Київський національний університет імені Тараса Шевченка

Факультет інформаційних технологій

Кафедра програмних систем і технологій

УДК _____

На правах рукопису

ВИПУСКНА КВАЛІФІКАЦІЙНА МАГІСТЕРСЬКА РОБОТА

Тема: “Семантичний аналіз економічних новин для прогнозування розвитку економіки на основі технологій обробки великих даних”

Спеціальність – 121 “Інженерія програмного забезпечення”

ПОЯСНЮВАЛЬНА ЗАПИСКА

БР.ІПЗ – 30.00.00.000

Студент

ІПЗм-21 _____ /Андрій КРАВЦОВ/

Науковий керівник

к.т.н., доц. _____ /Тетяна КОВАЛЮК/

Консультант

з питань нормоконтролю

к.т.н., асистент _____/ Анастасія ВЕЧЕРКОВСЬКА /

Допускається до захисту

Завідувач кафедри

д.т.н., проф. _____ /Олексій БИЧКОВ/

Київ – 2022

ЗАВДАННЯ

НА ВИПУСКНУ КВАЛІФІКАЦІЙНУ МАГІСТЕРСЬКУ РОБОТУ

СТУДЕНТУ

Кравцову Андрію Олександровичу

1. Тема випускної кваліфікаційної магістерської роботи «Семантичний аналіз економічних новин для прогнозування розвитку економіки на основі технологій обробки великих даних» та керівник проекту (роботи) Ковалюк Тетяна Володимирівна, к.т.н., доцент затверджені наказом вищого навчального закладу від «__»____20__ р. №_____

2. Строк здачі студентом закінченої роботи «__»____20__ р.

3. Вихідні дані до роботи: підручники, навчальні посібники, статті, Інтернет-ресурси.

4. Зміст пояснювальної записки (перелік питань, що їх належить розробити)

Аналітична частина:

- обґрунтувати актуальність розробки моделі прогнозування змін економіки на основі семантичного аналізу новин;
- дослідити та обрати оптимальну модель для розробки системи;
- дослідити засоби аналізу природного мовлення та семантичного аналізу;

Практична частина:

- спроектувати архітектуру системи відповідно до визначених задач;
- підготувати датасет;
- розробити систему прогнозування.

5. Консультанти з роботи із зазначенням розділів роботи, що їх стосуються

Розділ	Консультант	Підпис, дата	
		Завдання видав	Завдання прийняв
Аналіз предметної області та теоретичних аспектів	Ковалюк Т.В.	27.09.21	14.12.21
Проектування та вибір технологій програмування	Ковалюк Т.В.	14.12.21	17.01.22

Програмна реалізація	Ковалюк Т.В.	17.01.22	04.05.22
Огляд результату виконання поставленої задачі	Ковалюк Т.В.	04.05.22	10.05.22

7. Дата видачі завдання _____

Керівник _____ (Ковалюк Т.В.)

Завдання прийняв до виконання _____ (Кравцов А.О.)

КАЛЕНДАРНИЙ ПЛАН

Номер і назва етапів роботи	Термін виконання етапів роботи	Примітка
1: Дослідження та аналіз предметної області	27.09.21 – 22.10.21	
2: Дослідження та вибір технології аналізу природного мовлення	25.10.21 – 30.11.21	
3: Вибір та обґрунтування технологій розробки	30.11.21 – 14.12.21	
4: Розробка архітектури системи	14.12.21 – 24.12.21	
5: Вибір та обґрунтування моделей прогнозування	24.12.21 – 17.01.22	
6: Вибір та підготовка датасетів	18.01.22 – 04.02.22	
7: Розробка програмних компонент	07.02.22 – 11.04.22	
8: Тренування моделі	12.04.22 – 22.04.22	
9: Аналіз результатів	25.04.22 – 10.05.22	

Студент – магістр _____ (Кравцов А.О.)

Керівник роботи _____ (Ковалюк Т.В.)

АНОТАЦІЯ

Випускна кваліфікаційна магістерська робота складається зі вступу, 3 розділів, висновків та списку використаних джерел (38 найменувань). Містить у собі 13 рисунків, 1 таблиці та 2 додатків. Загальний обсяг роботи становить 87 сторінки.

Темою роботи є “Семантичний аналіз економічних новин для прогнозування розвитку економіки на основі технологій обробки великих даних”.

Об’єктом роботи є семантичний аналіз економічних новин. **Предметом** – прогнозування розвитку економіки.

Метою роботи є дослідження можливості використання семантичного аналізу новин для подальшого прогнозування змін в економіці.

Результати роботи: програмний засіб демонструючий можливість застосування семантичного аналізу разом з моделями часових рядів для прогнозування змін економіки.

Ключові слова: семантичний аналіз, прогнозування, машинне навчання, ARIMA, SARIMA.

ABSTRACT

The final qualifying master's thesis consists of an introduction, 3 chapters, conclusions and a list of sources used (38 items). Contains 13 figures, 1 table and 2 appendices. The total volume of the work is 87 pages.

The **theme** of the work is "Semantic analysis of economic news for forecasting economic development based on big data processing technologies".

The **object** of the work is semantic analysis of economic news. **Subject** - forecasting economic development.

The **goal** of the work is to study the possibility of using semantic analysis of news to further predict changes in the economy.

Results of work: software demonstrating the possibility of using semantic analysis together with time series models to predict changes in the economy.

Keywords: semantic analysis, forecasting, machine learning, ARIMA, SARIMA.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ СКОРОЧЕНЬ

ПЗ – програмне забезпечення

LTSM (Long Term Short Memory) – довга короткочасна пам'ять

NLP (Natural Language Processing) – обробка природної мови

SVM (support vector machines) – метод опорних векторів

ВСТУП

Сьогодні, питання семантичному аналізу тексту, стає дедалі популярнішим, так як у століття комп'ютеризації, стає більш актуальною, автоматична обробка тексту. Це пов’язано з великим обсягом текстової інформації. Семантичний аналіз є найпоширенішим способом автоматичної обробки інформації.

Семантичний аналіз тексту – одна з основних проблем як теорії створення систем штучного інтелекту, що стосується обробки природної мови (Natural Language Processing, NLP), так і комп'ютерної лінгвістики. У наш час комп'ютерних технологій надзвичайно важливо отримувати інформацію природною мовою і зробити так, щоб ця інформація могла оброблятися за допомогою комп'ютера.

Слід зазначити, що інформація за своєю сутністю є універсальним ресурсом. У сучасній науці її найчастіше відносять до різновиду економічних ресурсів, оскільки інформація безпосередньо впливає на відносини, які в кінцевому підсумку виражаються в грошовому еквіваленті. Але водночас інформація впливає як на соціально-економічні відносини, так і має важливе значення для наукового, технічного прогресу, представляє цінність для військових і політичних завдань.

Сучасна література, що описує процеси в поведінкових фінансах, вказує на людське сприйняття, інстинкт та настрої інвесторів як на важливі елементи, які можуть керувати їхніми судженнями та прийняттям рішень, в кінцевому підсумку впливаючи на їхні інвестиційні рішення. Щоб врахувати ці фактори, нещодавні дослідження пропонують вимірювати настрої з таких фрагментів тексту, як новини, блоги та інші форми письмового спілкування, і використовувати їх для моделювання та прогнозування подій на фінансовому ринку. Новини часто містять непередбачувану інформацію про стан економіки у вигляді опису стану ринків, коментарів щодо їхньої еволюції чи про можливі втручання в монетарну політику. Через призму новинних статей учасники ринку дізнаються про останні економічні події та тенденції, коригують своє сприйняття та очікування щодо динаміки фінансових ринків. Ряд досліджень витягують індикатори настроїв із новин і

використовують їх для моделювання та прогнозування розвитку економіки. Ці роботи обчислюють економічні настрої, дивлячись на те, скільки позитивних і негативних слів можна знайти в тексті відповідно до попередньо визначеного лексикону, наприклад списків слів. Емпіричні результати свідчать про те, що настрої новин передають корисну додаткову інформацію порівняно з кількісними фінансовими даними для прогнозування розвитку економіки.

За останні роки вимоги до обчислювальної потужності різко зросли. Це також має місце у багатьох завданнях обробки мови через величезну й постійно зростаючу кількість текстової інформації, яку необхідно обробляти за розумні проміжки часу. Цей сценарій призвів до зміни парадигми в обчислювальних архітектурах і великомасштабних стратегіях обробки даних, які використовуються в області обробки природних мов.

Мета дослідження – розробка програмного забезпечення семантичного аналізу економічних новин для прогнозування розвитку економіки на основі технологій обробки великих даних.

Завдання дослідження:

- Провести аналіз існуючих рішень та огляд предметної області.
- Описати методологію семантичного аналізу тексту та передбачення часових рядів та моделі прогнозування ARIMA та SARIMA.
- Розробити програмного забезпечення семантичного аналізу економічних новин для прогнозування розвитку економіки на основі технологій обробки великих даних.

Об’єкт дослідження – семантичний аналіз тексту на основі технологій обробки великих даних.

Предмет дослідження – прогнозування цін на нафту, як важливого фактору розвитку економіки.

РОЗДІЛ 1
АНАЛІЗ ІСНУЮЧИХ РІШЕНЬ ТА ОГЛЯД ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Засоби обробки природної мови

Одним із напрямків комп'ютерної лінгвістики є «обробка природної мови» (NLP). Обробка природної мови ставить перед собою завдання дослідження та розробки методів і систем, що забезпечують реалізацію процесу спілкування людини з комп’ютером природною мовою, тобто створення природного-мовного інтерфейсу, наприклад, за допомогою розпізнавання мови і синтезу мови по тексту, розпізнавання вхідного тексту, розробки системи «питання-відповідь» (Question Answering), отримання фактів і знань (Data Mining, Information Extraction/Retrieval), автоматичного машинного перекладу тощо [1].

Найчастіше автоматичний аналіз тексту розбивається на кілька етапів: графематичний, морфологічний, синтаксичний, семантичний, концептуальний, прагматичний, а також застосовуються спільно з ними або окремо статистичні методи аналізу [2].

Обробка природної мови включає розпізнавання і генерацію мови, класифікацію, екстракцію знань з текстів та інші дії, спрямовані на розуміння текстів з метою наповнення баз знань, формування відповідей на питання і ведення діалогу. Завдання обробки текстів уперше було розглянуто у п'ятдесятих роках у роботах американського лінгвіста Ноама Хомського [3] з граматики природної мови, в яких було описано ключову парадигму комп'ютерної лінгвістики – контекстно-незалежну граматику. Перші підходи до глибокої обробки текстів зазвичай зводилися до розбору мови із застосуванням такої граматики, а також шляхом перекладу з дерева розбору на деяке логічне представлення знань за допомогою зводу правил та спеціально заготовленого лексикону. Після цього логічне представлення можна було додати до бази знань і виконувати над ним різного роду операції, відповідати на запитання, перевіряти твердження тощо. Проте при спробах практичного застосування цього підходу

виникали складнощі, пов'язані у тому числі з необхідністю враховувати загальноприйняті (тобто елементарні, базові) знання про світ [4]. Наприклад, інтелектуальний асистент планування поїздки повинен мати велику кількість начебто примітивних даних: йому треба розуміти, що вночі люди сплять (щоб не планувати на ніч якісь переїзди), що у північних морях взимку більшість людей не купаються тощо. Ці відомості треба було якось зберігати і враховувати при прийнятті рішень, і в 1984 стартував проект CYC (від англійського encyclopedia) зі створення великої бази загальних знань та розробки прийнятного механізму формування висновків. Проект існує досі, але, по суті, провалився — лише деякі дослідницькі розробки використовують сьогодні цю базу знань.

На початку 90-х років стали розвиватися методи машинного навчання і одночасно було зроблено низку робіт зі статистичної лінгвістики. У машинному навчанні добре себе показали алгоритми класифікації для різних завдань, пов'язаних з обробкою текстів: детекція спаму, сортування документів за тематиками, виділення іменованих сутностей. З'явилася область тематичного моделювання, в якій документи вважаються породженням певного ймовірнісного процесу і складаються із суміші тем. У комп'ютерній лінгвістиці визначення частин мови стало високоточним завдяки таким статистичним методам, як приховані ланцюги Маркова та моделі максимальної ентропії. З'явилися парсери на основі ймовірнісних контекстно-незалежних граматик, а в корпорації IBM пройшов масштабний проект статистичного машинного перекладу. Нарешті, було закладено основи глибокого навчання (deep learning), яке недавно дало перші паростки, зумовлені прогресом у сфері високопродуктивних систем і появою великих обсягів даних, що використовуються для навчання. Глибоке навчання – навчання багаторівневих («глибоких») нейронних мереж на великих обсягах даних, що дозволяють виключити роботу зі створення ознак для машинного навчання, надаючи можливість одночасного навчання виділенню ознак та навчання безпосередньо самому завданню [5].

У 2010 році було запропоновано модель лексикалізованої ймовірнісної граматики, яка дозволила підвищити точність граматичного розбору до 93%, що,

звісно, далеко від ідеалу. Точність аналізу – це відсоток правильно побудованих граматичних зв'язків, і ймовірність того, що довге речення буде розібране правильно, зазвичай дуже низька. Одночасно завдяки новим алгоритмам і підходам, включаючи глибоке навчання, збільшилася швидкість граматичного розбору. Крім того, практично всі провідні алгоритми і моделі стали доступні широкому загалу дослідників, і, напевно, найвідомішою роботою в галузі глибокого навчання для NLP став алгоритм Томаса Миколова [6].

Сьогодні для створення інтелектуальних систем у дослідників, які працюють з природною мовою, є багато інструментів (див. таблицю 1.1), які можна умовно розбити на три класи: методи роботи з індивідуальними словами, методи роботи з реченнями та методи обробки довільних текстів з кількох речень.

Таблиця 1.1

Пакети для NLP та глибокого навчання

Пакет	Мова	Стати-стична лінгвістика	Векторні представ-лення	Глибоке навчання	Джерело
Stanford Core NLP	Java	∨			stanfordnlp.github.io/CoreNLP
NLTK	Python	∨			www.nltk.org
OpenNLP	Java	∨			opennlp.apache.org
word2vec	C		∨		code.google.com/archive/p/word2vec
gensim	Python		∨		radimrehurek.com/gensim
Spark MLib	Java/ Scala/ Python		∨		spark.apache.org/docs/latest/ml-guide.html
Glove	C		∨		nlp.stanford.edu/projects/glove
Google Tensorflow	Python/ C++			∨	www.tensorflow.org
Theano	Python			∨	github.com/Theano/Theano

Традиційно слова речення оброблялися як елементи безлічі слів зі словника, але при цьому виникали серйозні складнощі: якщо враховувати різні розмовні форми, наприклад технічний жаргон, то обсяг слів, навіть в англійській мові, ставав величезним і складання повного семантичного словника для широкої сфери застосування – завданням дуже трудомістким. Велику популярність отримав алгоритм word2vec, що входить до багатьох стандартних пакетів машинного навчання і навчається якісним представленням слів на великих нерозмічених корпусах (множині різноманітних текстів з різних тем, написаних у різних жанрах та стилях). На відміну від традиційних представлень слів, тут використовується нейроймовірна модель мови (звідси і зв'язок з глибоким навчанням) – кожне слово представляється вектором з речових чисел в маленькому (щодо розміру повного словника) просторі, наприклад розмірністю в 300 вимірювань. Спочатку векторам надаються випадкові значення. Далі у процесі навчання для слова підбирається вектор, максимально схожий на вектори інших слів, які у схожих контекстах. В якості контекста береться невелике вікно попередніх і наступних слів, наприклад, у п'ять слів.

Водночас у завданнях обробки текстів глибокі моделі використовуються рідко – наприклад, у word2vec не використовується глибока нейромережа, проте цей алгоритм вкладається у парадигму глибокого навчання: він сам знаходить ознаки в режимі навчання без вчителя, тим самим нагадуючи процес навчання у мозку людини [7]. Наприклад, вивчаючи нову мову, кілька разів зустрічається невідоме слово і спочатку не розуміється його значення, але потім людина починає здогадуватися про сенс контексту його застосування. При цьому людина не може дати строгого визначення слова, а оперує інтуїтивними поняттями близькості та схожості.

Стандартний алгоритм word2vec не дозволяє вирішити питання, пов'язані з омонімією. Наприклад, в англійській мові приблизно в 40% випадків використовуються омонімічні слова, що мають різні значення при одному написанні («машина»: автомобіль, комп'ютер, механічний пристрій). Це дуже складне завдання при обробці природної мови, і для її вирішення запропоновано

модифікацію алгоритму word2vec з метою автоматичного визначення омонімів та створення окремих векторів для окремих сенсів, а також процедури визначення правильного сенсу омоніму щодо заданого контексту.

Як відомо, більшість методів обробки природної мови успішно використовують лише представлення слів, ігноруючи синтаксис та семантику, які можна вивести із синтаксичної структури речень. Така модель представлення текстів називається bag of words – простий набір слів без урахування їх порядку. Наприклад, у разі векторних представлень можна об'єднати в кластери вектори слів корпусу, на яких тренується модель, та використовувати такі кластери для задач простої класифікації. Але якщо завдання полягає у добуванні більш якісних семантичних представлень, то знадобляться інструменти обробки текстів, що працюють із синтаксичною структурою речень або хоча б не ігнорують порядок слів у реченні [8].

Інструменти роботи з синтаксисом помітно прогресували за останні роки — лексикалізовані ймовірні граматики значно підвищили якість синтаксичного розбору, а зручніші для багатьох випадків граматики залежностей досягли якості, достатньої для вирішення великого класу завдань обробки текстів. Крім того, за останні кілька років у сотні разів зросла ефективність алгоритмів, що відновлюють синтаксис. Наприклад, якщо ще недавно граматичний парсер Стенфорда обробляв кілька речень на секунду на комп'ютері стандартної архітектури, то сьогодні можна за хвилину отримати граматику складових та граматику залежностей для всього корпусу Penn Treebank (1 млн лінгвістично розмічених речень, які зазвичай використовуються для навчання).

Методи глибокого навчання надають можливість іншого підходу до роботи з реченнями – моделювання речення як послідовності векторів, отриманих методом з класу word2vec, і використання його в алгоритмах машинного навчання [7]. Стандартні алгоритми машинного навчання працюють з фіксованим набором атрибутів, і їх не можна пристосувати до такої моделі, але з нею добре справляються рекурентні нейронні мережі, які на вході приймають одне слово у векторному поданні та мають кілька внутрішніх рівнів, а на виході будують

класифікатор чи регресор. Але, на відміну від звичайних нейромереж, внутрішні рівні рекурентної мережі (а іноді й верхній рівень) підключені назад у мережу, тобто стан мережі, який вона перейшла на попередньому слові, буде передано в мережу як додатковий вхід на наступному слові. Таким чином, у нейромережі з'являється «пам'ять», що дозволяє їй послідовно обробляти слова з речення і робити передбачення окремо щодо кожного слова або всього речення відразу. Інакше кажучи, мережі послідовно надається одне слово речення за іншим, а мережа використовує свій попередній стан визначення поточного кроку. Однак на практиці прості рекурентні мережі працюють не дуже добре через те, що пам'ять про попередні слова речення швидко втрачається при тренуванні та експлуатації мережі. Тому зазвичай використовуються спеціальні елементи пам'яті – LTSM (Long Term Short Memory), що є безліччю нейронів і керуючих елементів, які визначають, коли слід записувати, читати та очищати пам'ять [9]. Ці елементи дозволяють пам'яті не змінюватися протягом тривалого послідовного обчислення і дозволяють правильно атрибутувати помилку під час навчання. Керуючі елементи також навчаються у процесі навчання нейромережі.

Рекурентні нейромережі, особливо з LTSM, добре себе зарекомендували при вирішенні різних завдань, від моделювання мови до машинного перекладу, але цей клас мереж має істотний недолік — вони використовують тільки порядок слів у реченні, і їх не можна змусити працювати з граматичними структурами, отриманими традиційними інструменти. По суті, рекурентним мережам доводиться для кожного завдання з нуля «навчати» граматику мови. Крім того, рекурентна мережа не будує представлень для проміжних фраз, тому для завдань, в яких потрібні якісні представлення різних фраз, що становлять речення, використовуються рекурсивні нейронні мережі [10].

На відміну від рекурентних, рекурсивні мережі працюють не поверх послідовності слів у реченні, а на основі граматики залежностей речення – для кожної речення будується бінарне дерево для його розбору. Роботу рекурсивної мережі можна уявити так. Спочатку вона обробляє листя дерева розбору (листя дерева — вказівники на два слова речення і тип граматичної залежності між

ними), заміщаючи листя отриманим вектором тієї ж розмірності, як і вектора слів. І продовжує працювати далі, але тепер листя вже поєднує фрази, а не слова — будуються векторні представлення фраз речення. Отже, маючи дерево розбору, можна побудувати рекурсивну мережу з такою самою топологією, як і дерево, замінивши кожен вузол дерева на нейронну мережу. Звичайно, всі розмножені мережі мають загальні параметри, тобто під час навчання та експлуатації доводиться працювати з однією мережею. Пророцтва як класифікації чи регресії можуть відбуватися поверх будь-яких вузлів розмноженої мережі, включаючи верхній вузол [11].

При навчанні рекурсивна мережа може навчитися робити якісні представлення як повних речень, так і всіх фраз речення. У цьому нейромережа може послабити ефект помилок граматичного розбору, особливо які впливають на завдання, де навчається рекурсивна нейромережа. Таким чином, ми отримуємо міру семантичної близькості як для слів, так і для всіх фраз у реченні. При цьому до рекурсивної нейронної мережі можна додати елементи пам'яті LTSM і отримати дуже якісні векторні представлення.

Інший підхід для отримання векторів речень полягає в тому, що для кожного речення, параграфа або цілого документа тренується окремий вектор, який також бере участь у пророкуванні контексту кожного слова речення або параграфа, і в процесі навчання вибираються вектори, що найбільше покращують прогнози. За якістю отриманих векторів цей метод (його зазвичай називають doc2vec) змагається з рекурсивними нейромережами, при цьому для навчання не потрібна розмічена навчальна вибірка [12]. Правда, цей метод має два істотні недоліки: йому потрібні великі речення або цілі параграфи — він не працює на рівні коротких фраз; він обчислювально дорожчий, ніж нейромережі, кожен вектор речення оптимізується окремо.

Слід згадати ще про два підходи до моделювання слів та речень — згорткові нейромережі та нейромережі, що працюють із символьними представленнями слів або змішаними представленнями. Зазвичай у згорткових нейромережах на вхід мережі подається відразу все речення у вигляді матриці векторних представлень

окремих слів. Згорткова мережа обробляє довільно довгу послідовність підмережами фіксованого розміру, які застосовуються на послідовності вікон поверх вхідних даних. Таким чином емулюється операція згортки, причому мережа навчається самому фільтру, що використовується в пакеті. Пакувальні мережі також показують гарні результати на рівні рекурсивних мереж.

За допомогою рекурентних та рекурсивних нейромереж можна ефективно вирішувати прості завдання, пов'язані з автоматичною обробкою текстів: класифікації, визначення тональності, виділення іменованих сутностей та простих фактів тощо [13].

У традиційних способах отримання семантики з тексту слід відзначити прогрес у створенні семантичних словників з широким охопленням, таких як ConceptNet і FrameNet, а також методи машинного навчання, що здійснюють прив'язку слів з текстів до словників. Щоправда, у випадку FrameNet, який містить у собі різні семантичні ролі, якість автоматичної прив'язки ще досить низька – точність не перевищує 60%.

Як тільки йдеться про обробку текстів, що складаються з кількох речень, які потрібно розглядати не як незалежні сутності, а як взаємопов'язану послідовність висловлювань, всі наявні технології стикаються з серйозними проблемами. У цьому випадку виникає семантичний контекст, який збагачується та модифікується наступними реченнями, та моделювати його дуже складно. У комп'ютерній лінгвістиці є просте і краще формалізоване завдання — вирішення кореферентності або анафоричних відносин. На сьогоднішній день навіть це завдання не вирішене, а наявні методи поки що дають незадовільні результати. Проте якщо сильно обмежити прикладну область, можна будувати цілком прийнятні семантичні моделі для текстів, щоскладаються з кількох речень і діалогів [14].

Завдяки новим методам глибокого навчання сьогодні можна отримати якісні семантичні представлення для слів, фраз і речень, навіть без навчальної вибірки. Дедалі менше зусиль зараз потрібно для створення власних семантичних словників та баз знань, тому розробляти системи автоматичної обробки текстів

стало простіше. Усі відомі сьогодні методи успішно працюють або під час вирішення завдань «поверхневого» розуміння мови, або за суттєвого обмеження предметної області. Наприклад, у компанії Toprater.com за допомогою різних інструментів для автоматичної обробки текстів вирішується завдання щодо виявлення оцінних суджень у відгуках про об'єкти електронної комерції щодо широкого набору властивостей об'єкта.

1.2 Семантичний аналіз інформації

Семантичний аналіз – важке математичне завдання, вирішення якого застосовується у процесі створення штучного інтелекту, ускладнюється необхідністю обробки природної мови. Складність у тому, що комп'ютер не вміє правильно пояснювати образи, які людина передає за допомогою символів. Дані якісного семантичного аналізу можуть використовуватись у торгівлі для аналізу попиту на товари за отриманими відгуками, у пошукових системах, системах автоматичного перекладу та ін [15].

Концептуальний аналіз є невід'ємною частиною семантичного аналізу. На вхід семантичного компонента має надійти синтаксично розмічений текст. Текст повинен бути помічений і в ньому повинна бути різна інформація: основні поняття, що відповідають слову або терміну, вид синтаксичного зв'язку та ін. Для того, щоб передати семантичний компонент, повинні бути виявлені терміни – словосполучення, власні імена, представлення числової інформації.

Прецедентний аналіз – це аналіз, який ґрунтується на використанні корпусу попередньо розмічених текстів. Якщо система аналізу побудована грамотно, вона повинна допомогти отримати знання з конкретного тексту, і навіть накопичити результати, як у синтаксичному, так і на семантичному рівні. У ході семантичної розмітки практично всім словам з тексту присвоюється одна або кілька семантичних і словотворчих ознак, наприклад, «person», «substance», «space», «speed», «movement». Таку розмітку можна зробити автоматично за допомогою

програми Semmarkup. Вона здійснюється відповідно до семантичного словника корпусу.

Моделі семантичного аналізу тексту [16]:

— Тезаурус – лексичний словник, де показані семантичні відносини між лексичними одиницями. Завдяки цьому словнику можна зрозуміти сенс, не лише за допомогою визначення, а й співвіднісши слова з іншими поняттями та їхніми групами, завдяки чому можна використовувати штучний інтелект для того, щоб наповнити бази знань. Зазвичай у тезаурусах використовують такі семантичні відносини як, синоніми, антоніми, гіпоніми, гіпероніми, мероніми, холоніми та пароніми. WordNet можна навести як приклад тезаурусу. Синсет (синонімічний ряд) є основною словниковою одиницею WordNet, він поєднує слова зі схожими значеннями. Синсети це слова однієї і тієї ж частини мови, що і слово, яке вводиться. Кожне таке слово має свою дефініцію, яка пояснює значення цього слова. Наприклад, ручка (pen) в такому словнику має різні значення: crayon, pencil, marker.

— Семантична мережа – модель предметної області, має вигляд орієнтованого графа, вершини якого відповідають об'єктам предметної області, а дуги (ребра) ставлять відносини з-поміж них. Семантична мережа з'явилася на основі математичних формул. Семантична мережа є схемою, де є концепти предметів, подій, станів, за допомогою ліній показано відношення між концептами.

— Фреймові моделі – це структура, яка потрібна для того, щоб описати поняття чи ситуації з характеристик цієї ситуації та його значень. Фрейм можна вважати елементом семантичної мережі.

— Онтологічна модель – це детальний опис якоїсь предметної чи проблемної галузі, що може використовувати формулювання

тверджень загального характеру. Ця модель допомагає створювати поняття, які згодом стають придатними для машинної обробки.

З вищесказаного можна дійти висновку, що у століття комп'ютеризації, питання семантичного аналізу тексту стає дедалі популярним. Через це область автоматичної обробки текстів сфокусувалася на прикладному аспекті. Семантичний аналіз є одним із складних математичних завдань, незважаючи на затребуваність практично у всіх сферах життя сучасної людини. Головним завданням семантичного аналізу – це зробити так, щоб комп'ютер зміг коректно трактувати образи, що передаються користувачам.

1.3 Технології великих даних

Big Data або великі дані – це структуровані або неструктуровані масиви великого обсягу даних. Їх обробляють за допомогою спеціальних автоматизованих інструментів, щоб використовувати для статистики, аналізу, прогнозів та прийняття рішень.

Сам термін «Big Data» запропонував редактор журналу Nature Кліффорд Лінч у спецвипуску 2008 [17]. Він говорив про вибухове зростання обсягів інформації у світі. До великих даних Лінч відніс будь-які масиви неоднорідних даних понад 150 Гб на добу, проте єдиного критерію досі немає.

До 2011 року аналізом великих даних займалися лише у рамках наукових та статистичних досліджень. Але до початку 2012-го року обсяги даних зросли до величезних масштабів, і виникла потреба в їх систематизації та практичному застосуванні.

З 2014 року на Big Data звернули увагу провідні світові вузи, де навчають прикладним інженерним та IT-спеціальностям. Потім до збору та аналізу підключилися IT-корпорації – такі, як Microsoft, IBM, Oracle, EMC, а потім і Google, Apple, Facebook та Amazon. Сьогодні великі дані використовують компанії у всіх галузях.

Компанія Meta Group запропонувала основні характеристики великих даних [18]:

- Volume – обсяг даних: від 150 Гб на добу;
- Velocity – швидкість накопичення та обробки масивів даних. Великі дані регулярно оновлюються, тому необхідні інтелектуальні технології для їх обробки в режимі онлайн;
- Variety – різноманітність типів даних. Дані можуть бути структурованими, неструктурованими або структурованими частково. Наприклад, у соцмережах потік даних не структурований: це можуть бути текстові пости, фото чи відео.

Сьогодні до цих трьох додають ще три ознаки:

- Veracity – достовірність як самого набору даних, так і результатів його аналізу;
- Variability – мінливість. У потоків даних бувають свої піки та спади під впливом сезонів чи соціальних явищ. Чим нестабільніший і мінливіший потік даних, тим складніше його аналізувати;
- Value – цінність чи значимість. Як і будь-яка інформація, великі дані можуть бути простими або складними для сприйняття та аналізу. Приклад простих даних – це пости у соцмережах, складних – банківські транзакції.

Сучасні обчислювальні системи забезпечують миттєвий доступ до масивів великих даних. Для їх зберігання використовують спеціальні дата-центри із найпотужнішими серверами.

Крім традиційних, фізичних серверів використовують хмарні сховища, «озера даних» (data lake – сховища великого обсягу неструктурованих даних з одного джерела) та Hadoop – фреймворк, що складається з набору утиліт для розробки та виконання програм розподілених обчислень. Для роботи з Big Data

застосовують передові методи інтеграції та управління та підготовки даних для аналітики [19].

Завдяки високопродуктивним технологіям, таким як грід-обчислення або аналітика в оперативній пам'яті, компанії можуть використовувати будь-які обсяги великих даних для аналізу. Іноді Big Data спочатку структурують, відбираючи ті, що потрібні для аналізу. Все більші дані застосовують для завдань у рамках розширеної аналітики, включаючи штучний інтелект.

Виділяють чотири основні методи аналізу Big Data [20]:

1. Описова аналітика (descriptive analytics) – найпоширеніша. Вона відповідає на питання «Що сталося?», аналізує дані, які у реальному часі, так і історичні дані. Головна мета – з'ясувати причини та закономірності успіхів чи невдач у тій чи іншій сфері, щоб використовувати ці дані для найефективніших моделей. Для описової аналітики використовують основні математичні функції. Типовим прикладом є соціологічні дослідження або дані веб-статистики, які компанія отримує через Google Analytics.

2. Прогнозна або предикативна аналітика (predictive analytics) – допомагає спрогнозувати найімовірніший розвиток подій на основі наявних даних. Для цього використовують готові шаблони на основі якихось об'єктів або явищ з аналогічним набором характеристик. За допомогою предикативної (прогнозної) аналітики, наприклад, можна прорахувати обвал або зміну цін на фондовому ринку. Або оцінити можливості потенційного позичальника із виплати кредиту.

3. Директивна аналітика (prescriptive analytics) – наступний рівень порівняно з прогнозною. За допомогою Big Data та сучасних технологій можна виявити проблемні точки у бізнесі чи будь-якій іншій діяльності та розрахувати, за яким сценарієм їх можна уникнути в майбутньому.

4. Діагностична аналітика (diagnostic analytics) – використовує дані, щоб проаналізувати причини того, що сталося. Це допомагає виявляти аномалії та випадкові зв'язки між подіями та діями.

Наприклад, Amazon аналізує дані про продажі та валовий прибуток для різних продуктів, щоб з'ясувати, чому вони принесли менше доходу, ніж очікувалося.

Дані обробляють та аналізують за допомогою різних інструментів та технологій:

— Спеціальне ПЗ: NoSQL, MapReduce, Hadoop, R;

— Data mining – вилучення з масивів раніше невідомих даних за допомогою великого набору техніки;

— ШІ (штучний інтелект) та нейромережі – для побудови моделей на основі Big Data, включаючи розпізнавання тексту та зображень. Наприклад, за допомогою Big Data та штучного інтелекту компанія аналізує клієнтський досвід та пропонує персоніфіковані продукти та сервіси;

— Візуалізація аналітичних даних — анімовані моделі чи графіки, створені з урахуванням великих даних.

Hadoop – це проект із відкритим програмним кодом, розроблений фондом Apache Software Foundation. Hadoop використовується для розподіленої (паралельної) обробки великого обсягу даних. За своєю суттю Hadoop представляє цілий набір простих програм, утиліт та бібліотек для вирішення розподілених завдань, заснованих на кількох кластерах із тисяч вузлів. Порівняно простий дизайн інфраструктури Hadoop забезпечив його популярність та масштабність використання, навіть у разі виконання завдання на тисячах обчислювальних машин, кожна з яких має власні можливості обробки та зберігання даних.

Даних на незалежні блоки, які обробляються паралельно відповідно до картки завдань («to map» – планувати; складати схему, карту). Таким чином, основне завдання з обробки вхідних даних ділиться на підзавдання між паралельними процесами. При цьому кожне підзавдання вирішується в строго визначеному напрямку, щоб отримані дані в результаті множинного аналізу

відповідали один одному. Результати підзадач об'єднуються у підсумку відповідно до основного завдання на етапі згортання («to reduce» – скорочувати, згортати).

У сукупності вони утворюють своєрідний контролер і під час планування завдань на ведених пристроях, стежать за виконанням і відправляють на повторну обробку невирішені завдання. Ведені пристрої виконують завдання відповідно до вказівок контролера;

1) HBase – розподілена база даних NoSQL, заснована на моделі Google BigTable, яка використовує HDFS як носій. Основне завдання HBase – розміщення таблиць з великою кількістю рядків (мільярди рядків та мільйони стовпців) на кластерах стандартного обладнання. Ця утиліта використовується у програмах Hadoop, які вимагають довільні операції читання/запису для великого обсягу даних або для програм з великою кількістю користувачів. HBase складається з трьох основних компонентів: клієнтської бібліотеки, головного та кількох розподілених серверів. Є самостійною утилітою Hadoop;

2) Hive – платформа зберігання даних, що використовується для читання, внесення записів та управління великими обсягами даних із розподіленого сховища;

3) Oozie – інструмент для управління робочим процесом та координації завдань MapReduce. Він дозволяє об'єднати кілька завдань у єдине логічне завдання всього робочого процесу;

4) Whirr – бібліотека з алгоритмами управління хмарними сервісами;

5) Zookeeper – служба координації розподілених програм екосистеми Hadoop. Її завдання полягає у підтримці, налаштуванні та присвоєнні імені великому обсягу даних. Також забезпечує розподілену синхронізацію та визначення групових завдань. Zookeeper включає головний та підлеглий вузли, зберігає інформацію про налаштування. Служба орієнтована на виявлення конфліктуючих завдань усередині системи, нераціонального використання ресурсів;

— Рівень зберігання даних у розподілених реєстрах.

— Рівень запиту на дані. На цьому рівні задіяні утиліти, відповідальні за запит необхідних даних у рамках поставленого завдання та подальшої передачі їх на рівень обробки даних.

1.4 Огляд існуючих програмних аналогів

Завданням дипломної роботи є розробка програмного забезпечення семантичного аналізу економічних новин для прогнозування розвитку економіки на основі технологій обробки великих даних. Зважаючи на це, було проведено дослідження різних програм даного типу.

1) Critical Mention (рис.1.1.) [21].

Critical Mention аналізує новини та інші публікації, які посилаються на обраний вид бізнесу. Оскільки висвітлення новин тепер працює цілодобово, 7 днів на тиждень, це допомагає мати програмне забезпечення, яке може контролювати Інтернет і попереджати про будь-які тенденції в новинах, які створює обраний вид бізнесу.

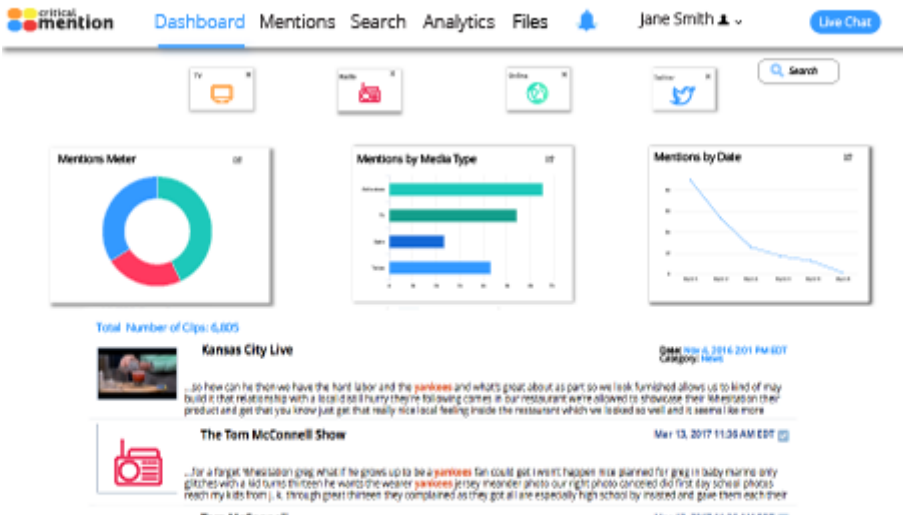

Рис. 1.1. Інтерфейс Critical Mention

Critical Mention може навіть попередити про сюжети, які з'являються на телебаченні. Можливо шукати у відеофайлах згадки про конкретну компанію та знімати відео, щоб поділитися ними з іншими співробітниками. Це може

допомогти створити ефективний онлайн-контент, який використовує сучасні маркетингові можливості.

2) Sentiment Analyzer (рис.1.2.) [22].

Працювати з Sentiment Analyzer просто. Потрібно перейти на їх сайт, скопіювати текст, який потрібно проаналізувати, і вставити текст у поле. Вибрати «Аналізувати!» і веб-сайт оцінить текст і дасть «оцінку настроїв».

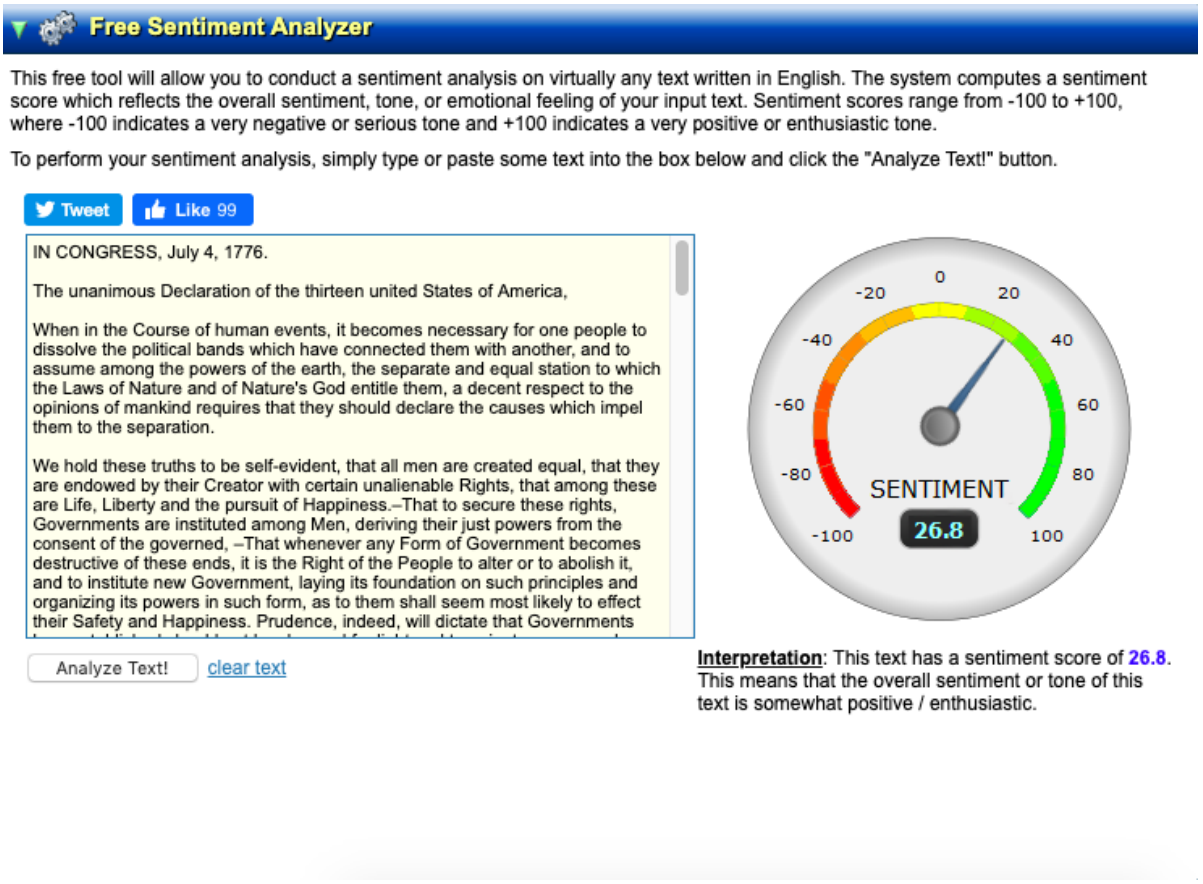

Рис. 1.2. Інтерфейс Sentiment Analyzer

Sentiment Analyzer використовує «комп'ютерну лінгвістику та аналіз тексту», щоб визначити настрої, які стоять за фрагментом тексту. Потім він поєднує та порівнює свої висновки, щоб отримати загальну оцінку. Це робить його чудовим інструментом для компаній, які прагнуть швидко розшифрувати намір, що стоїть за незрозумілою відповіддю від клієнта.

3) SAS Visual Analytics (рис.1.3.) [23].

Будучи малим підприємством, більше не є перешкодою для отримання ринкової та бізнес-аналітики, за словами SAS, лідера програмного забезпечення та

послуг бізнес-аналітики з 1976 року. SAS перетворює дані на знання, які допомагають приймати рішення та дають новий погляд на бізнес, будь то мале, середнє чи велике підприємство.

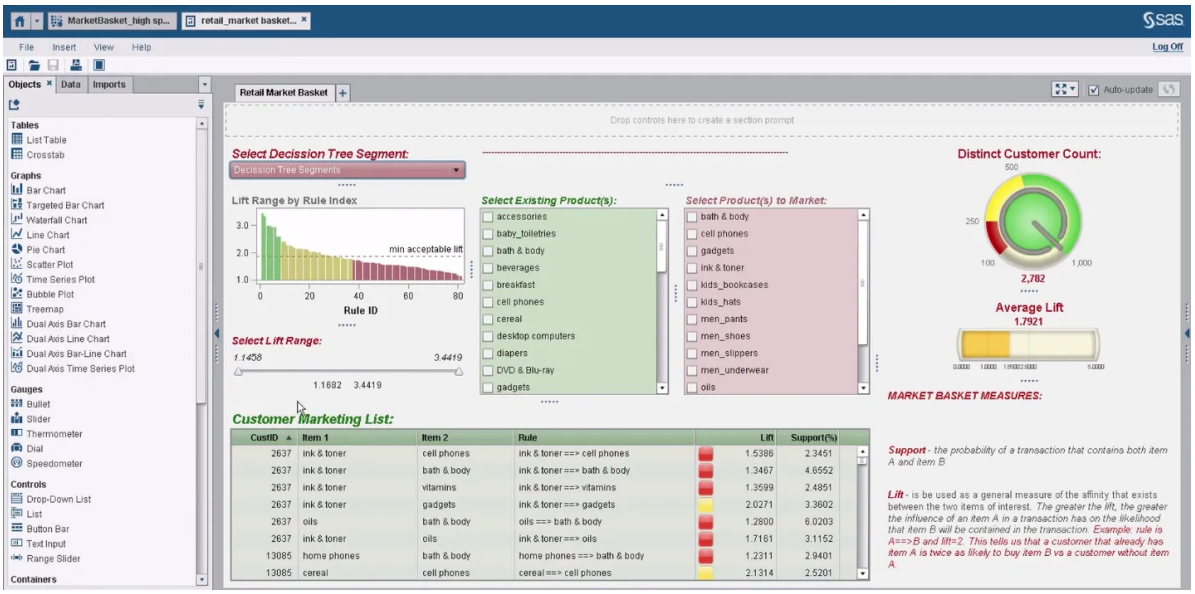

Рис. 1.3. Інтерфейс SAS Visual Analytics

Малий і середній бізнес (МСБ) стикається з багатьма з тих же проблем, що і великі підприємства. Проста у використанні аналітика, автоматизоване прогнозування та аналіз даних SAS дають змогу підприємствам, які не мають великих ресурсів, досягати більшого за менші витрати. Ця аналітика допомагає компаніям долати труднощі для зростання та конкуренції. Повідомлення SAS для малого та середнього бізнесу просте: «Визначте, що працює. Виправте те, чого немає. І відкрийте нові можливості».

4) Alteryx (рис.1.4.) [24].

Alteryx пропонує передові інструменти аналізу даних та аналітики, які також представляють інформацію простим і зрозумілим способом.

Alteryx поєднує внутрішні дані бізнесу з загальнодоступною інформацією, щоб допомогти приймати кращі бізнес-рішення. Ці ідеї дозволяють створювати графіки, сюжетні лінії та інтерактивні візуальні елементи з інформаційної панелі. Він також пропонує функції спільної роботи, які дозволяють обговорювати команди.

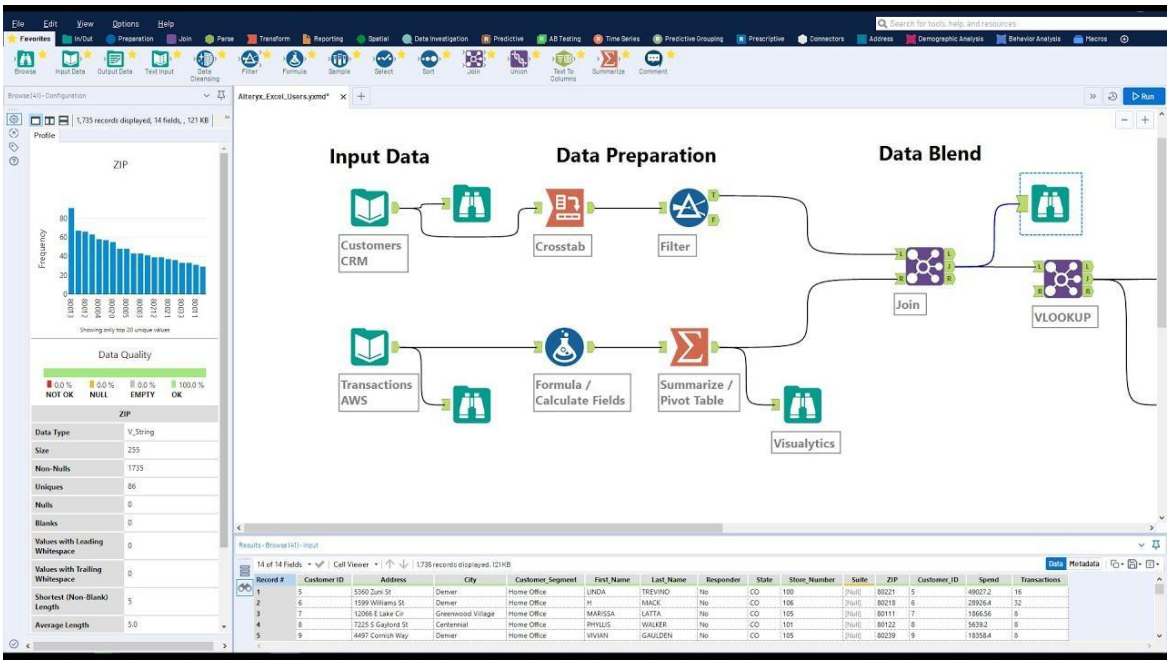

Рис. 1.4. Інтерфейс Alteryx

На додаток до бізнес-даних, Alteryx може надавати дані окремого відділу, включаючи маркетинг, продажі, операції та аналітику клієнтів. Платформа також охоплює широкий спектр галузей, таких як роздрібна торгівля, продукти харчування та напої, засоби масової інформації та розваги, фінансові послуги, виробництво, споживчі упаковані товари, охорона здоров'я та фармацевтика.

5) Google Analytics (рис.1.5.) [25].

Не потрібне вишукане дороге програмне забезпечення, щоб почати збирати дані. Це може початися з активу, який вже є у багатьох компаній – власного веб-сайту. Google Analytics, безкоштовна платформа цифрової аналітики Google, надає малому бізнесу інструменти для аналізу даних веб-сайту з усіх точок дотику в одному місці.

За допомогою Google Analytics можливо отримувати довгострокові дані, щоб виявити тенденції та іншу цінну інформацію, щоб приймати зважені рішення на основі даних. Наприклад, відстежуючи та аналізуючи поведінку відвідувачів – наприклад, звідки надходить трафік, як аудиторія залучає та як довго відвідувачі залишаються на веб-сайті (відомий як показник відмов), можливо приймати кращі рішення, щоб відповідати вимогам веб-сайту чи інтернет-магазину.

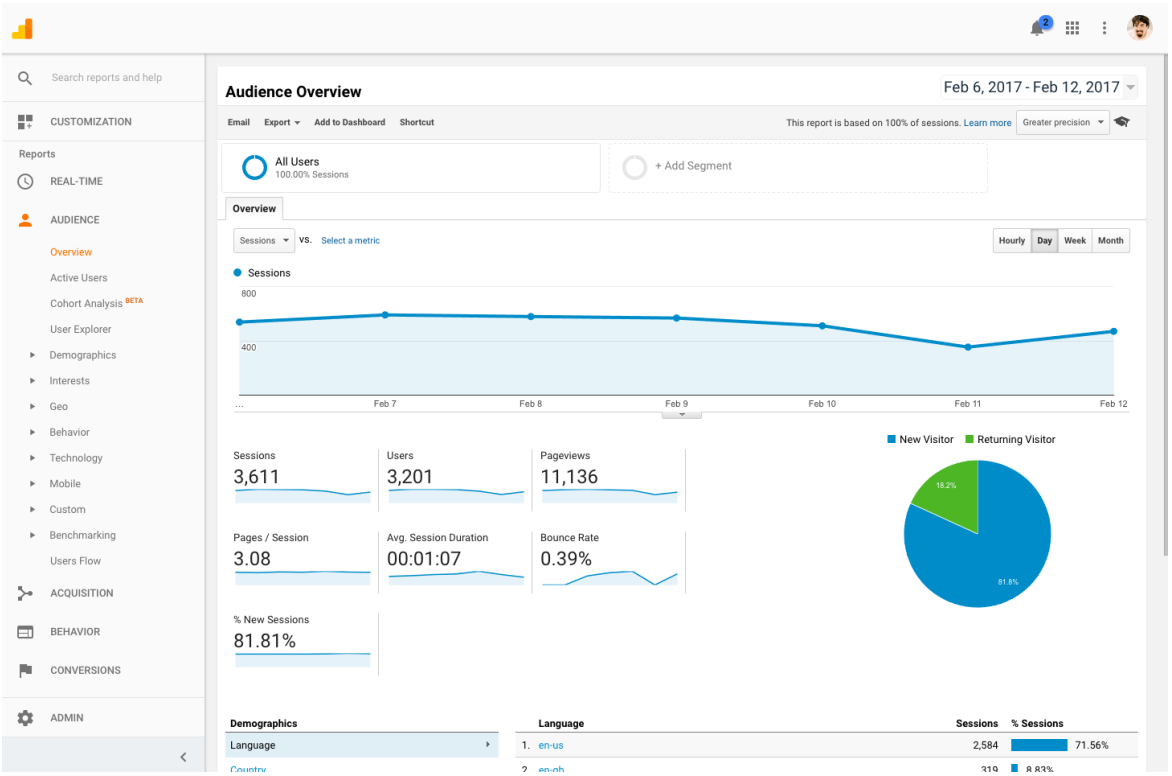

Рис. 1.5. Інтерфейс Google Analytics

Також можливо аналізувати трафік соціальних мереж, що дає змогу вносити зміни в маркетингові кампанії в соціальних мережах на основі того, що працює, а що не працює. Вивчення відвідувачів із мобільних пристроїв може допомогти отримати інформацію про клієнтів, які переглядають сайт на своїх мобільних пристроях, щоб забезпечити кращий мобільний досвід.

6) IBM Cognos Analytics (рис.1.6.) [26].

Хоча багато рішень для великих даних створені для надзвичайно обізнаних науковців та аналітиків, Cognos Analytics від IBM робить передову та прогнозну бізнес-аналітику легкодоступною для бізнесу. Платформа не вимагає будь-яких навичок використання складних систем інтелекту та аналізу даних; натомість він автоматизує процес. Це аналітичне рішення самообслуговування включає в себе набір послуг доступу, уточнення та зберігання даних, що надає інструменти для самостійної підготовки та представлення даних у простий і ефективний спосіб, щоб керувати рішеннями.

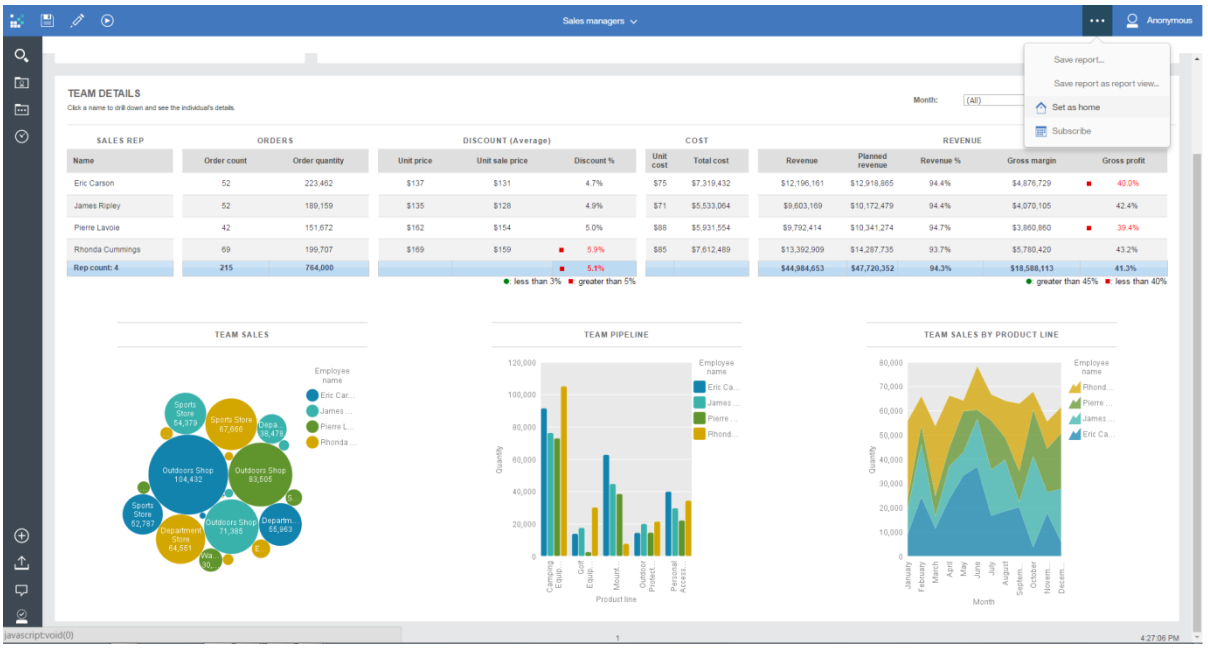

Рис. 1.6. Інтерфейс IBM Cognos Analytics

На відміну від багатьох аналітичних рішень, які зосереджені на одній сфері бізнесу, IBM Cognos Analytics об’єднує всі проекти аналізу даних в єдину платформу. Є можливість використовувати його для всіх типів аналізу даних, включаючи маркетинг, продажі, фінанси, людські ресурси та інші частини операцій. Його технологія «обробки природної мови» допомагає виявляти проблеми, розпізнавати закономірності та отримати значущі ідеї, щоб відповісти на ключові запитання, наприклад, що в кінцевому підсумку стимулює продажі, які угоди, ймовірно, будуть закриті та як зробити співробітників щасливими.

Висновки до розділу 1

Аналітика великих даних привернула широку увагу як наукових кіл, так і промисловості та бізнесу, оскільки зростає попит на розуміння тенденцій у масивних наборах даних. Останні розробки сенсорних мереж, кібер-фізичних систем і повсюдність Інтернету речей (IoT) збільшили збір даних (включаючи охорону здоров’я, соціальні медіа, розумні міста, сільське господарство, фінанси, освіту тощо) до величезних масштабів. Однак зібрані дані, за своєю суттю є

невизначеними через шум, неповноту та невідповідність. Аналіз таких величезних обсягів даних вимагає передових аналітичних методів для ефективного перегляду та/або прогнозування майбутніх дій з високою точністю та передовими стратегіями прийняття рішень. Зі збільшенням кількості, різноманітності та швидкості даних збільшується невизначеність, яка притаманна їм, що призводить до відсутності впевненості в результуючому процесі аналітики та прийнятих у ньому рішень. У порівнянні з традиційними методами та платформами даних, методи штучного інтелекту (включаючи машинне навчання, обробку природної мови та обчислювальний інтелект) забезпечують більш точні, швидші та масштабовані результати в аналітиці великих даних.

РОЗДІЛ 2
СЕМАНТИЧНИЙ АНАЛІЗ ТЕКСТУ ТА ПЕРЕДБАЧЕННЯ ЧАСОВИХ РЯДІВ

2.1 Попередня обробка тексту для семантичного аналізу

Аналіз настроїв – це тема зростаючого інтересу. Одне з його основних завдань – виявити емоційну полярність інформації та з'ясувати, чи виражає вона позитивні, негативні чи нейтральні емоції.

Здебільшого процес аналізу почуттів поділяється на три фази. Перший етап – попередня обробка тексту з наступним застосуванням методів аналізу та алгоритмів аналізу на підготовленому тексті. Ця фаза включає ідентифікацію, класифікацію, аналіз почуттів та ін. Результати, отримані на цьому етапі, далі обробляються в зрозумілій формі, перш ніж представляти їх, як показано на рис. 2.1. [27].

Рис.2.1 Процес попередньої обробки тексту [27]

Аналіз даних значною мірою залежить від їхньої якості та способу їх організації. Алгоритми машинного навчання навчаються на даних. Дуже важливо постачати їм потрібні дані для вирішення проблеми. Навіть якщо дані придатні, необхідно ще раз переконатися в тому, що вони знаходяться в потрібному масштабі, форматі, а також у тому, що вони містять значні частини. Дані завжди «забруднюються», і їх потрібно відфільтровувати або додавати відсутні значення. Процес попередньої обробки тексту включає різні етапи [27].

1. Вибір даних

Він полягає у виборі підмножини всіх доступних даних. Завжди є бажання включити всі дані, однак це не так у більшості випадків. Завжди є символи чи слова, які можна відкинути із даних. Потрібно мати правильне розуміння тексту чи даних та мету його аналізу. Потрібно зрозуміти формат, в якому доступні дані, відсутні дані та надмірність доступних даних. Усі ці фактори впливають на вибір методології.

2. Попередня обробка даних

Існує три основні етапи попередньої обробки даних: форматування, очищення та вибірка. Форматування включає зміну формату даних або тексту. Текст може бути у формі, яка не відповідає або не сумісна з програмою. Він має бути виправлений. Наприклад, дані можуть бути в у власному форматі файлу, а потрібно їх мати в реляційній базі даних або текстовому файлі.

Очищення включає видалення або виправлення деяких відсутніх даних. Можливо, дані можуть містити деякі неповні екземпляри, а не всі дані, необхідні для обробки. Можливо, ці екземпляри потрібно видалити. Крім того, деякі атрибути, які необхідно повністю видалити з даних, можуть містити інформацію. Семплювання передбачає меншу репрезентативну вибірку даних, яка може бути оброблена набагато швидше для вивчення та прототипування рішень, перш ніж буде розглянуто весь набір даних.

Існує кілька способів оцінити важливість кожного набору даних.

Важливим серед них є використання частоти появи набору даних набору документів, що описуються за допомогою рівняння:

$$FP = FF * LOG (N / DF), \quad (2.1)$$

де FF – частота появи набору даних. N вказує на кількість документів, а DF – кількість документів, що містять цей набір даних.

3. Перетворення даних

Це важливий етап підготовки тексту перед його аналізом. Текст, відповідно до його типу, має певну структуру, яка в деяких випадках не дозволяє алгоритмам або машині правильно її аналізувати.

Важливість попередньої обробки [28]:

1. Текст може мати певні символи, які відволікають процеси аналізу тексту.

2. Для ефективного аналізу тексту необхідно звернути увагу на кожне слово, його характеристики та його групу.

3. Кожен алгоритм або програма передбачає власну структуру і конструкцію для аналізованого тексту. Текст має бути підготовлений відповідним чином.

4. У разі використання штучного інтелекту програми також обробляють певну структуру тексту та аналізують її.

5. Ефективним буде лише аналіз належним чином структурованого та підготовленого тексту.

6. Необхідно зрозуміти мету аналізу тексту та типи алгоритмів, які будуть застосовуватись на наступному етапі, таких як класифікація, метод опорних векторів SVM тощо.

7. Алгоритми машинного навчання навчаються з урахуванням даних. Дуже важливо, щоб їм подавалися правильні дані для проблеми, яку потрібно вирішити.

Процес попередньої обробки даних або тексту залежить від вимог та мети програми. У деяких додатках достатньо провести основне форматування та очищення даних додатків, тоді як в інших додатках потрібні складніші процеси, такі як агрегація даних та їх дискретизація.

Тексти, зібрані з навколишнього світу, зазвичай непослідовні, неповні та зашумлені. В них можуть бути відсутні певні елементи, які потрібні. Зашумлені тексти містять викиди, помилки, неузгоджені елементи можуть містити розбіжності у кодуванні.

Процес підготовки даних у разі аналізу настроїв відрізняється від категоризації тексту. У разі категоризації тексту ціль полягає в тому, щоб визначити тему документа, порівнявши її з деякими вже певними темами. В аналізі настроїв асоціації, залежності та відносини всередині тексту вимагають ідентифікації з використанням різних алгоритмів та методів зіставлення із шаблонами.

2.2 Застосування логістичної регресії для семантичного аналізу

Аналіз тональності текстів є одним із найпоширеніших завдань комп'ютерної лінгвістики. Зі збільшенням користувальницької активності в мережі (новини, соціальні мережі, блоги, форуми, онлайн системи відгуків на фільми, ресторани та ін.) необхідність у цій задачі лише зростає, як і вимоги до її точності.

Базові поняття машинного навчання [29]:

— X – безліч об'єктів (наприклад, тексти природною мовою);

— У – безліч допустимих відповідей (наприклад, тональність тексту: позитивна або негативна);

— у *: $X \to Y$- цільова функція (target function), значення якої відомі лише на кінцевій підмножині X (наприклад, тексти для яких наперед визначена тональність).

Завданням машинного навчання є побудова вирішальної функції (decision function) а: $X \to$ У, яка б наближала цільову функцію. У задачі тональності вирішальна функція повинна ставити тексту певну тональність.

Логістична регресія (Logistic regression) – це метод побудови лінійного класифікатора, що дозволяє оцінювати апостеріорні ймовірності належності об'єктів класам [29].

Імовірність настання події y = 1 обчислюється за формулою:

$$\mathrm{P}\{\mathrm{y} = 1|\mathrm{x}\} = f(z), \tag{2.2}$$

де $f(z)$ – логістична функція (сигмоїда):

$$f(z) = \frac{1}{1+e^{-z}}, \tag{2.3}$$

$$z = \theta^T x = \theta_1 x_1 + \cdots + \theta_n x_n, \tag{2.4}$$

де $x_1, \cdots, x_n$ – незалежні змінні $\theta_1, \cdots, \theta_n x_n$ коефіцієнтів регресії.

Змінні $x_1, \cdots, x_n$ є ознаками об'єкта x (тексту природною мовою) з множини X .

Для зменшення ефекту перенавчання практично часто розглядається логістична регресія з регуляризацією. Регуляризація полягає в тому, що параметри розглядаються як випадковий вектор з певною заданою апріорною щільністю розподілу. В якості тональності текстів природною мовою апріорного розподілу часто виступає багатовимірний нормальний розподіл з нульовим середнім, що відповідає апріорному переконанню про те, що всі коефіцієнти регресії повинні бути невеликими числами, в ідеалі – більшість малозначущих коефіцієнтів повинні бути нулями. У цьому випадку метод називається L2-регуляризованою логістичною регресією. Якщо використовувати розподіл Лапласу, як апріорний, замість нормального, то ця варіація логістичної регресії називається Ll-регуляризованою.

Результати роботи методу логістичної регресії залежать від вибору норми регуляризації (L1 або L2) та параметра регуляризації. В якості параметра регуляризації виступає щільність регуляризації C: чим менше значення параметра C, тим сильніша регуляризація.

2.3 Прогнозування часових рядів

Прогнозування часових рядів полягає у побудові моделі для передбачення майбутніх подій ґрунтуючись на відомих подіях минулого, передбачення майбутніх даних до того, як вони будуть виміряні. Типовий приклад – передбачення ціни відкриття біржі, спираючись на попередню її діяльність [30].

Як показано Боксом і Дженкінсом, моделі часових рядів можуть мати різні форми та представляти різні стохастичні процеси [31]. При моделюванні змін рівня процесу можна виділити три широкі класи, що мають практичну цінність: авторегресійні моделі, інтегральні моделі та моделі ковзного середнього. Ці три класи лінійно залежать від попередніх даних. На їх основі побудовані моделі авторегресійного ковзного середнього (Autoregressive Moving Average, ARMA) та авторегресійного інтегрованого ковзного середнього (Autoregressive Integrated Moving Average, ARIMA). Ці моделі у свою чергу узагальнює модель авторегресійного дробноінтегрованого ковзного середнього (autoregressive fractionally integrated moving average, ARFIMA). Розширення моделей на випадки, коли дані надаються не скалярно, а векторно, називають моделями багатовимірних часових рядів. Для таких моделей у скорочених назвах з'являється буква «v» від слова «vector». Існують розширення моделей на випадок, коли часовий ряд, що досліджується, є веденим для деякого «змушувального» ряду (який, однак, може не бути причиною виникнення досліджуваного ряду). Відмінність від багатовимірного ряду полягає в тому, що змушуючий ряд може бути детермінованим або керуватися дослідником, який проводить експеримент. Для таких моделей у скороченні з'являється буква «x» від «exogenous» (екзогенний, що викликається зовнішніми причинами).

Нелінійна залежність рівня низки від попередніх точок цікава, частково через можливість генерації хаотичних часових рядів. Але головним все ж таки є те, що досвідчені дослідження вказують на перевагу прогнозів, отриманих від нелінійних моделей, над прогнозами лінійних моделей.

Серед інших типів нелінійних моделей часових рядів можна виділити моделі, що описують зміни дисперсії ряду з часом (гетероскедастичність). Такі моделі називають моделями авторегресійної умовної гетероскедастичності (AutoRegressive Conditional Heteroscedasticity, ARCH). До них відноситься велика кількість моделей: GARCH, TARCH, EGARCH, FIGARCH, CGARCH та ін. У цих моделях зміни дисперсії пов'язують із найближчими попередніми даними. Противагою такому підходу є представлення локально мінливої дисперсії, при якому дисперсія може бути змодельована залежною від окремого процесу, що змінюється з часом.

Останнім часом значну увагу здобули дослідження в галузі безмодельного аналізу та методи, що ґрунтуються на вейвлет-перетвореннях (наприклад локально стаціонарні вейвлети). Методи багатомасштабного аналізу розкладають заданий тимчасовий ряд на складові, щоб показати залежність від часу з різним масштабом.

Існує велика кількість варіантів позначення часових рядів. Одним із типових є $X = \{X_1, \cdots, X_n\}$, що позначає ряд із натуральними індексами. Інше стандартне представлення: $Y = \{Y_t : t \in T\}$.

Існують дві групи припущень, за умов яких будується більшість теорій:

— стаціонарність процесу;

— ергодичність (або транзитивність).

Ідея стаціонарності трактується в широкому сенсі, охоплюючи дві основні ідеї: строга стаціонарність і стаціонарність другого порядку (стаціонарність в широкому сенсі). На підставі цього можуть бути побудовані і моделі, і програми, хоча моделі надалі можуть розглядатися як частково задані.

Аналіз часового ряду може проводитись і коли ряд сезонно стаціонарний або нестаціонарний.

Загальний вид авторегресивної моделі задається так:

$$Y_t = a_0 + a_1 Y_{t-1} + a_2 Y_{t-2} + \cdots + a_p Y_{t-p+\epsilon_t}, \quad (2.5)$$

де ϵ_t – джерело випадковість, білий шум. Білий шум має такі властивості:

1. $E\left[\epsilon_t\right] = 0$
2. $E\left[\epsilon_t^{\,2}\right] = \sigma^2$
3. $E\left[\epsilon_t \epsilon_s\right] = 0 \; t \neq s$

У цих припущеннях процес визначений до моментів другого порядку і, за певних умов на коефіцієнти, може бути стаціонарним у широкому сенсі.

Якщо шуми мають нормальне розподілення, їх називають нормальним білим шумом.

2.4 Моделі прогнозування ARIMA та SARIMA

Авторегресійне інтегроване ковзне середнє, чи ARIMA, одна із найбільш широко використовуваних моделей прогнозування для однофакторного прогнозування даних часових рядів.

Методологія Бокса-Дженкінса підбору ARIMA-моделі для конкретного ряду спостережень складається з чотирьох етапів [31]:

— ідентифікація моделі – процес вибору моделі, найкраще відповідної аналізованому реальному процесу;

— використання моделі для прогнозування.

У першу чергу необхідно з'ясувати, чи має досліджуваний ряд властивість стаціонарності. Оцінка стаціонарності вихідного часового ряду здійснюється з

використанням формальних тестів, наприклад, за допомогою розширеного критерію Діккі-Фуллера. Крім цього, при ідентифікації змішаної моделі проводиться аналіз корелограм ряду, для чого будується графік вибіркової автокореляційної функції (ACF). Корелограма стаціонарного часового ряду швидко зменшується зі зростанням порядку поза межами кількох перших значень. Якщо графік зменшується досить повільно, є підстави вважати ряд нестаціонарним; якщо ж не зменшується, то досліджуваний ряд безумовно не стаціонарний.

Реальні процеси можуть не мати властивість стаціонарності, проте за допомогою нескладних процедур часто можна привести спостерігається ряд до стаціонарного процесу. До таких перетворень можна віднести:

— взяття кінцевих різниць виду

$$X_t = \Delta Y_t = Y_t - Y_{t-1}, \tag{2.6}$$

де X_t – перша різниця,

$$Z_t = \Delta X_t = X_t - X_{t-1} = Y_t - 2Y_{t-1} + Y_{t-2} = \Delta 2Y_t, \tag{2.7}$$

Z_t – друга різниця;

— логарифмування ланцюгових індексів виду:

$$X_t = ln(Y_t / Y_{t-1}) = lnY_t - lnY_{t-1}, \tag{2.8}$$

— розрахунок темпів приросту виду

$$X_t = (Y_t - Y_{t-1})/Y_{t-1} = (Y_t / Y_{t-1}) - 1, \tag{2.9}$$

— логарифмування ряду виду

$$X_t = lnY_t, \tag{2.10}$$

— розрахунок темпів зростання виду

$$X_t = \left(Y_t - Y_{t-1}\right), \tag{2.11}$$

та ін.

При виборі процедури перетворення для отримання стаціонарного ряду необхідно виходити з графіка тимчасового ряду X_t. Коректний вибір повинен забезпечувати приблизне виконання умови:

$$X_t = f(Y_t) \approx const, \tag{2.12}$$

Послідовність значень вихідного ряду Y_t є ARIMA(p,d,q) моделлю, якщо послідовність значень проінтегрованого ряду $\Delta^d Y = (\Delta^d Y_t)$ утворює ARMA (p,q)-модель.

ARIMA відбувається на підставі аналізу автокореляційної функції та приватної функції автокореляції з використанням формальних критеріїв, наприклад, інформаційних критеріїв Акаіке і Шварца.

При побудові ARIMA(p,d,q)-моделі тимчасового ряду Y_t необхідно прагнути мінімізації числа її параметрів. Це правило відоме як «принцип економії» і полягає у перевазі простої моделі над складнішою. Параметри моделей типу ARIMA оцінюються з урахуванням коефіцієнтів автокореляції вихідного процесу.

Завдання полягає у визначенні загального виду моделі з класу моделей ARMA(p,q) з найменшою кількістю параметрів порівняно з іншими можливими варіантами, без втрат точності опису вихідного процесу. Цей процес

супроводжується процедурами оцінки параметрів альтернативних варіантів моделей та вибору найкращого з них на основі критеріїв якості [32].

Ідентифікація моделі ARMA(p,q) можлива лише для стаціонарних часових рядів. Вкрай важлива інформація про порядок p моделі авторегресії AR міститься в приватній авторкореляційній функції PACF. Для процесу, що описується моделлю AR(p), її значеннями є останні значення коефіцієнтів моделей авторегресії порядків, що не перевищують p.

Хоча модель ARIMA може обробляти дані з трендом, вона не підтримує тимчасові ряди із сезонним компонентом.

Розширення ARIMA, яке підтримує пряме моделювання сезонного компонента ряду, називається SARIMA.

ARIMA очікує дані, які не є сезонними або сезонний компонент видалений, наприклад, сезонно коригується за допомогою таких методів, як сезонна відмінність.

Сезонне авторегресійне інтегроване середнє ковзне, SARIMA або Seasonal ARIMA, є розширенням ARIMA, яке явно підтримує одновимірні дані часових рядів із сезонним компонентом.

Він додає три нові гіперпараметри для вказівки авторегресії (AR), різниці (I) та ковзного середнього (MA) для сезонної складової ряду, а також додатковий параметр для періоду сезонності.

Налаштування SARIMA потребує вибору гіперпараметрів як для трендових, так і для сезонних елементів ряду.

Є три елементи тренду, які вимагають налаштування.

Вони такі ж як в моделі ARIMA; зокрема:

— n: Порядок авторегресії тренду.

— d: Порядок зміни тренда.

— Q: Тренд ковзної середньої.

Є чотири сезонні елементи, які не є частиною ARIMA, які мають бути налаштовані; це:

— n: Сезонний порядок авторегресії.

— D: Порядок сезонних різниць.

— Q: Сезонний порядок ковзних середніх.

— m: Кількість тимчасових кроків за сезон.

Водночас позначення для моделі SARIMA задається як:

SARIMA(p, d, q) (P, D, Q) m

Де вказані спеціально вибрані гіперпараметри для моделі; наприклад:

SARIMA(3,1,0)(1,1,0)12

Важливо відзначити, що m параметр впливає на n, D, а також Q параметри. Наприклад, m 12 для місячних даних передбачає річний сезонний цикл.

n = 1 використовуватиме перше сезонне зміщення в моделі, наприклад, т-(m*1) або т-12. n=2, використовуватиме останні два сезонні зміщення спостережень t-(m*1), t-(m*2).

Так само, D1 розраховуватиме сезонну різницю першого порядку і Q=1 використовуватиме помилки першого порядку моделі (наприклад, ковзне середнє).

Сезонна модель ARIMA використовує різницю із запізненням, що дорівнює кількості сезонів, для усунення адитивних сезонних ефектів. Як і у випадку з різницею у запізнюванні 1 для видалення тренду, відмінність у запізнюванні вводить термін ковзне середнє. SARIMA включає умови авторегресії та ковзного середнього із затримкою.

Висновки до розділу 2

Авторегресійне інтегроване ковзне середнє або ARIMA – це метод прогнозування для одномірних даних тимчасових рядів.

Як випливає з назви, він підтримує елементи авторегресії та ковзного середнього. Інтегрований елемент відноситься до різниці, що дозволяє методу підтримувати дані часових рядів за допомогою тренду.

Проблема з ARIMA у тому, що вона не підтримує сезонні дані. Це тимчасовий ряд з циклом, що повторюється.

Сезонні моделі ARIMA потенційно можуть мати велику кількість параметрів та комбінацій термінів. Тому доцільно випробувати широкий спектр моделей при доборі даних та обрати найбільш підходящу модель, використовуючи відповідний критерій.

РОЗДІЛ 3
ПРОГРАМНА РЕАЛІЗАЦІЯ СЕМАНТИЧНОГО АНАЛІЗУ ЕКОНОМІЧНИХ НОВИН ДЛЯ ПРОГНОЗУВАННЯ РОЗВИТКУ ЕКОНОМІКИ НА ОСНОВІ ТЕХНОЛОГІЙ ОБРОБКИ ВЕЛИКИХ ДАНИХ

3.1 Постановка задачі

Ціни на нафту мають велике значення для світової економіки. Нинішні ринкові умови є більш нестабільними, ніж зазвичай, частково через геополітичну невизначеність та через те, що жорсткі ринки продуктів, зокрема на бензин, посилюють тиск на підвищення цін на нафту. Вищі ціни сприяють високому рівню безробіття та загострюють проблеми з бюджетним дефіцитом у багатьох країнах.

Завдання розробленого ПЗ – на основі семантичного аналізу новин про ціни на нафту та тенденцій заданої економічної метрики зробити прогноз розвитку економіки.

3.2 Вибір мови програмування

Python — це інтерпретована, об’єктно-орієнтована мова програмування високого рівня. Оскільки вона є універсальною, має широкий спектр застосувань від веб-розробки, створення графічного інтерфейсу робочого столу до наукових і математичних обчислень.

Python популярна завдяки своєму простому та відносно зрозумілому синтаксису. Читабельність його синтаксису підвищує продуктивність, оскільки дозволяє більше зосередитися на проблемі, а не на структуруванні коду [33].

Python популярний з кількох причин. Ось більш глибокий погляд на те, що робить його настільки універсальним і простим у використанні для розробників:

— Python має простий синтаксис, який імітує природну мову, тому його легше читати та розуміти. Це дозволяє швидше створювати проекти та швидше їх покращувати.

— Python універсальний. Python можна використовувати для багатьох різних завдань, від веб-розробки до машинного навчання.

— Python зручний для початківців, що робить його популярним для програмістів початкового рівня.

— Це відкритий вихідний код, що означає, що його можна безкоштовно використовувати та поширювати навіть у комерційних цілях.

— Архів модулів і бібліотек Python — пакети коду, які сторонні користувачі створили для розширення можливостей Python — величезний і зростає.

— Python має велику та активну спільноту, яка вносить свій внесок у пул модулів і бібліотек Python, а також є корисним ресурсом для інших програмістів. Велика спільнота підтримки означає, що якщо розробники стикаються з каменем спотикання, знайти рішення відносно легко; хтось обов'язково стикався з такою ж проблемою раніше.

Scikit-learn (sklearn) – це один з найбільш широко використовуваних пакетів Python для Data Science та Machine Learning. Він містить функції та алгоритми для машинного навчання: класифікації, прогнозування чи розбивки даних на групи. Sklearn написана мовами Python, C, C++ та Cython.

Scikit-learn застосовується:

— у рекомендаційних системах: наприклад, реклама на основі раніше виявлених інтересів або сервіс музики з «розумними» рекомендаціями;

— для розпізнавання тексту та зображень;

— для передбачення поведінки користувачів, фінансових коливань та інших прогнозованих явищ з урахуванням існуючих даних;
— для класифікації даних та автоматичної побудови метрик у бізнес-аналізі;
— при розподілі та обробці результатів досліджень – наукових, медичних та інших.

Scikit-learn заснована на інших бібліотеках, які також застосовуються в машинному навчанні, аналізі даних, комп'ютерному зорі та суміжних сферах.

NumPy – бібліотека для роботи з багатовимірними масивами числових даних та зі складними математичними операціями.

SciPy – набір просунутих математичних та наукових функцій. Заснована на NumPy, але глибша та функціональніша.

Matplotlib – бібліотека для візуалізації та побудови графіків. Підтримує двовимірні та тривимірні побудови.

Pandas – бібліотека для обробки даних, аналізу та маніпуляцій з ними.

SymPy – бібліотека для роботи із символьною математикою.

IPython – інтерактивна консоль для зручнішої роботи з бібліотеками. Серед її можливостей – авто доповнення фраз, підсвічування коду, підтримка візуалізації даних та багато іншого.

Завдання, що вирішує бібліотека Scikit-learn [34]:

— Препроцесинг. Термін «препроцесинг» (preprocessing) означає попередню обробку даних. Вони приводяться до виду, необхідного для подачі на вхід будь-якого алгоритму. Наприклад, зображення підганяються під один розмір та колірну схему. З даних вилучаються ключові ознаки, за якими буде навчатися модель, і також призводять до потрібного формату.
— Зменшення розмірності. Часто у даних міститься надмірна інформація. Наприклад, деякі ознаки можна отримати з інших. Щоб

зробити подальший аналіз ефективнішим, розмірність вибірки зменшують – так, щоб зберегти максимум корисних даних. І тому використовуються спеціальні методи, наприклад метод головних компонент.

— Виявлення аномалій. Алгоритми «відсікають» з набору даних помилкові чи не потрібні записи, які лише додають зайві похибки. Це потрібно, щоб аналіз та навчання працювали точніше.

— Вибір датасету. Вони також є у бібліотеці.

— Вибір моделі. Функції та алгоритми допомагають оцінити ефективність різних моделей для вирішення задачі. Їх можна порівнювати один з одним, проводити валідацію результатів, вибирати точніші. Це потрібно для покращення якості навчання.

— Регресія. Це прогнозування показників за наявними даними, які можуть приймати нескінченну кількість різних значень. Ці показники мають бути пов'язані з будь-яким об'єктом, тобто, бути його атрибутами. Наприклад, прогноз кількості користувачів на сайті у різні дні – це завдання регресії.

— Кластеризація. Це розподіл даних з датасета за великими групами – кластерами, тобто їх угруповання. Схожі об'єкти об'єднуються у класи. Критерії, якими визначається схожість, залежить від моделі та умов завдання.

Моделі машинного навчання:

— Лінійні. Моделі з лінійною залежністю одних значень від інших. Дозволяють будувати площини, що розділяють, або апроксимувати результати і використовувати лінійну регресію.

— Метричні. Передбачають властивості об'єкта з урахуванням його сусідів. Належать до так званого лінивого навчання.

— Дерева рішень. Це деревоподібні моделі, що складаються з наборів правил. Вони приймають рішення з урахуванням вхідних умов.

— Ансамблеві. Методи, які використовують велику кількість дерев рішень.

— Нейромережі. Моделі, структура яких ґрунтується на біологічних нейронних мережах, коли нейрони з'єднані між собою синапсами. Нейрон – обчислювальна одиниця, синапс – шлях передачі інформації від одного нейрона до іншого. Така модель самонавчається з урахуванням попереднього досвіду і з кожним разом робить все менше помилок.

— Крос-валідація. Метод багаторазового навчання, для якого використовується датасет цілком. Для кожного кроку валідація походить з однієї з частин датасету. Цей метод розбиває дані на кілька секцій і навчає кілька алгоритмів в цих секціях.

У бібліотеку входять такі популярні методи, як PCA (метод головних компонент, один з основних способів зменшити розмірність даних з мінімальними втратами інформації), SVM (метод опорних векторів, набір схожих алгоритмів навчання, які використовуються для класифікації задач і регресійного аналізу) та інші.

Переваги Scikit-learn [34]:

— Багатофункціональність. Scikit-learn має багато можливостей, починаючи з підготовки даних і закінчуючи візуалізацією результату. Вона вже пов'язана з багатьма технологіями, якими користуються у сфері машинного навчання, тому бібліотеки достатньо для більшості завдань, які постають перед фахівцем з Data Science. Це майже універсальне рішення машинного навчання.

— Багато алгоритмів. У Scikit-learn входить великий набір методів та алгоритмів, які потрібні для прогнозування, кластеризації, розпізнавання тощо. Їх не потрібно писати з нуля – для кожного є своя функція. Достатньо подати на вхід необхідні дані. Для типових завдань є готові рішення, аж до навчальних датасетів та підготовлених моделей.

— Зручність використання. Зв'язок з іншими бібліотеками, зрозумілі назви функцій та простий синтаксис роблять роботу зручною та простою.

— Безкоштовний доступ. Scikit-learn безкоштовна та має відкритий вихідний код. Це означає, що будь-який розробник може його подивитися. Sklearn розповсюджується за вільною ліцензією BSD: бібліотеку можна використовувати і в комерційних проектах, і безкоштовному ПЗ. Репозиторій на сервісі GitHub відкритий всім бажаючим.

— Крос-платформність. Бібліотека Scikit-learn підтримується операційними системами Linux, Windows та MacOS, як і сам Python.

— Широке ком'юніті. Фахівці з Data Science активно пишуть туторіали, гайди та допоміжні матеріали. Отримати інформацію легко. А якщо є питання — можна поставити їх на спеціалізованому ресурсі та отримати відповіді від ком'юніті.

— Популярність. Sklearn використовують у ряді великих проектів: міжнародні сервіси Spotify та Booking, платформа пошуку вакансій HeadHunter та ін.

Недоліки Scikit-learn [34]:

— Відсутність низки можливостей. У Sklearn мало інструментів для побудови нейронних мереж. Тому для деяких завдань однієї Scikit-learn буде недостатньо, незважаючи на її широкість.

— Велика кількість залежностей. Глибокий зв'язок з іншими бібліотеками – одночасно і плюс, і мінус. Звичайно, зручніше вибудовувати роботу з кількома інструментами одночасно, такий підхід розширює функціональність. Але це означає, що для роботи з Scikit-learn потрібно розбиратися в її залежностях і бажано вміти користуватися всіма пов'язаними інструментами. Це збільшує поріг входу для новачків.

— Складність у вивченні. Data Science – складна галузь. Потрібно добре розбиратися не тільки у програмуванні, а й у математиці, алгоритмах прогнозування і кластеризації. Потрібна глибока теоретична база. Незважаючи на те, що самі моделі не потрібно будувати з нуля, розробник повинен розуміти, як вони працюють.

ARIMA (Auto Regressive Integrated Moving Average) є класом моделей, які описують поведінку даних на основі історичних значеннях ряду. Integrated – процес що призводить тимчасовий ряд до стаціонарності.

Покрокова реалізація ARIMA:

1. Завантаження даних. Перший крок для побудови моделі – це завантаження набору даних.

2. Попередня обробка.

3. Перетворення ряду на стаціонарний.

4. Визначення значення d: для того, щоб зробити ряд стаціонарним, кількість виконання операцій взяття різниці прийматиметься як значення d.

5. Створення графіків ACF та PACF: це найважливіший крок у реалізації ARIMA. Графіки ACF та PACF використовуються для визначення вхідних параметрів моделі ARIMA.

6. Визначення значень p та q (з графіків з попереднього кроку).

7. Підгонка моделі ARIMA. Використовуючи оброблені дані та значення параметрів, які були розраховані з попередніх кроків.

8. Прогнозування значень на перевірочному наборі. Прогнозування майбутніх значень.

9. Розрахунок середньоквадратичної помилки. Для перевірки ефективності моделі перевіряються значення середньоквадратичної помилки моделі, використовуючи прогнозні та фактичні значення у перевірочному наборі даних.

Метод прогнозування часових рядів SARIMA, так як і ARIMA підтримується в Python через бібліотеку Statsmodels [35].

Щоб використовувати SARIMA, потрібно виконати три кроки:

— Визначити модель.

— Відповідність моделі.

— Зробити прогноз за допомогою відповідної моделі.

1. Визначення моделі.

Примірник класу SARIMAX може бути створений шляхом надання навчальних даних та безлічі параметрів конфігурації моделі.

Реалізація називається SARIMAX замість SARIMA, тому що додавання X до імені методу означає, що реалізація також підтримує екзогенні змінні.

Це паралельні часові ряди, які не моделюються безпосередньо за допомогою процесів AR, I або MA, але стають доступними як виважене введення для моделі.

Трендовий і сезонний гіперпараметри вказуються як кортежі з 3 і 4 елементів відповідно до порядку, а також seasonal_order Аргументи.

2. Відповідність моделі.

Після того, як модель створена, її поміщають на тренувальні дані.

Підгонка моделі повертає екземпляр SARIMAX Results учбовий клас. Багато елементів процесу підгонки можуть бути налаштовані.

3. Прогнозування даних.

Після підгонки модель можна використовувати для складання прогнозу.

Функція прогнозу приймає один параметр, який визначає кількість тимчасових кроків з вибірки для прогнозу, або передбачає однокроковий прогноз, якщо аргументи не надані.

Крім того, якщо при визначенні моделі були надані зовнішні змінні, вони також повинні бути надані на період прогнозування.

Для зберігання даних використається бібліотека Pandas.

Pandas – це opensource-бібліотека, тобто її вихідний код у відкритому доступі розміщено на GitHub. Користувачі можуть додавати туди свій код: вносити пояснення, доповнювати методи роботи та оновлювати розділи. Для роботи знадобиться компілятор (програма, яка перекладає текст з мови програмування в машинний код) C/C++ та середовище розробки Python.

Цей пакет Python розроблений на основі бібліотеки NumPy. Такий вибір зумовлює успіх та швидке поширення pandas. Він також користується всіма перевагами NumPy та робить pandas сумісною з більшістю інших модулів.

Ще одне важливе рішення – розробка спеціальних структур для аналізу даних. Замість того, щоб використовувати вбудовані в Python або інші бібліотеки структури, були розроблені дві нових.

Вони спроектовані для роботи з реляційними та класифікованими даними, що дозволяє управляти даними способом, схожим на той, що використовується у реляційних базах SQL та таблицях Excel.

Бібліотека Pandas Python містить у собі 3 основні структури даних:

— Series – це невеликий об'єкт, схожий на одновимірний масив, який не змінює розміри;

— DataFrame – це об'єкт, що володіє табличною структурою даних, який може змінювати розміри;

— Panel – це об'єкт, який має структуру тривимірного масиву, здатного змінювати розміри.

DataFrame у Pandas відіграє дуже важливу роль, так як дана структура даних є основним та стандартним способом зберігання інформації. Вона містить рядки та стовпці.

Щоб створити об'єкт DataFrame Pandas, можна застосувати наступний конструктор: pandas.DataFrame(data, index, columns, dtype, copy).

Коротке пояснення конструктора:

— data – це створення об'єкта DataFrame із вхідних даних, у ролі таких даних можуть виступати: списки, інші об'єкти DataFrame, Series, масиви NumPy та ін;

— index – представляє рядкові мітки;

— columns – застосовується для конструювання підписів стовпців;

— dtype – необов'язковий параметр, вказує на типи даних стовпців;

— copy – застосовується для копіювання даних, коли потрібно.

У Pandas працюють з форматами CSV, Excel, SQL, HTML, HDF та іншими. Повний список можна переглянути за допомогою методу .read, де через нижнє підкреслення «_» будуть вказані всі доступні формати.

Matplotlib складається з безлічі модулів. Модулі наповнені різними класами та функціями, які ієрархічно пов'язані між собою.

Термінологія:

— Figure є закінченим вікном або сторінкою, на якій побудований графік.

— Axes – це область, де відображаються дані. Це може бути вісь X, вісь Y і т.д.

— Spines – це лінії, які сполучають точки Axes.

Matplotlib має об'єктно-орієнтований інтерфейс, тобто користувач безпосередньо взаємодіє з кожним об'єктом. За допомогою коду можна задавати будь-який елемент діаграми, у тому числі ярлики та позначки на осях.

Архітектура Matplotlib логічно розділена на три шари, розташовані на трьох рівнях. Комунікація непряма – кожен шар може взаємодіяти тільки з тим, який розташований під ним, але не над.

Архітектура Matplotlib:

— Шар сценарію.

— Художній шар.

— Шар Backend.

1) Шар Backend.

Шар Backend нижній на діаграмі з архітектурою всієї бібліотеки. Він містить всі API та набір класів, що відповідають за реалізацію графічних елементів на низькому рівні.

— FigureCanvas – це об'єкт, що уособлює область малювання.

— Renderer – об'єкт, який малює FigureCanvas.

— Event – об'єкт, що обробляє введення від користувача (події з клавіатури та миші)

2) Художній шар.

Середнім шаром є художній (artist). Усі елементи, що становлять графік, такі як назва, мітки осей, маркери тощо, є екземплярами цього об'єкта. Кожен з них грає свою роль в ієрархічній структурі.

Є два художні класи: примітивний та складовий.

— Примітивний – це об'єкти, які є базовими елементами для формування графічного представлення графіка, наприклад, Line2D, або геометричні фігури, такі як прямокутник коло або навіть текст.

— Складові – об'єкти, що складаються з кількох базових (примітивних). Це осі, шкали та діаграми.

На цьому рівні часто доводиться мати справу з об'єктами, що займають високе положення в ієрархії: графік, система координат, осі. Тому важливо

повністю розуміти, яку роль вони відіграють. Наступне зображення показує три основні художні (складові об'єкти), які часто використовуються на цьому рівні.

Figure – об'єкт, що займає верхню позицію в ієрархії. Він відповідає всьому графічному представленню і може містити багато систем координат.

Axes – це той самий графік. Кожна система координат належить лише одному об'єкту Figure і має два об'єкти Axis (або три, якщо йдеться про тривимірний графік). Інші об'єкти, такі як назва, мітки x та y, належать окремо до осей.

Axis враховує числові значення в системі координат, визначає межі та керує позначеннями на осях, а також відповідним них текстом. Положення шкал визначається об'єктом Locator, а зовнішній вигляд – Formatter.

3) Шар сценарію (pyplot).

Художні класи та пов'язані з ними функції (API matplotlib) підходять усім розробникам, особливо тим, хто працює із серверами веб-додатків або розробляє графічні інтерфейси. Але для обчислень, зокрема для аналізу та візуалізації даних, найкраще підходить шар сценарію. Він включає інтерфейс pyplot.

Seaborn – бібліотека для створення статистичних графіків на Python [38]. Вона побудована на основі Matplotlib і тісно інтегрується зі структурами даних Pandas. Seaborn допомагає вивчити та зрозуміти дані. Її функції побудови графіків працюють із датасетами і виконують всі необхідні перетворення для створення інформативних графіків.

Seaborn пропонує безліч функцій, які роблять його корисним та простим у порівнянні з іншими фреймворками. Деякі з цих функцій:

— Функція для побудови статистичних даних тимчасових рядів із гнучкою оцінкою та представленням невизначеності оцінки.

— Функції для візуалізації одновимірних та двовимірних розподілів або для порівняння їх між підмножиною даних.

— Функції, які візуалізують матриці даних та використовують алгоритми кластеризації для виявлення структури у цих матрицях.

— Абстракції високого рівня структурування сіток графіків, які дозволяють легко створювати складні візуалізації.
— Декілька вбудованих тем для стилізації графіки Matplotlib.
— Інструменти для вибору палітри кольорів для створення красивих графіків, що розкривають закономірності у даних.
— Інструменти, які підходять та візуалізують моделі лінійної регресії для різних типів незалежних та залежних змінних.

Візуалізація відіграє важливу роль, під час дослідження та розуміння даних, Seaborn націлений на те, щоб спростити це завдання та стати центром процесу. Для порівняння: якщо ми говоримо, що matplotlib спрощує і робить складні речі можливими, то seaborn намагається спростити і це складне завдання, причому чітко певним чином. Але seaborn не є альтернативою matplotlib, вважаючи його доповненням до попереднього.

3.3 Результати розробки програмного забезпечення

Вхідними даними в проект є два датасети: набір формату дата-новина англійською мовою та набір формату дата-ціна.

Часові проміжки повинні перетинатися, але не обов'язково ідеально збігатися.

Після запуску програма автоматично завантажує датасет, якщо він знаходиться у хмарі або завантажує з диска, якщо він був кешований. Програма автоматично видаляє ряди з набору даних, які містять неправильні дані або порожні комірки.

Далі ПЗ проводить попередню обробку даних, видаляє з тексту новин «стоп-слова» (слова, які не несуть емоційного навантаження, наприклад: also, a, the), далі видаляє пунктуацію та зайві спец-символи.

Потім проводиться сентиментальний аналіз тексту і кожній новині надається оцінка: «-1» – негативна або «1» – позитивна. З набору цін видаляються порожні рядки.

Після того як проведена попередня обробка даних, набори даних стикуються за датами, дати які не перетинаються видаляються.

Далі, підсумковий набір даних розділяється на 2 частини, тренувальну та тестову. Тренувальна частина використовується для навчання моделей (рис. 3.1.), тестова для визначення точності передбачень. Після тренування моделей із тестового датасету беруться дати та оцінки новин, модель семантичного аналізу на їх основі передбачує ціни. Модель семантичного аналізу в розробленому ПЗ – це логістична регресія (бібліотека sklearn).

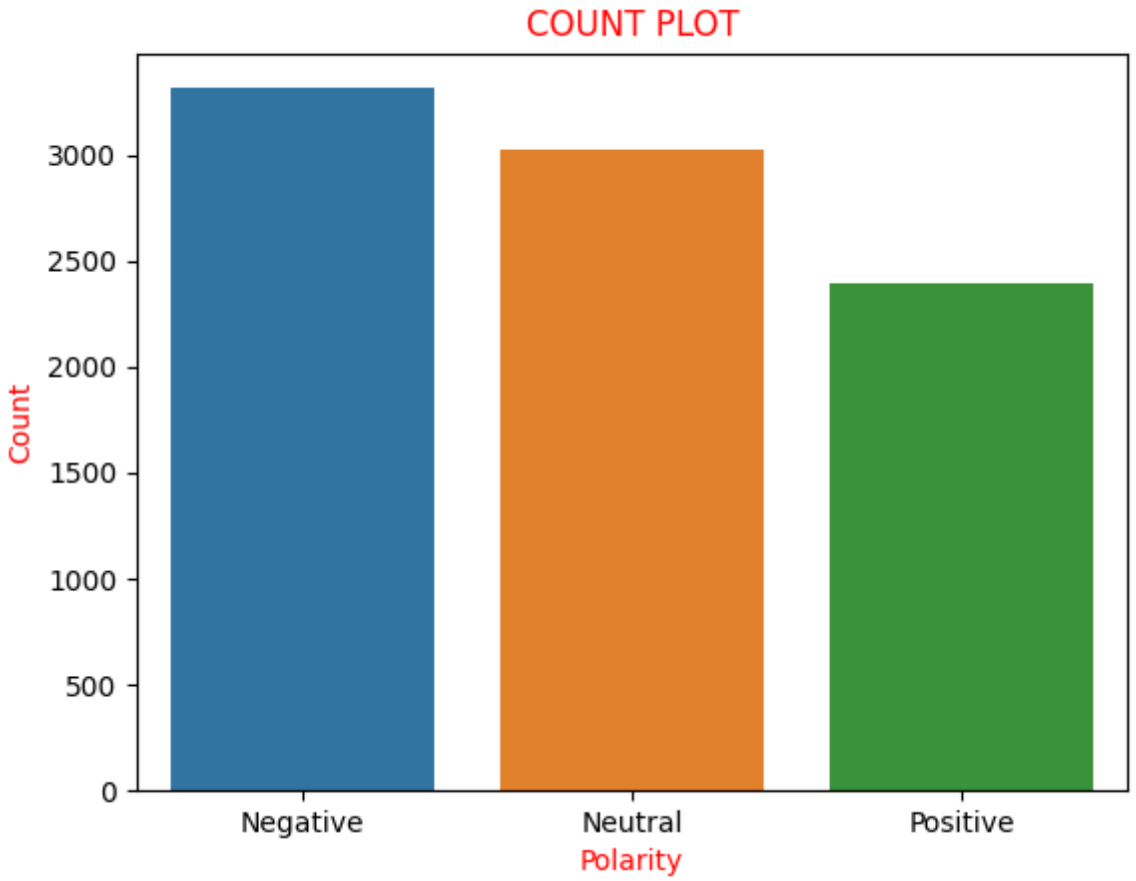

Рис.3.1. Початковий розподіл семантичних оцінок у датасеті новин для навчання моделі

Для передбачення використовується моделі передбачення часових рядів з екзогенними змінними (семантичні оцінки новин) ARIMA та SARIMAX (рис. 3.2.) (бібліотека statsmodels), їх результати порівнюються.

```
RUNNING THE L-BFGS-B CODE

           * * *

Machine precision = 2.220D-16
 N =           18     M =           10

At X0         0 variables are exactly at the bounds

At iterate    0    f=  3.58405D+00    |proj g|=  3.16297D-01

At iterate    5    f=  3.39161D+00    |proj g|=  5.84143D-02

At iterate   10    f=  3.32341D+00    |proj g|=  4.99326D-02

At iterate   15    f=  3.24391D+00    |proj g|=  3.59862D-02

At iterate   20    f=  3.22884D+00    |proj g|=  2.79401D-02

At iterate   25    f=  3.19693D+00    |proj g|=  2.02936D-02

At iterate   30    f=  3.18654D+00    |proj g|=  1.65665D-02

At iterate   35    f=  3.18558D+00    |proj g|=  2.47565D-03

At iterate   40    f=  3.18436D+00    |proj g|=  6.30797D-03

At iterate   45    f=  3.18136D+00    |proj g|=  2.65363D-02
```

Рис.3.2. Процес навчання та автокалібрування моделі SARIMAX

Для зберігання даних використовується бібліотека pandas.

Графіки малюються за допомогою matplotlib і seaborn.

Для навчання моделі прибираються нейтральні новини та робиться розподіл 50\50 (рис.3.3.).

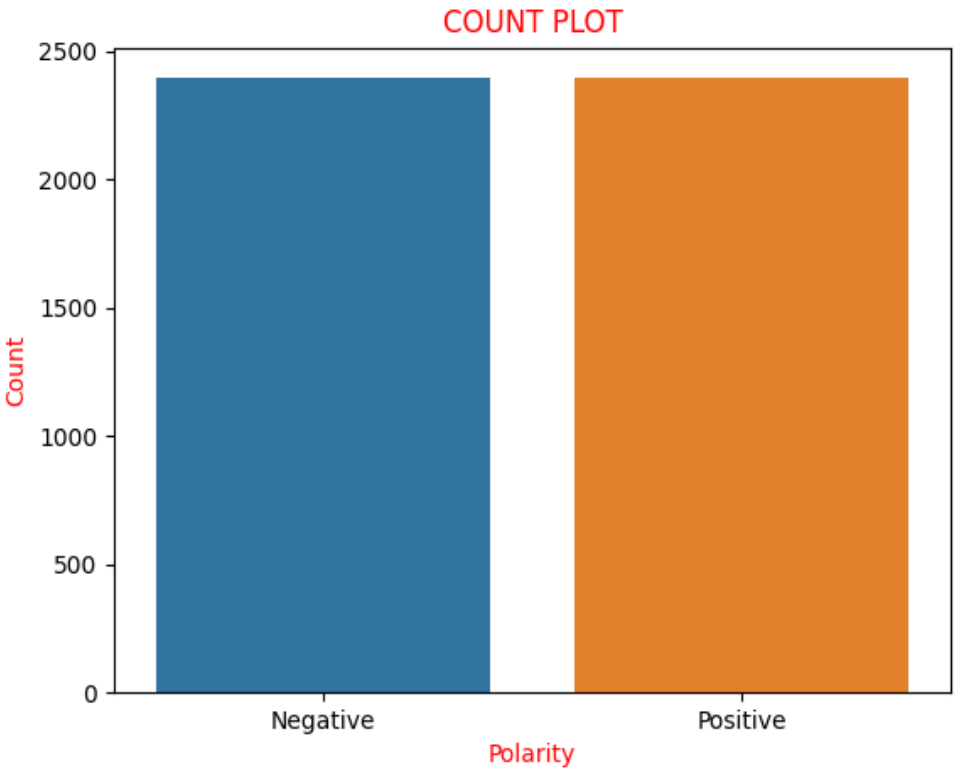

Рис.3.3. Розподіл після видалення нейтральних новин

Користувач отримує графіки результату прогнозування (рис.3.4. – 3.6).

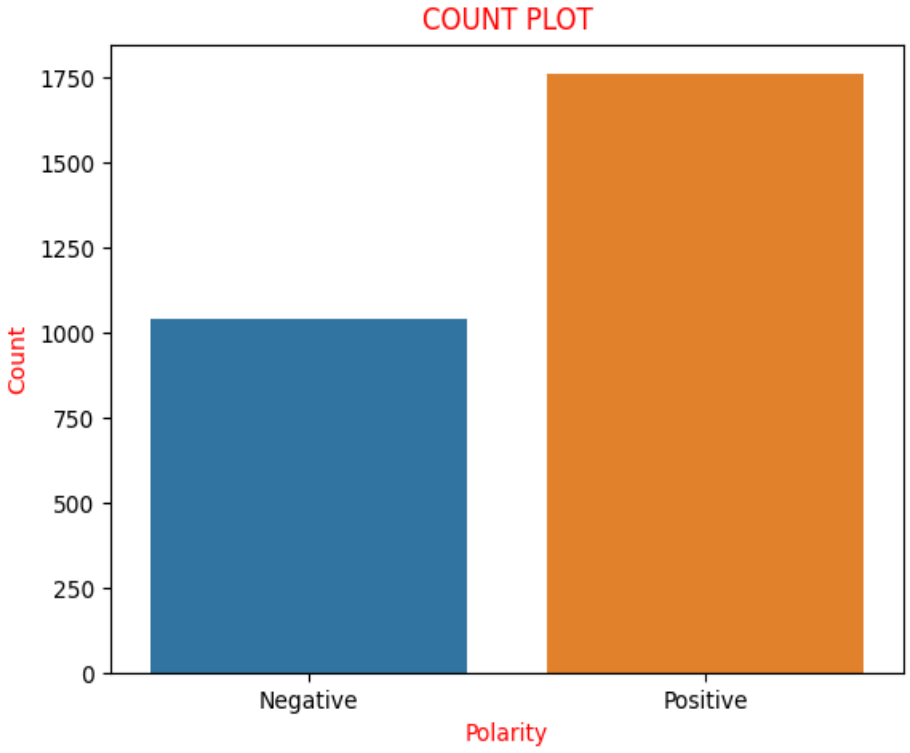

Рис. 3.4. Розподіл полярності новин у завантаженому датасеті новин щодо оцінки моделі дослідження

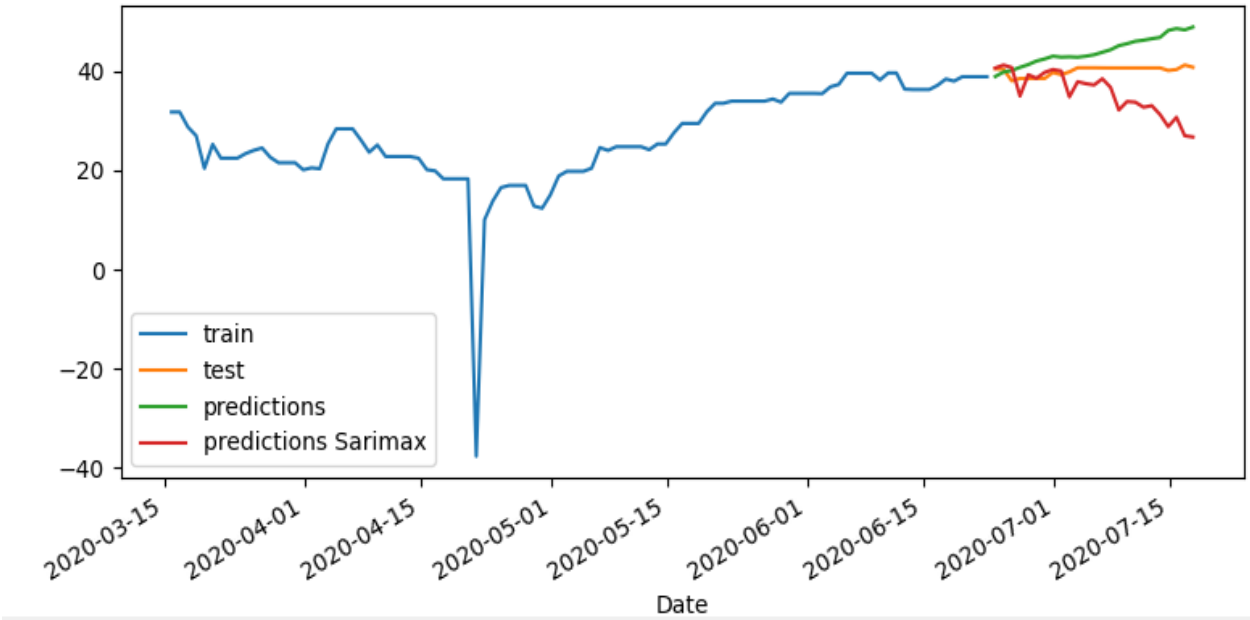

Рис. 3.5. Результат роботи

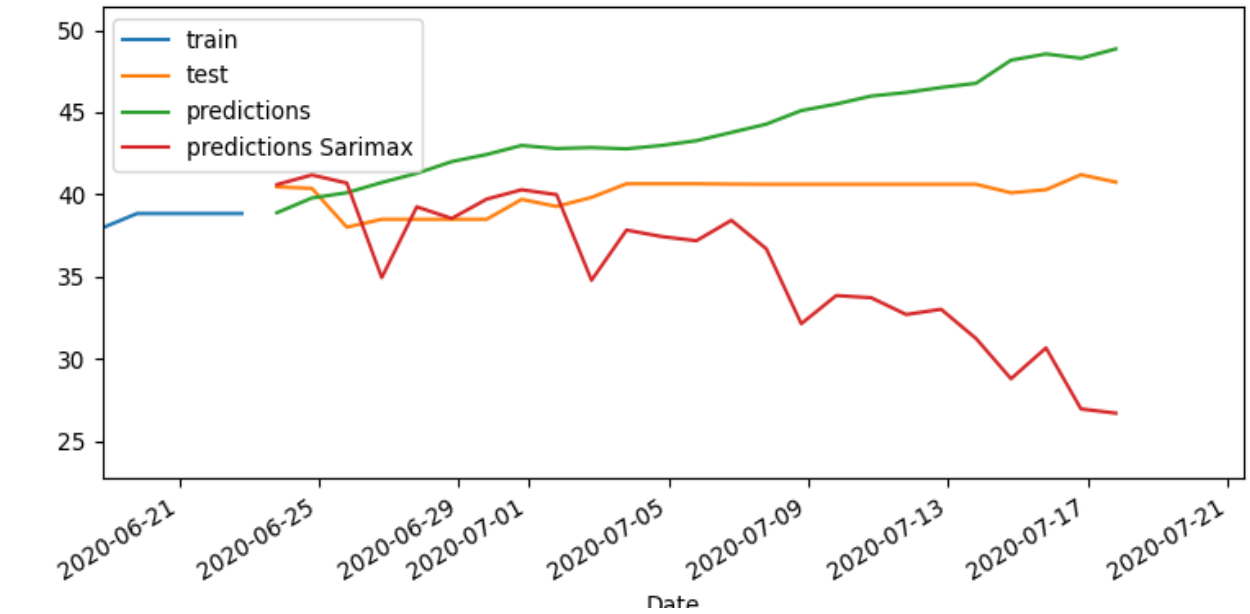

Рис. 3.6. Результат роботи в наближеному вигляді

Як можна побачити на рис. 3.6 – модель ARIMA (зелена лінія) хоч видала і більш схожий за волатильністю графік не співпала з напрямком руху графіка

фактичних змін (жовта лінія). Модель SARIMAX (червона лінія) хоча і більш волатильна, змогла протягом тижня досить точно передбачити зміну цін на нафту.

Висновки до розділу 3

Багато властивостей мови програмування Python роблять її дуже привабливим варіантом при виборі мови програмування для проекту NLP, а саме для обробки природної мови. До таких властивостей Python можна віднести насамперед простий синтаксис і прозору семантику мови. Крім того, є можливість користуватися підтримкою інтеграції з іншими мовами та інструментами – це стане у нагоді для таких методів, як машинне навчання.

В результаті аналізу економічних новин розроблене програмне забезпечення демонструє майбутні зміни цін на нафту за допомогою моделей ARIMA та SARIMAX.

ВИСНОВКИ

В результаті дослідження було розроблено програмне забезпечення семантичного аналізу економічних новин для прогнозування розвитку економіки на основі технологій обробки великих даних.

Виконані наступні завдання дослідження:

- Проведено аналіз існуючих рішень та огляд предметної області.
- Описана методологія семантичного аналізу тексту, передбачення часових рядів та моделі прогнозування ARIMA та SARIMA.
- Розроблено програмне забезпечення семантичного аналізу економічних новин для прогнозування розвитку економіки на основі технологій обробки великих даних.

Для розробки ПЗ була використана мова програмування Python. Були використані моделі передбачення часових рядів з екзогенними змінними ARIMA та SARIMAX.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Jurafsky, D., & Martin, J. H. (2009). Speech and language processing: An introduction to natural language processing, speech recognition, and computational linguistics (2nd ed.). New Jersey: Prentice-Hall.

2. Behrang Mohit, Nathan Schneider, Rishav Bhowmick, Kemal Oflazer, and Noah A Smith. 2012. Recalloriented learning of named entities in Arabic Wikipedia. In Proceedings of the 13th Conference of the European Chapter of the Association for Computational Linguistics. Association for Computational Linguistics.

3. Noam Chomsky. Three models for the description of language. In IRE Transactions on Information Theory, volume IT2, pages 113–124. New York: Wiley, 1956.

4. Noam Chomsky. Syntactic Structures. The Hague: Mouton, 1957.

5. Murdoch, W. J., Singh, C., Kumbier, K., Abbasi-Asl, R., & Yu, B. “Definitions, methods, and applications in interpretable machine learning.” Proceedings of the National Academy of Sciences, 116(44), 22071-22080, 2019.

6. T. Mikolov, A. Deoras, D. Povey, L. Burget, J. ˇCernock´y. Strategies for Training Large ScaleNeural Network Language Models, In: Proc. Automatic Speech Recognition and Understand-ing, 2011.

7. Goldberg, Yoav; Levy, Omer. "word2vec Explained: Deriving Mikolov et al.'s Negative-Sampling Word-Embedding Method", 2014.

8. Nguyen, Anh, Alexey Dosovitskiy, Jason Yosinski, Thomas Brox, and Jeff Clune. “Synthesizing the preferred inputs for neurons in neural networks via deep generator networks.” Advances in neural information processing systems 29, 2016.

9. Hochreiter, S.; Schmidhuber, J. Long short-term memory. Neural Comput. 1997, 9, 1735–1780.

10. Batbaatar, E.; Park, H.W.; Li, D.; Li, M.; Ryu, K.H. DeepEnergy: Prediction of appliances energy with long-short term memory recurrent neural network. In Asian Conference on Intelligent Information and Database Systems; Springer: Cham, Swithzerland, 2018; pp. 224–234.

11. Ghaderi, A.; Sanandaji, B.M.; Ghaderi, F. Deep Forecast: Deep Learning-Based Spatio-Temporal Forecasting; Cornell University: Ithaca, NY, USA, 2017.

12. Yugesh Verma. A guide to document embeddings using Distributed Bag-of-Words (DBOW) model, 2022. URL: https://analyticsindiamag.com/a-guide-to-document-embeddings-using-distributed-bag-of-words-dbow-model/.

13. Nguyen A. Рекурентна нейронна мережа для обробки великих текстових даних / A. Nguyen, Y. Sidorov // Системи управління, навігації та зв'язку. Збірник наукових праць. – Полтава: ПНТУ, 2018. – Т. 4 (50). – С. 135-138.

14. Christopher Manning. Computational linguistics and deep learning. Computational Linguistics. — 2016.

15. Зеленський А. А. Актуальність дослідження програм для семантичного аналізу текстів та огляду методів їх реалізації / Зеленський Аркадій А. // Computer Science & Software Engineering : Proceedings of the 1st Student Workshop (CS&SE@SW 2018), Кривий Ріг, Україна, 30 листопада 2018 р. – С. 63-69.

16. Комарницька О. І. Моделі та методи лінгвістичного аналізу тексту інтелектуальної системи оцінювання знань : автореф. дис. канд. філол. наук : 10.02.21 / Оксана Іванівна Комарницька, НАН України. Нац. б-ка України ім. В. І. Вернадського.– К. : [б.в.], 2015.– 20 с.

17. Clifford A. Lynch, "Big data: How do your data grow?" Nature, vol. 455, no. 7209.

18. Laney, D. (2001) 3D Data Management: Controlling Data Volume, Velocity and Variety. META Group Research Note, 6.

19. White, T. (2015). Hadoop – The definitive guide: Storage and analysis at Internet scale, 4th edn. Sebastopol, CA: O'Reilly Media.

20. Верес О. М. Класифікація методів аналізу великих даних / О. М. Верес, Р. М. Оливко // Вісник Національного університету "Львівська політехніка" – 2017. – Випуск 872. – С.84-92.

21. Critical Mention. URL: https://www.criticalmention.com/

22. Sentiment Analyzer. URL: https://monkeylearn.com/sentiment-analysis-online/

23. SAS Visual Analytics. URL: https://www.sas.com/en_us/software/visual-analytics.html

24. Alteryx. URL: https://www.alteryx.com/

25. Google Analytics. URL: https://analytics.google.com/

26. IBM Cognos Analytics. URL: https://www.ibm.com/products/cognos-analytics

27. Haddi, E., Liu, X., & Shi, Y. (2013). The role of text pre-processing in sentiment analysis. Procedia computer science, 17, 26-32.

28. S. Russell, P. Norving, Artificial Intelligence: A Modern Approach, second edidtion Edition, Prentice Hall Artificial Intelligence Series, Pearson Education Inc., 2003.

29. V. Vapnik, The nature of statistical learning theory, springer, 1999.

30. Берзлев, О. Ю. Сучасний стан інформаційних систем прогнозування часових рядів [Текст] / О. Ю. Берзлев // Управління розвитком складних систем. – 2013. – Вип. 13. – С. 78–82.

31. Box G.E., Jenkins G.M., Reinsel G.C. Time series analystrol: Forecasting and Control. John. Wiley & Sons, Hoboken, 2015. P. 456.

32. Sandhu, K.S.; Nair, A.R. A Comparative Study of ARIMA and RNN for Short Term Wind Speed Forecasting. In Proceedings of the 2019 10th International Conference on Computing, Communication and Networking Technologies (ICCCNT), Kanpur, India, 6–8 July 2019; IEEE: New York, NY, USA, 2019; pp. 1–7.

33. Python. URL: https://www.python.org/

34. Scikit-learn. URL: https://scikit-learn.org/

35. Statsmodels. URL: https://www.statsmodels.org/

36. Pandas. URL: https://pandas.pydata.org/

37. Matplotlib. URL: https://matplotlib.org/

38. Seaborn. URL: https://seaborn.pydata.org/

ДОДАТОК А

```
from datasets import load_dataset

import pandas as pd                # data processing, CSV file I/O (e.g. pd.read_csv)
import matplotlib.pyplot as plt    #For Visualisation
import seaborn as sns              #For better Visualisation
from bs4 import BeautifulSoup      #For Text Parsing
import nltk
nltk.download('stopwords')
nltk.download('punkt')

datasetDict = load_dataset("fhamborg/news_sentiment_newsmtsc")
dataset = datasetDict['train']

data = dataset.to_pandas()
del data['mention']
del data['from']
del data['to']
del data['id']
print(data.columns)
print(data.isnull().sum())

data = data.dropna()

sns.countplot(data['polarity'])
plt.xlabel('Polarity', color = 'red')
plt.ylabel('Count', color = 'red')
plt.xticks([0,1,2],['Negative','Neutral','Positive'])
plt.title('COUNT PLOT', color = 'r')
plt.show()
```

```
amountOfSamples = 5000

amountOfSamples = min(amountOfSamples, data['polarity'].value_counts()[-1], data['polarity'].value_counts()[1])
print(amountOfSamples)

select =[]
p = 0
n = 0

i=0

for row in data.iterrows():
    if (row[1]['polarity'] == 1) and (p < amountOfSamples):
        p = p + 1
        select.append(row[1])
    elif row[1]['polarity'] == -1 and n < amountOfSamples:
        n = n + 1
        select.append(row[1])

print(len(select))
select = pd.DataFrame.from_records(select)
print(select['polarity'].value_counts())
sns.countplot(select['polarity'])
plt.xlabel('Polarity', color = 'red')
plt.ylabel('Count', color = 'red')
plt.xticks([0,1],['Negative','Positive'])
plt.title('COUNT PLOT', color = 'r')
plt.show()
```

```
def strip_html(text):
    soup = BeautifulSoup(text, "html.parser")
    return soup.get_text()
select['text'] = select['sentence'].apply(strip_html)
select=select.drop('sentence',axis=1)
select.head()
print(select.columns)

def punc_clean(text):
    import string as st
    a=[w for w in text if w not in st.punctuation]
    return ''.join(a)

select['text'] = select['text'].apply(punc_clean)
select.head(2)

def remove_stopword(text):
    stopword=nltk.corpus.stopwords.words('english')
    stopword.remove('not')
    a=[w for w in nltk.word_tokenize(text) if w not in stopword]
    return ' '.join(a)
select['text'] = select['text'].apply(remove_stopword)

from sklearn.feature_extraction.text import TfidfVectorizer

vectr = TfidfVectorizer(ngram_range=(1,2),min_df=1)
vectr.fit(select['text'])
vect_X = vectr.transform(select['text'])
```

```
from sklearn.linear_model import LogisticRegression
model = LogisticRegression()
clf=model.fit(vect_X,select['polarity'])
print(clf.score(vect_X,select['polarity'])*100)

print(clf.predict(vectr.transform(["Piece of shit"])))
print(clf.predict(vectr.transform(["Great honor"])))
print(clf.predict(vectr.transform(["plain text"])))

news = pd.read_csv("cnbc_headlines.csv")
print(len(news))
news=news.dropna()
print(len(news))
print(news.columns)

sentiment =[]

for row in news.iterrows():
    sentiment.append(clf.predict(vectr.transform([row[1]['Headlines']]))[0])
print(len(sentiment))

def rankSentiment(text):
    return clf.predict(vectr.transform([text]))[0]*10

news['sentiment'] = news['Headlines'].apply(rankSentiment)

print(news['sentiment'][0])
print(news['Time'][0])

sns.countplot(news['sentiment'])
```

```
plt.xlabel('Polarity', color = 'red')
plt.ylabel('Count', color = 'red')
plt.xticks([0,1],['Negative','Positive'])
plt.title('COUNT PLOT', color = 'r')
plt.show()

print(type(news['Time'][0]))

news['Date'] = pd.to_datetime(news['Time'],infer_datetime_format=True)

print(type(news['Date'][0]))
print(news.groupby(['Date']).value_counts())
print(news.groupby(pd.DatetimeIndex(news['Date']).normalize()).value_counts())
news = news.groupby(pd.DatetimeIndex(news['Date']).normalize()).nth(0)
print(len(news))
sns.countplot(news['sentiment'])
plt.xlabel('Polarity', color = 'red')
plt.ylabel('Count', color = 'red')
plt.xticks([0,1],['Negative','Positive'])
plt.title('COUNT PLOT', color = 'r')
plt.show()

print(news.iloc[[0,-1]])

oilPrice = pd.read_csv('Oil (WTI)_07_17_20-12_22_17.csv')
print(len(oilPrice))
print(oilPrice.columns)
del oilPrice['Open']
del oilPrice['High']
del oilPrice['Low']
```

```
del oilPrice['Volume']
oilPrice['Date']=pd.to_datetime(oilPrice['Date'],infer_datetime_format=True)

#oilPricesel = oilPrice[oilPrice['Date'].isin(news['Time'].index)]

print(news.index)
oilPrice = oilPrice.set_index('Date')
print(oilPrice.index)
merge = news.merge(oilPrice,left_index=True, right_index=True)

print(len(merge))
merge = merge.dropna()
print(len(merge))
print(merge.columns)

#del merge['Date']

import matplotlib.pyplot as plt
print('-------------------------------------------')
print(merge['Date'])
merge = merge.set_index('Date')
merge = merge.asfreq('D','pad')
print(merge.isnull().sum())
print(merge)
steps = 25
train = merge.iloc[:-steps, :]
test  = merge.iloc[-steps:, :]
train = train.iloc[-100:,:]
len(train)
len(test)
```

```
print(train.isnull().sum())
print(train.isnull().sum())

from statsmodels.tsa.arima.model import ARIMA
from statsmodels.tsa.statespace.sarimax import SARIMAX


ARIMAmodel = ARIMA(train['Close'], order = (25, 3, 2),exog=train['sentiment'])
ARIMAmodel = ARIMAmodel.fit()

test['Predictions'] = ARIMAmodel.predict(start = test.index[0], end = test.index[-1],exog=test['sentiment'])

SARIMAXmodel = SARIMAX(train['Close'], order = (5, 3, 4),exog=train['sentiment'], seasonal_order=(6,1,1,6))
SARIMAXmodel = SARIMAXmodel.fit()

test['Sarimax'] = SARIMAXmodel.predict(start = test.index[0], end = test.index[-1],exog=test['sentiment'])

fig, ax=plt.subplots(figsize=(9, 4))
train['Close'].plot(ax=ax, label='train')
test['Close'].plot(ax=ax, label='test')
test['Predictions'].plot(ax=ax, label='predictions')
test['Sarimax'].plot(ax=ax, label='predictions Sarimax')

plt.legend()
plt.show()
```

ДОДАТОК Б SOFTWARE ARCHITECTURE DOCUMENT

Kyiv national university name of

Taras Schevchenko

Master’s course work
Software Architecture Document (SAD)

CONTENT OWNER: Kravtsov Andrii

DOCUMENT NUMBER:	RELEASE/REVISION:	RELEASE/REVISION DATE:
1	1.0	01.06.2021

Table of Contents

List of Figures

List of Tables

1.Documentation Roadmap

The Documentation Roadmap should be the first place a new reader of the SAD begins. But for new and returning readers, it is intended to describe how the SAD is organized so that a reader with specific interests who does not wish to read the SAD cover-to-cover can find desired information quickly and directly.

Sub-sections of Section 1 include the following.

- Section 1.1 (“Document Management and Configuration Control Information”) explains revision history. This tells you if you’re looking at the correct version of the SAD.
- Section 1.2 (“Purpose and Scope of the SAD”) explains the purpose and scope of the SAD, and indicates what information is and is not included. This tells you if the information you’re seeking is likely to be in this document.
- Section 1.3 (“How the SAD Is Organized”) explains the information that is found in each section of the SAD. This tells you what section(s) in this SAD are most likely to contain the information you seek.
- Section 1.4 (“Stakeholder Representation”) explains the stakeholders for which the SAD has been particularly aimed. This tells you how you might use the SAD to do your job.
- Section 1.5 (“Viewpoint Definitions”) explains the *viewpoints* (as defined by IEEE Standard 1471-2000) used in this SAD. For each viewpoint defined in Section 1.5, there is a corresponding view defined in Section 3 (“Views”). This tells you how the architectural information has been partitioned, and what views are most likely to contain the information you seek.
- Section 1.6 (“How a View is Documented”) explains the standard organization used to document architectural views in this SAD. This tells you what section within a view you should read in order to find the information you seek.

1.1. Document Management and Configuration Control Information

- Revision Number: 1
- Revision Release Date: 19.04.2022
- Purpose of Revision: Initial release
- Scope of Revision: Initial release

1.2. Purpose and Scope of the SAD

This SAD specifies the software architecture for NLP tool to predict economy chabge**.** All information regarding the software architecture may be found in this document, although much information is incorporated by reference to other documents.

What is software architecture? The software architecture for a system[1] is the structure or structures of that system, which comprise software elements, the externally-visible properties of those elements, and the relationships among them [Bass 2003]. "Externally visible” properties refers to those assumptions other

[1] Here, a system may refer to a system of systems.

elements can make of an element, such as its provided services, performance characteristics, fault handling, shared resource usage, and so on. This definition provides the basic litmus test for what information is included in this SAD, and what information is relegated to downstream documentation.

Elements and relationships. The software architecture first and foremost embodies information about how the elements relate to each other. This means that architecture specifically omits certain information about elements that does not pertain to their interaction. Thus, a software architecture is an *abstraction* of a system that suppresses details of elements that do not affect how they use, are used by, relate to, or interact with other elements. Elements interact with each other by means of interfaces that partition details about an element into public and private parts. Software architecture is concerned with the public side of this division, and that will be documented in this SAD accordingly. On the other hand, private details of elements—details having to do solely with internal implementation—are not architectural and will not be documented in a SAD.

Multiple structures. The definition of software architecture makes it clear that systems can and do comprise more than one structure and that no one structure holds the irrefutable claim to being the architecture. The neurologist, the orthopedist, the hematologist, and the dermatologist all take a different perspective on the structure of a human body. Ophthalmologists, cardiologists, and podiatrists concentrate on subsystems. And the kinesiologist and psychiatrist are concerned with different aspects of the entire arrangement's behavior. Although these perspectives are pictured differently and have very different properties, all are inherently related; together they describe the architecture of the human body. So it is with software. Modern systems are more than complex enough to make it difficult to grasp them all at once. Instead, we restrict our attention at any one moment to one (or a small number) of the software system's structures. To communicate meaningfully about an architecture, we must make clear which structure or structures we are discussing at the moment—which *view* we are taking of the architecture. Thus, this SAD follows the principle that documenting a software architecture is a matter of documenting the relevant views and then documenting information that applies to more than one view.

For example, all non-trivial software systems are partitioned into implementation units; these units are given specific responsibilities, and are the basis of work assignments for programming teams. This kind of element will comprise programs and data that software in other implementation units can call or access, and programs and data that are private. In large projects, the elements will almost certainly be subdivided for assignment to sub-teams. This is one kind of structure often used to describe a system. It is a very static structure, in that it focuses on the way the system's functionality is divided up and assigned to implementation teams.

Other structures are much more focused on the way the elements interact with each other at runtime to carry out the system's function. Suppose the system is to be built as a set of parallel processes. The set of processes that will exist at runtime, the programs in the various implementation units described previously that are strung together sequentially to form each process, and the synchronization relations among the processes form another kind of structure often used to describe a system.

None of these structures alone is *the* architecture, although they all convey architectural information. The architecture consists of these structures as well as many others. This example shows that since architecture can comprise more than one kind of structure, there is more than one kind of element (e.g., implementation unit and processes), more than one kind of interaction among elements (e.g., subdivision and synchronization), and even more than one context (e.g., development time versus runtime). By intention, the definition does not specify what the architectural elements and relationships are. Is a software element an object? A process? A library? A database? A commercial product? It can be any of these things and more.

These structures will be represented in the views of the software architecture that are provided in Section 3.

Behavior. Although software architecture tends to focus on structural information, *behavior of each element is part of the software architecture* insofar as that behavior can be observed or discerned from the point of view of another element. This behavior is what allows elements to interact with each other, which is clearly part of the

software architecture and will be documented in the SAD as such. Behavior is documented in the element catalog of each view.

1.3. Viewpoint Definitions

The SAD employs a stakeholder-focused, multiple view approach to architecture documentation, as required by ANSI/IEEE 1471-2000, the recommended best practice for documenting the architecture of software-intensive systems [IEEE 1471].

As described in Section 1.2, a software architecture comprises more than one software structure, each of which provides an engineering handle on different system qualities. A *view* is the specification of one or more of these structures, and documenting a software architecture, then, is a matter of documenting the relevant views and then documenting information that applies to more than one view [Clements 2002].

ANSI/IEEE 1471-2000 provides guidance for choosing the best set of views to document, by bringing stakeholder interests to bear. It prescribes defining a set of viewpoints to satisfy the stakeholder community. A viewpoint identifies the set of concerns to be addressed, and identifies the modeling techniques, evaluation techniques, consistency checking techniques, etc., used by any conforming view. A view, then, is a viewpoint applied to a system. It is a representation of a set of software elements, their properties, and the relationships among them that conform to a defining viewpoint. Together, the chosen set of views show the entire architecture and all of its relevant properties. A SAD contains the viewpoints, relevant views, and information that applies to more than one view to give a holistic description of the system.

The remainder of Section 1.5 defines the viewpoints used in this SAD. The following table summarizes the stakeholders in this project and the viewpoints that have been included to address their concerns.

Table 1: Stakeholders and Relevant Viewpoints

Stakeholder	**Viewpoint(s) that apply to that class of stakeholder's concerns**
Kravtsov Andrii	Student and project developer

1.3.1. "Student and project developer" Viewpoint Definition

1.3.1.1. Abstract

This viewpoint summarizes the views of a student developing this project and their concerns

1.3.1.2. Stakeholders and Their Concerns Addressed

The main concern of a student is to get the highest grade possible

1.3.1.3. Elements, Relations, Properties, and Constraints

Project consists of 3 primary components:

- Text pre-processor
- Semantic scorer
- Prediction module

A dataset of news with timestamps and associated value of desired metric is fed into the program. The software processes text and attempts to predict change on metric based on the news.

1.3.1.4. Language(s) to Model/Represent Conforming Views

Project built using python and HuggingFace database for datasets

2.Architecture Background

2.1. Problem Background

CONTENTS OF THIS SECTION: The sub-parts of Section 2.1 explain the constraints that provided the significant influence over the architecture.

2.1.1. System Overview

The general purpose of the system is to provide predictions on economy change based on news sentiment. To achieve this a neural network is used to correlate news sentiment to economy change tendency and predict based on the sentiment of news.

2.1.2. Goals and Context

The main goal of the project was to achieve relatively accurate prediction of economy based on news.

2.1.3. Significant Driving Requirements

The most important requirements are as follows:

- Automated text pre-processing
- Automated sentiment analysis
- Automated model training
- Prediction based on any news format

2.2. Solution Background

In order to satisfy upper mentioned requirements natural language processing and neural network solutions were chosen for their flexibility and ability to correlate data.

As for programing language Python 3.7 was used due to it's popularity and a wide variety of modules provided by the community. .

2.2.1. Architectural Approaches

A web server architecture was chosen due to telegram APIs functioning as a HHTP based request handler. In order to most efficiently handle these request a web server suits best.

2.2.2. Analysis Results

Based on similar projects and general project background, upper mentioned architecture is deemed as an optimal solution and fit for this purpose.

2.2.3. Requirements Coverage

In order for the solution to function it needs:

- a machine with enough space for a dataset
- reasonable processing power
- Python interpreter installed
- A proper dataset

3.Referenced Materials

Barbacci 2003	Barbacci, M.; Ellison, R.; Lattanze, A.; Stafford, J.; Weinstock, C.; & Wood, W. *Quality Attribute Workshops (QAWs)*, Third Edition (CMU/SEI-2003-TR-016). Pittsburgh, PA: Software Engineering Institute, Carnegie Mellon University, 2003. <http://www.sei.cmu.edu/publications/documents/03.reports/03tr016.html>.
Bass 2003	Bass, Clements, Kazman, *Software Architecture in Practice,* second edition, Addison Wesley Longman, 2003.
Clements 2001	Clements, Kazman, Klein, *Evaluating Software Architectures: Methods and Case Studies,* Addison Wesley Longman, 2001.
Clements 2002	Clements, Bachmann, Bass, Garlan, Ivers, Little, Nord, Stafford, *Documenting Software Architectures: Views and Beyond*, Addison Wesley Longman, 2002.
IEEE 1471	ANSI/IEEE-1471-2000, *IEEE Recommended Practice for Architectural Description of Software-Intensive Systems*, 21 September 2000.
“Programming in Python”	Nico Loubser, 2021

4.Directory

4.1. Index

4.2. Glossary

Term	Definition
software architecture	The structure or structures of that system, which comprise software elements, the externally visible properties of those elements, and the relationships among them [Bass 2003]. "Externally visible" properties refer to those assumptions other elements can make of an element, such as its provided services, performance characteristics, fault handling, shared resource usage, and so on.
view	A representation of a whole system from the perspective of a related set of concerns [IEEE 1471]. A representation of a particular type of software architectural elements that occur in a system, their properties, and the relations among them. A view conforms to a defining viewpoint.
view packet	The smallest package of architectural documentation that could usefully be given to a stakeholder. The documentation of a view is composed of one or more view packets.
viewpoint	A specification of the conventions for constructing and using a view; a pattern or template from which to develop individual views by establishing the purposes and audience for a view, and the techniques for its creation and analysis [IEEE 1471]. Identifies the set of concerns to be addressed, and identifies the modeling techniques, evaluation techniques, consistency checking techniques, etc., used by any conforming view.

4.3. Acronym List

API	Application Programming Interface; Application Program Interface; Application Programmer Interface
ATAM	Architecture Tradeoff Analysis Method
CMM	Capability Maturity Model
CMMI	Capability Maturity Model Integration
CORBA	Common object request broker architecture
COTS	Commercial-Off-The-Shelf
EPIC	Evolutionary Process for Integrating COTS-Based Systems
IEEE	Institute of Electrical and Electronics Engineers
KPA	Key Process Area
OO	Object Oriented
ORB	Object Request Broker
OS	Operating System
QAW	Quality Attribute Workshop
RUP	Rational Unified Process
SAD	Software Architecture Document
SDE	Software Development Environment
SEE	Software Engineering Environment
SEI	Software Engineering Institute Systems Engineering & Integration Software End Item
SEPG	Software Engineering Process Group
SLOC	Source Lines of Code
SW-CMM	Capability Maturity Model for Software
CMMI-SW	Capability Maturity Model Integrated - includes Software Engineering
UML	Unified Modeling Language

5. Figures & Tables

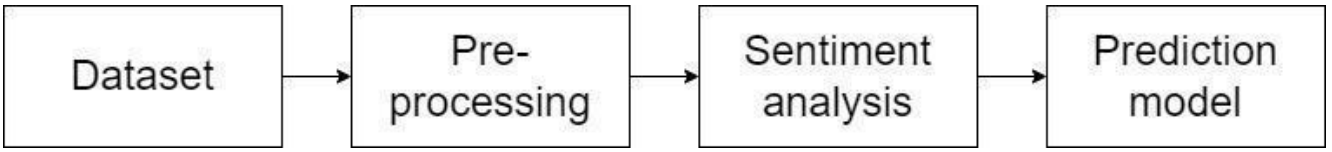

Figure 1: Architecture

Table 2: NLP vs Deep Learning

Deep Learning vs NLP

Comparison Chart

Deep Learning	NLP
Deep learning is a subset of the field of machine learning based on artificial neural networks that teaches computers to learn by example.	Natural Language Processing is the ability of a computer program to understand human language as it is spoken.
It is a function of artificial intelligence that imitates human brain in processing data and creating patterns for decision making uses.	It investigates the use of computers to process or to understand human languages for the purpose of performing useful tasks.
It is an AI function that mimics human learning and thinking process to process data that is both unstructured and unlabeled.	NLP is the relationship between computers and human language.
Deep learning algorithms are used in Google language translation services, Alexa, self-driving cars, voice synthesis, facial recognition, etc.	Applications include machine translation, automatic summarization, automatic speech recognition, chatbots, market intelligence, customer service, etc.

DB Difference Between.net