аналізу емоційності тексту. В результаті було створено веб-додаток, який показав гарні результати в точності алгоритму та середні результати в часі виконання. Для покращення часу виконання роботи програми можна переписати методи, в яких вхідні дані розбивалися на паралельні потоки та записувалися до мапи атомних змінних. Але в контексті відгуків на фільми додаток не має проблем зі швидкістю адже відгуки зазвичай не перевищують 1000 слів. Хоча цей додаток можна використовувати з майже будь яким текстом у повсякденному житті. Найяскравішим прикладом буде аналіз відгуків користувачів щодо якогось IT-проекту, або коментарі в соцмережах якогось відомого бренду. З допомогою цього додатка компанії більш точно зможуть зрозуміти відношення аудиторії, та краще розуміти які помилки треба виправити.

Список використаної літератури

1. Zhang, Y., Jin, R., & Zhou, Z. H. (2010). Understanding bag-of-words model: a statistical framework. International Journal of Machine Learning and Cybernetics, 1(1-4), 43-52.Opinion Mining and Sentiment Analysis - Bo Pang, Lillian Lee
2. Liu, B. (2012). Sentiment analysis and opinion mining. Synthesis lectures on human language technologies, 5(1), 1-167.
3. Korenius, T., Laurikkala, J., Järvelin, K., & Juhola, M. (2004, November). Stemming and lemmatization in the clustering of finnish text documents. In Proceedings of the thirteenth ACM international conference on Information and knowledge management (pp. 625-633).
4. Gobet, F., Lane, P. C., Croker, S., Cheng, P. C., Jones, G., Oliver, I., & Pine, J. M. (2001). Chunking mechanisms in human learning. Trends in cognitive sciences, 5(6), 236-243.
5. Gorrell, P. (1995). Syntax and parsing (Vol. 76). Cambridge University Press.
6. Ratnaparkhi, A. (1996). A maximum entropy model for part-of-speech tagging. In Conference on Empirical Methods in Natural Language Processing.
7. Hutto, C. J., & Gilbert, E. (2014, May). Vader: A parsimonious rule-based model for sentiment analysis of social media text. In Eighth international AAAI conference on weblogs and social media.
8. Twitter Usage Statistics [Електронний ресурс]. Режим доступу: https://www.internetlivestats.com/twitter-statistics/
9. Sentiment Analysis: What is it and Why Does it Matter? [Електронний ресурс]. Режим доступу: https://www.upgrad.com/blog/sentiment-analysis-what-is-it-and-why-does-it-matter/
10. Sentiment Analysis Explained [Електронний ресурс]. Режим доступу: https://www.lexalytics.com/technology/sentiment-analysis
11. The 7 Basic Functions of Text Analytics & Text Mining [Електронний ресурс]. Режим доступу: https://www.lexalytics.com/lexablog/text-analytics-functions-explained

12. Understanding Sentiment Analysis: What It Is & Why It's Used [Електронний ресурс]. Режим доступу: https://www.brandwatch.com/blog/understanding-sentiment-analysis/
13. What is Sentiment Analysis: Definition, Key Types and Algorithms [Електронний ресурс]. Режим доступу: https://theappsolutions.com/blog/development/sentiment-analysis/
14. Sentiment Analysis [Електронний ресурс]. Режим доступу: https://monkeylearn.com/sentiment-analysis/
15. Stemming and lemmatization [Електронний ресурс]. Режим доступу: https://nlp.stanford.edu/IR-book/html/htmledition/stemming-and-lemmatization-1.html
16. Spring Framework [Електронний ресурс]. Режим доступу: **https://en.wikipedia.org/wiki/Spring_Framework**
17. Drakos, N., & Mann, H. (2011). Computer Based Learning Unit, University of Leeds. Ross Moore, Mathematics Department, Macquarie University, Sydney. Rapporté de http://www. etis. ensea. fr/~ revel/html/cours_IA/cours_IA. html.
18. Moore, R., & Griffin, F. (2001). Mathematics Department, Macquarie University, Sydney.
19. Natural Language Toolkit [Електронний ресурс]. Режим доступу: https://www.nltk.org/
20. Bird, S., Klein, E., & Loper, E. (2009). Natural language processing with Python: analyzing text with the natural language toolkit. " O'Reilly Media, Inc.".
21. PyTorch-Transformers [Електронний ресурс]. Режим доступу: https://pytorch.org/hub/huggingface_pytorch-transformers/
22. TextBlob: Simplified Text Processing [Електронний ресурс]. Режим доступу: https://textblob.readthedocs.io/en/dev/
23. Industrial-Strength Natural Language Processing [Електронний ресурс]. Режим доступу: https://spacy.io/

24. Stanford CoreNLP – Natural language software [Електронний ресурс]. Режим доступу: https://stanfordnlp.github.io/CoreNLP/index.html
25. Apache OpenNLP Developer Documentation [Електронний ресурс]. Режим доступу: https://opennlp.apache.org/docs/1.9.2/manual/opennlp.html

ДОДАТОК А. Код програми.

SentimentAnalysisController

```
package com.ilyayevtushenko.nerapplication.controller;
import com.ilyayevtushenko.nerapplication.model.Type;
import edu.stanford.nlp.ling.CoreAnnotations;
import edu.stanford.nlp.ling.CoreLabel;
import edu.stanford.nlp.pipeline.CoreDocument;
import edu.stanford.nlp.pipeline.CoreSentence;
import edu.stanford.nlp.pipeline.StanfordCoreNLP;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.web.bind.annotation.*;
import java.util.*;
import java.util.stream.Collectors;

import static java.util.Map.Entry.comparingByValue;
import static java.util.stream.Collectors.toMap;

@RestController
@RequestMapping(value = "/api/v1")
public class SentimentAnalysisController {
    @Autowired
    private StanfordCoreNLP stanfordCoreNLP;



    @PostMapping
    @RequestMapping(value = "/sentiment")
    public String sentiment(@RequestBody final String input) {
        CoreDocument coreDocument = new CoreDocument(input);
        stanfordCoreNLP.annotate(coreDocument);
        List<CoreSentence> sentences = coreDocument.sentences();

        Map<String, Integer> hashMap = new HashMap<>();
```

```
        hashMap.put("Positive", 0);
        hashMap.put("Negative", 0);
        hashMap.put("Neutral", 0);

        for (CoreSentence sentence : sentences){
          String sentiment = sentence.sentiment();
          hashMap.put(sentiment, hashMap.get(sentiment)+1);
        }

        int maxValueInMap=(Collections.max(hashMap.values()));
        for (Map.Entry<String, Integer> entry : hashMap.entrySet()) {
            if (entry.getValue()==maxValueInMap) {
                return entry.getKey();


            }
        }
        return "Neutral";
    }

}
```

NERController

```
package com.ilyayevtushenko.nerapplication.controller;
import com.ilyayevtushenko.nerapplication.model.Type;
import edu.stanford.nlp.ling.CoreAnnotations;
import edu.stanford.nlp.ling.CoreLabel;
import edu.stanford.nlp.pipeline.CoreDocument;
import edu.stanford.nlp.pipeline.StanfordCoreNLP;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.web.bind.annotation.*;
import java.util.HashSet;
import java.util.List;
```

```
import java.util.Set;


import java.util.stream.Collectors;
@RestController
@RequestMapping(value = "/api/v1")


public class NERController {
    @Autowired
    private StanfordCoreNLP stanfordCoreNLP;


    @PostMapping
    @RequestMapping(value = "/ner")
    public Set<String> ner(@RequestBody final String input, @RequestParam final Type type){
        CoreDocument coreDocument = new CoreDocument(input);
        stanfordCoreNLP.annotate(coreDocument);
        List<CoreLabel> coreLabels = coreDocument.tokens();
        return new HashSet<>(collectList(coreLabels, type));
    }

    private List<String> collectList (List<CoreLabel> coreLabels, final Type type){
        List<String> collectList = coreLabels
                .stream()
                .filter(coreLabel ->
type.getName().equalsIgnoreCase(coreLabel.get(CoreAnnotations.NamedEntityTagAnnotation.c
lass)))
                .map(CoreLabel::originalText)
                .collect(Collectors.toList());
                 return collectList;
    }
}
```

Pipeline

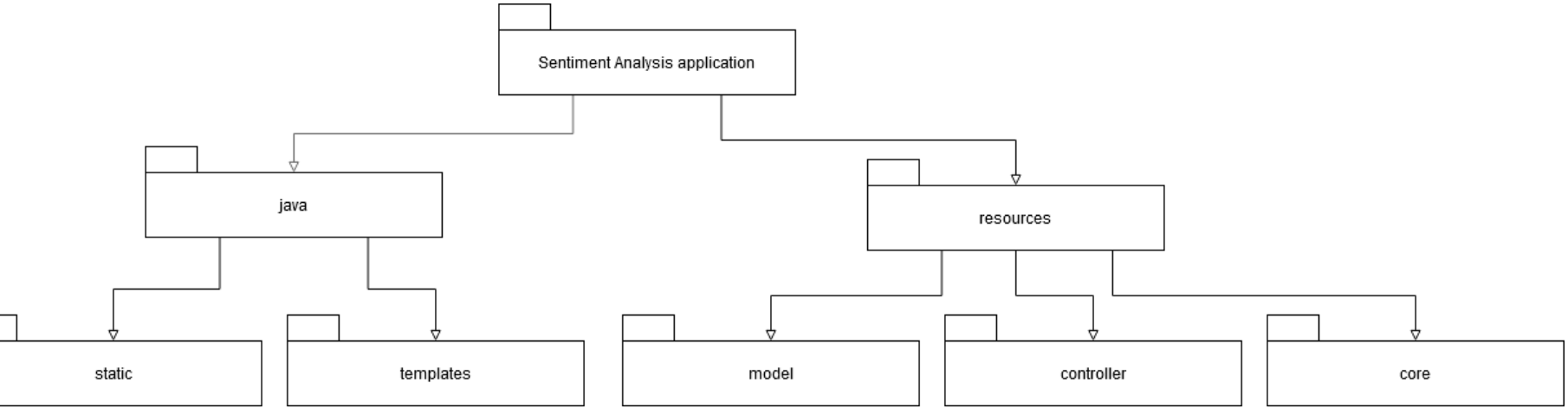
```
package com.ilyayevtushenko.nerapplication.core;

import edu.stanford.nlp.pipeline.StanfordCoreNLP;
import org.springframework.context.annotation.Bean;
import org.springframework.stereotype.Service;
import java.util.Properties;

@Service
public class Pipeline {
    private static Properties properties;
    private static String propertiesName = "tokenize, ssplit, pos, lemma, ner, parse, sentiment";
    private static StanfordCoreNLP stanfordCoreNLP;
    private Pipeline(){
    }
    static {
        properties = new Properties();
        properties.setProperty("annotators", propertiesName);
    }


    @Bean (name = "stanfordCoreNLP")
    public static StanfordCoreNLP getInstance(){
        if (stanfordCoreNLP == null){
            stanfordCoreNLP = new StanfordCoreNLP(properties);
        }
        return stanfordCoreNLP;
    }
}
```

Type

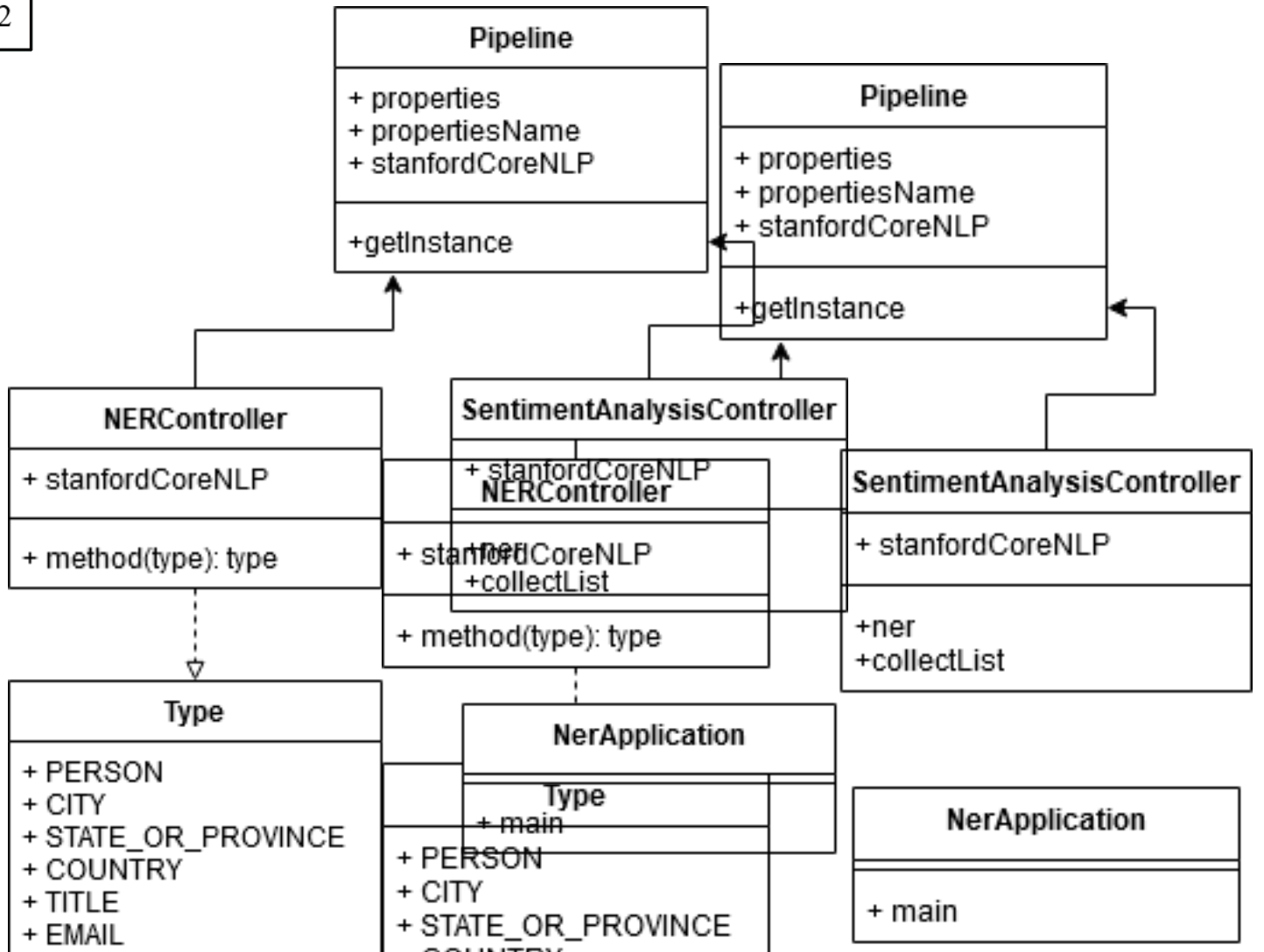
```
package com.ilyayevtushenko.nerapplication.model;
public enum Type {
   PERSON("Person"),



   CITY("City"),
   STATE_OR_PROVINCE("State_Or_Province"),
   COUNTRY("Country"),
   EMAIL("Email"),
   TITLE("Title");
   private String type;
   Type(String type){
      this.type = type;
   }

   public String getName(){
      return type;
   }
}
```

NERApplication

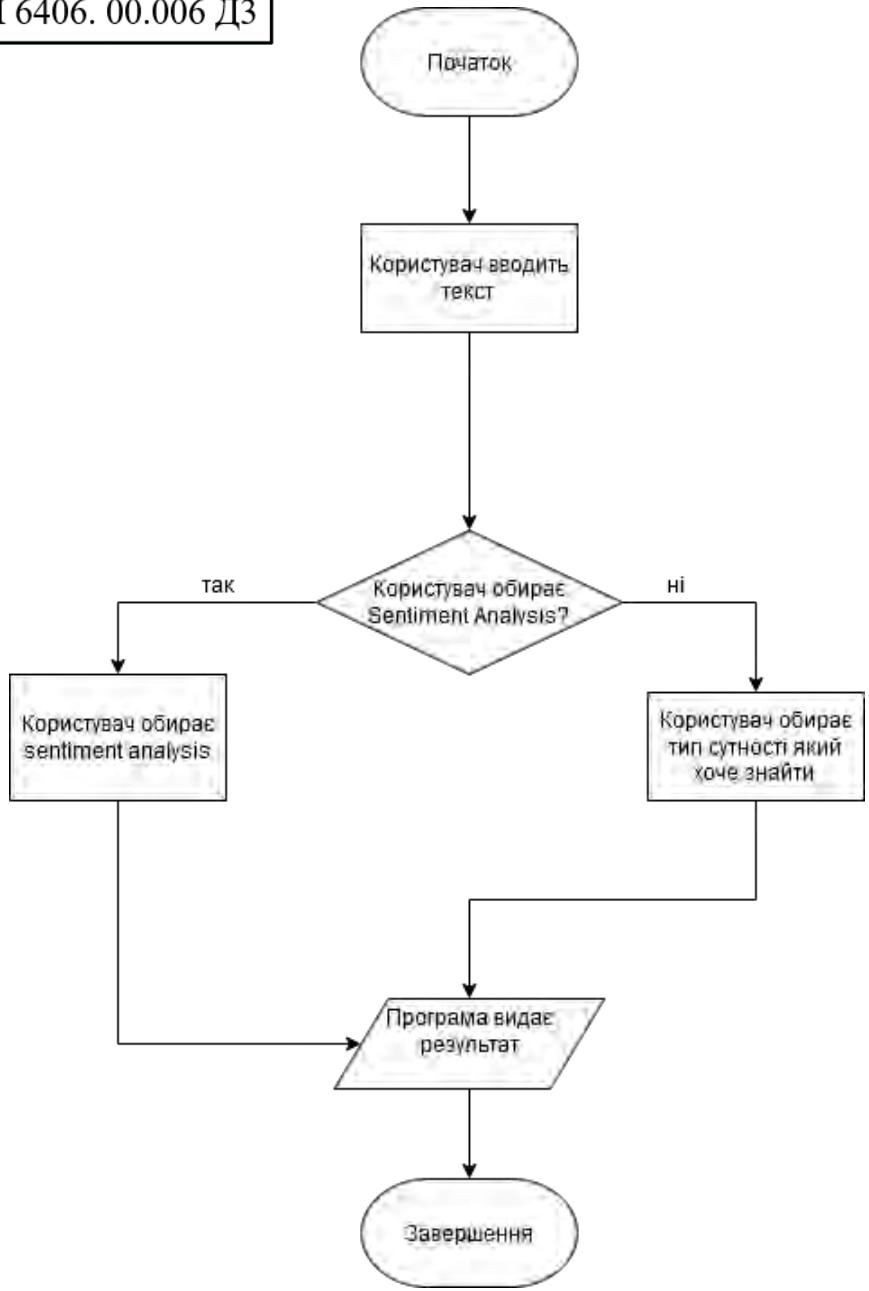
```
package com.ilyayevtushenko.nerapplication;

import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;
@SpringBootApplication
public class NerApplication {
   public static void main(String[] args) {
      SpringApplication.run(NerApplication.class, args);
   }

}
```



					ДП 6406. 00.004 Д1	
					Принципова схема системи	Літ. / Маса / Масштаб
Зм.	Арк.	№ докум.	Підпис	Дата		
Розроб.		Євтушенко І.І.				Арк. 1 / Аркушів 1
Перевір.		Гордієнко Ю.Г.				
Н. контр.		Сімоненко В. П.			Дипломний проект	КПІ ФІОТ кафедра ОТ гр. ІП-64
Затверд.		Стіренко С.Г.				



					ДП 6406. 00.005 Д2	
Зм.	Арк.	№ докум.	Підпис	Дата	Структурна схема системи	Літ. / Маса / Масштаб
Розроб.		Єстушенко І.І.				
Перевір.		Гордієнко Ю.Г.				Арк. 1 / Аркушів 1
Н. контр.		Симоненко В. П.			Дипломний проект	КПІ ФІОТ кафедра ОТ гр. ІП-64
Затверд.		Стіренко С.Г.				



					ДП 6406. 00.006 Д3	
Зм.	Арк.	№ докум.	Підпис	Дата	Функціональна схема системи	Літ. / Маса / Масштаб
Розроб.		Євтушенко І.І.				
Перевір.		Гордієнко Ю.Г.				Арк. 1 / Аркушів 1
Н. контр.		Сімоненко В. П.			Дипломний проект	КПІ ФІОТ кафедра ОТ гр. ІП-64
Затверд.		Стіренко С.Г.				