НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ імені ІГОРЯ СІКОРСЬКОГО»

Теплоенергетичний факультет

Кафедра автоматизації проектування енергетичних процесів і систем

До захисту допущено

Завідувач кафедри

_____ О.В. Коваль
(підпис) (ініціали, прізвище)

“___”_____2020р

Дипломна робота
на здобуття ступеня бакалавра

з напряму підготовки 121 “ Інженерія програмного забезпечення “

на тему___“ Попередня обробка даних (в тому числі заповнення відсутніх даних, видалення аномальних, згладжування даних для вирішення завдання зниження випадкових шумів, стиснення і нормалізація даних) “_____

Виконав (-ла): студент (-ка) __4___курсу, групи_ТІ-61_

_____Волков Олександр Олегович_____ _____
(прізвище, ім’я, по батькові) (підпис)

Керівник в.о. зав. Кафедри, к.т.н,доцент доцент Коваль О.В._ _____
(посада, вчене звання, науковий ступінь, прізвище та ініціали) (підпис)

Консультант _____ _____ _____
(назва розділу) (вчені ступінь та звання, прізвище, ініціали) (підпис)

Рецензент_____к.т.н. Сенченко В’ячеслав Родіонович_____ _____
(посада, вчене звання, науковий ступінь, прізвище та ініціали) (підпис)

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

Київ – 2020

Національний технічний університет України
“Київський політехнічний інститут імені Ігоря Сікорського”

Факультет теплоенергетичний

Кафедра автоматизації проектування енергетичних процесів і систем

Рівень вищої освіти перший , бакалаврський

Напрям підготовки 121 “ Інженерія програмного забезпечення “

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ О.В. Коваль
(підпис)
”___”_____2020р.

ЗАВДАННЯ
на дипломну роботу студенту

_____Волков Олександр Олегович_____
(прізвище, ім’я, по батькові)

1. Тема роботи ____“Попередня обробка даних (в тому числі заповнення відсутніх даних, видалення аномальних, згладжування даних для вирішення завдання зниження випадкових шумів, стиснення і нормалізація даних)”_____
керівник роботи ____Коваль Олександр Васильович к.т.н.,доцент_____
(прізвище, ім’я, по батькові науковий ступінь, вчене звання)

затверджена наказом вищого навчального закладу від ”__”___ 201_р. № __

2. Строк подання студентом роботи _____
3. Вихідні дані до роботи __див. Пояснювальну записку_____
_____
_____

4.Зміст розрахунково-пояснювальної записки (перелік завдань, які потрібно розробити)

1) Огляд існуючих рішень, 2) Огляд архітектури нейронних мереж, 3) Методи оптимізації моделювання, 4) Розробка та результати

5. Перелік ілюстративного матеріалу
45 ілюстрації, 2 таблиці _____
_____

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв

7. Дата видачі завдання ”___”_____20____р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів виконання дипломної роботи	Термін виконання етапів роботи	Примітки
1.	Вивчення та аналіз задачі	10.01.2019	
2	Розробка архітектури та загальної структури системи	01.02.2020	
3.	Розробка структур окремих підсистем	03.03.2020	
4.	Програмна реалізація системи	27.03.2020	
5.	Оформлення пояснювальної записки	30.04.2020	
6.	Захист програмного продукту	11.06.2020	
7.	Передзахист	11.06.2020	
8.	Захист	15.06.2020	

Студент _____ _____Волков О.О._____
(підпис) (прізвище та ініціали,)

Керівник роботи _____ _____Коваль О.В._____
(підпис) (прізвище та ініціали,)

АНОТАЦІЯ

Волков О.О. «Попередня обробка даних (в тому числі заповнення відсутніх даних, видалення аномальних, згладжування даних для вирішення завдання зниження випадкових шумів, стиснення і нормалізація даних)». КПІ ім. Ігоря Сікорського, Київ, 2020.

Дипломний проект присвячений розробці системи, що слугуватиме для машинної класифікації текстових відгуків користувачів на придбані товари на позитивні та негативні.

Текстова документація містить інформацію про алгоритми, методи оптимізації, розробку, тестування системи класифікації відгуків на фільми. Велика увага була приділена вибору технічних засобів для реалізації та проектуванню, оскільки подана задача може бути вирішена з використанням великої кількості різних алгоритмів.

Ключові слова: попередня обробка даних, система аналізу тексту, Python, Tensorflow, нейронні мережі, відгук, класифікація, навчання з учителем.

Розмір пояснювальної записки – 76 аркушів, містить 45 ілюстрації, 2 таблиці.

ANNOTATION

Volkov O. " Pre-processing of data (including filling in missing data, deleting abnormal, smoothing data to solve the problem of reducing random noise, compression and normalization of data)". Igor Sikorsky Kyiv Polytechnic Institute, Kyiv, 2020.

Diploma project dedicated to develop systems that will serve as an assistant in machine classification of the users' textual reviews of the purchased goods on the positive and the negative.

Text documentation contains information about algorithms, optimization methods, development and testing of movie reviews classification system. Great attention was paid to the choice of means to implement and design, as given problem could be solved with applying of a large number of different algorithms.

Keywords: data pre-processing, text processing system, Python, Tensorflow, neural networks, review, classification, supervised learning.

Size explanatory notes - 76 sheets, contains 45 illustrations, 2 tables.

ЗМІСТ

ПЕРЕЛІК ТЕРМІНІВ……..……..……..5

ВСТУП ...…….……...……..6

1. ОГЛЯД ІСНУЮЧИХ РІШЕНЬ…...……..……8

1.1 Огляд методів попередньої обробки даних.....8

1.1.1 Метод очищення даних.....12

1.1.2 Метод зміни (суперечки) даних.....14

1.2 Огляд алгоритмів вирішення задачі сентиментального аналізу……....15

1.2.1 Алгоритми навчання з учителем……..15

1.2.2 Алгоритми навчання без учителя…….……...25

1.3 Огляд різновидів сентиментальної класифікації…….……..……30

1.3.1 Наявність ієрархічності……..……...31

1.3.2 Рівень класифікації……..……..32

1.4 Огляд мов та технологій програмування……..……...……32

1.4.1 Python……..……34

1.4.2 R …….……...36

1.4.3 Matlab……..……37

1.4.4 LabVIEW…….……...39

1.4.5 TensorFlow……..……40

2. ОГЛЯД АРХІТЕКТУРИ НЕЙРОННИХ МЕРЕЖ…...……...……45

2.1 Детальний огляд загальної архітектури нейронних мереж…….…...…45

2.2 Детальний огляд архітектур нейнонних мереж…….……....52

2.2.1 Багатошаровий перцептрон…….……52

2.2.2 Згорткова нейронна мережа…….……53

2.2.3 Рекурсивна нейронна мережа……..……54

2.2.4 Рекурентна нейронна мережа……..……55

2.2.5 Нейронні мережі з довгою короткочасною пам’яттю……...55

3. МЕТОДИ ОПТИМІЗАЦІЇ МОДЕЛЮВАННЯ……52
3.1 Підготовка даних для нейроної мережі…52
3.2 Вплив методу навчання нейронної мережі на швидкість навчання…..59
3.3 Вибір функції втрат……61
3.3.1 Середньоквадратична помилка……62
3.3.2 Помилка перехресної ентропії……62
3.4 Вибір коефіцієнта швидкості навчання……63
4. РОЗРОБКА ТА РЕЗУЛЬТАТИ……66
4.1 Задачі реалізації й вибір інтерфейсу……66
4.2 Робота з даними……68
4.3 Результати моделювання……70
4.4 Можливості вдосконалення алгоритму……71
ВИСНОВКИ……73
ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ……74

ПЕРЕЛІК ТЕРМІНІВ

1) сентиментальний аналіз – аналіз, що передбачає визначення ставлення автору тексту до об’єкту обговорення (наприклад позитивна оцінка покупця до придбаного товару);

2) попередня обробка даних - це техніка обміну даними, яка використовується для перетворення необроблених даних у корисний та ефективний формат.

3) градієнтний спуск – метод ітеративного зменшення значень функції шляхом зміни аргументів у шляху, протилежному до її градієнту;

4) mini-batch – деяка невелика вибірка фіксованої величини з даних (дослівно з англійскої – міні-пакет);

5) функція активації – функція, що застосовується на виході сигналу нейрона для приведення сигналу до того чи іншого вигляду (наприклад, для зміни діапазону значень вихідного сигналу на діапазон від нуля до одиниці, що дуже часто застосовується у нейронних мережах);

6) ReLU (Rectifier) – функція активації, вихід якої дорівнює входу, якщо вхід більше нуля, та вихід якої дорівнює нулю, якщо вхід менше нуля;

7) softmax функція активації – при застосуванні якої вихід кожного нейрона залежить від виходів всіх інших нейронів шару, а сума вихідних значень шару дорівнює одиниці.

ВСТУП

Актуальність теми

Сьогодні технології штучного інтелекту швидко розвиваються. Широкий спектр складних задач, що технології штучного інтелекту здатні вирішувати без участі людини, робить їх дуже привабливими для розробників. Розвиток та застосування технологій машинного навчання нині дозволяють вдосконалити й здешевшити вирішення задач, які раніше могли бути вирішені тільки за участі людини.

Разом з тим, швидкий розвиток веб-технологій призводить до стрімкого збільшення доступної людям текстової інформації, з якою все складніше взаємодіяти за участю лише людських ресурсів. У тому числі, онлайн-кінотеатри, а також компанії, що займаються продажем великого обсягу товарів, мають потребу оцінювати їхню популярність серед користувачів, для чого їм потрібно проаналізувати великий обсяг текстових даних. Це призводить до необхідності машинного аналізу текстової інформації.

Мета і задачі дослідження

Метою даної роботи є знаходження оптимального методу попередньої обробки даних в рамках вирішення проблеми сентиментального аналізу великого обсягу (25000) текстових відгуків на фільми.

Задачею є дослідження актуальних методів попередньої обробки даних та новітніх підходів щодо їх вдосконалення та впровадження.

Об’єкт і предмет дослідження

Об’єктом дослідження є відгуки на фільми. Предметом дослідження є методи попередньої обробки використані в рамках поставленої задачі.

Наукова новизна одержаних результатів

Новизна даної роботи полягає в аналізі існуючих рішень і знаходженні оптимального методу попередньої обробки даних для подальшої сентиментальної класифікації текстових відгуків.

Практичне значення наукових результатів

Розроблений метод можна використовувати для удосконалення та здешевшення попередньої обробки даних та класифікації великих об’ємів текстових даних.

Структура роботи

У розділі 1 виконаний огляд готових рішень та предметної області.

У розділі 2 проведений огляд основних архітектур нейронних мереж.

У розділі 3 оглянуті та досліжені методи оптимізхації моделювання при використанні нейронних мереж.

У розділі 4 реалізовано модель попередньої обробки відгуків глядачів на фільми на практиці.

1 ОГЛЯД ІСНУЮЧИХ РІШЕНЬ ТА ПРЕДМЕТНОЇ ОБЛАСТІ

З розвитком інтернету з’являються все більші за обсягом користувачів сайти. Їхнім розробникам доводиться взаємодіяти з усе більшими об’ємами даних, адже все більше дій, пов’язаних, наприклад, з придбанням тих чи інших послуг або зі спілкуванням людей, проводиться через інтернет через більшу зручність та швидкість такої взаємодії. В часи повільного інтернету спектр послуг, який можна було б придбати у мережі, був досить малим. В свою чергу, сучасний інтернет це сукупність мільйонів сайтів, багато з яких у свою чергу мають багатомільйонну аудиторію, дані про активність якої вони постійно записують. Це призвело до накопичення величезних об’ємів даних, що у свою чергу надало можливість проводити аналіз цих даних та вилучати з них певну корисну інформацію.

Таким чином, для того, щоб сайт залишився конкурентоспроможним, при його розвитку має також вирішуватися така задача, як збір та аналіз даних щодо думки цільової аудиторії з метою виявлення тенденцій та пріоритетних напрямків розвитку.

Вирішення даної задачі потребує за наявності певного тексту користувача засобами штучного інтелекту виділити об’єкт обговорення та відношення користувача до нього. Важливими складовими вирішення даної задачі є аналіз тональності тексту, або сентиментальний аналіз, який як раз дозволяє визначити відношення автора користувача до об’єкту обговорення та попередня обробка текстових даних.

За весь час існування даної проблеми з’явилися різні підходи, алгоритми та принципи її вирішення.

1.1 Огляд методів попередньої обробки даних

Аналізувати можна як якісні, так і неякісні дані. Результат буде досягнутий і в тому, і в іншому випадку. Для забезпечення якісного аналізу необхідно

проведення попередньої обробки даних, яка є необхідним етапом процесу Data Mining.

Оцінювання якості даних. Дані, отримані в результаті збору, повинні відповідати певним критеріям якості. Таким чином, можна виділити важливий підетапів процесу Data Mining - оцінювання якості даних.

Якість даних (Data quality) - це критерій, який визначає повноту, точність, своєчасність і можливість інтерпретації даних.

Дані можуть бути високої якості і низької якості, останні - це так звані брудні або "погані" дані.

Дані високої якості - це повні, точні, своєчасні дані, які піддаються інтерпретації.

Такі дані забезпечують отримання якісного результату: знань, які зможуть підтримувати процес прийняття рішень.

Прогноз. Багато компаній стали звертати більше уваги на якість даних, оскільки низька якість коштує грошей в тому сенсі, що веде до зниження продуктивності, прийняття неправильних бізнес-рішень і неможливості отримати бажаний результат, а також ускладнює виконання вимог законодавства. Тому компанії дійсно мають намір робити конкретні дії для вирішення проблем якості даних.

Реальність. Дана тенденція зберігається, особливо в індустрії фінансових послуг. В першу чергу це відноситься до фірм, які намагається виконувати угоду Basel II. Неякісні дані не можуть використовуватися в системах оцінки ризиків, які застосовуються для установки цін на кредити і обчислення потреб організації в капіталі. Цікаво відзначити, що істотно змінилися погляди на способи вирішення проблеми якості даних. Спочатку менеджери звертали основну увагу на інструменти оцінки якості, вважаючи, що "власник" даних повинен вирішувати проблему на рівні джерела, наприклад, очищаючи дані та перенавчитися співробітників. Але зараз їх погляди суттєво змінилися. Поняття якості даних набагато ширше, ніж просто їх акуратне введення в

систему на першому етапі. Сьогодні вже багато хто розуміє, що якість даних повинне забезпечуватися процесами вилучення, перетворення і завантаження (Extraction, Transformation, Loading - ETL), а також отримання даних з джерел, які готують дані для аналізу.

Розглянемо поняття якості даних більш детально.

Дані низької якості, або брудні дані - це відсутні, неточні або даремні дані з точки зору практичного застосування (наприклад, представлені в невірному форматі, який не відповідає стандарту). Брудні дані з'явилися не сьогодні, вони виникли одночасно з системами введення даних.

Брудні дані можуть з'явитися з різних причин, такі як помилка при введенні даних, використання інших форматів представлення або одиниць вимірювання, невідповідність стандартам, відсутність своєчасного оновлення, невдале оновлення всіх копій даних, невдале видалення записів-дублікатів і т.д. Необхідно оцінити вартість наявності брудних даних; іншими словами, наявність брудних даних може дійсно привести до фінансових втрат і юридичної відповідальності, якщо їх присутність не запобігає або вони не виявляються і не очищаються.

Для більш детального знайомства з брудними даними можна рекомендувати, де представлена таксономія 33 типів брудних даних і також розроблена таксономія методів запобігання або розпізнавання і очищення даних. Описано різні типи брудних даних, серед них виділено такі групи:

• брудні дані, які можуть бути автоматично виявлені і очищені;

• дані, поява яких може бути припинено;

• дані, які непридатні для автоматичного виявлення і очищення;

• дані, поява яких неможливо запобігти.

Тому важливо розуміти, що спеціальні засоби очищення можуть впоратися не з усіма видами брудних даних.

Розглянемо найбільш поширені види брудних даних:

• пропущені значення;

• дублікати даних;

• шуми і викиди.

Пропущені значення (Missing Values).

Деякі значення даних можуть бути пропущені у зв'язку з тим, що:

• дані взагалі не були зібрані (наприклад, при анкетуванні прихований вік);

• деякі атрибути можуть бути незастосовні для деяких об'єктів (наприклад, атрибут "річний дохід" непридатний до дитини).

Як ми можемо бути з пропущеними даними?

• Виключити об'єкти з пропущеними значеннями з обробки.

• Розрахувати нові значення для пропущених даних.

• Ігнорувати пропущені значення в процесі аналізу.

• Замінити пропущені значення на можливі значення.

Дублювання даних (Duplicate Data).

Набір даних може включати продубльовані дані, тобто дублікати.

Дублікатами називаються записи з однаковими значеннями всіх атрибутів.

Наявність дублікатів в наборі даних може бути способом підвищення значущості деяких записів. Така необхідність іноді виникає для особливого виділення певних записів з набору даних. Однак в більшості випадків, продубльовані дані є результатом помилок при підготовці даних.

Як ми можемо бути з продубльованими даними?

Існує два варіанти обробки дублікатів. При першому варіанті видаляється вся група записів, що містить дублікати. Цей варіант використовується в тому випадку, якщо наявність дублікатів викликає недовіру до інформації, повністю її знецінює.

Другий варіант полягає в заміні групи дублікатів на один унікальний запис.

Шуми і викиди.

Викиди - різко відрізняються об'єкти або спостереження в наборі даних.

Шуми і викиди є досить загальною проблемою в аналізі даних. Викиди можуть як являти собою окремі спостереження, так і бути об'єднаними в якісь

групи. Завдання аналітика - не тільки їх виявити, але і оцінити ступінь їх впливу на 210 результати подальшого аналізу. Якщо викиди є інформативною частиною аналізованого набору даних, використовують робастні методи і процедури.

Досить поширена практика проведення двоетапного аналізу - з викидами і з їх відсутністю - і порівняння отриманих результатів.

Різні методи Data Mining мають різну чутливість до викидів, цей факт необхідно враховувати при виборі методу аналізу даних. Також деякі інструменти Data Mining мають вбудовані процедури очищення від шумів і викидів.

Візуалізація даних дозволяє представити дані, в тому числі і викиди, в графічному вигляді. Приклад наявності викидів зображений на діаграмі розсіювання на рис. 1.23. Ми бачимо кілька спостережень, різко відрізняються від інших (що знаходяться на великій відстані від більшості спостережень).

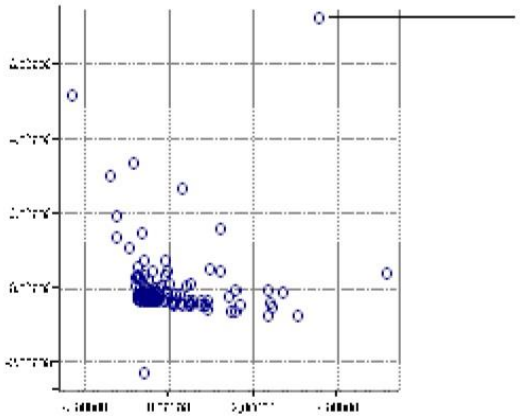


Рис. 1.23. Приклад набору даних з викидами

Очевидно, що результати Data Mining на основі брудних даних не можуть вважатися надійними і корисними. Однак наявність таких даних не обов'язково означає необхідність їх очищення або ж запобігання появи. Завжди повинен бути розумний вибір між наявністю брудних даних і вартістю і / або часом, необхідним для їх очищення.

1.1.1 Метод очищення даних

Очищення даних (data cleaning, data cleansing або scrubbing) займається виявленням і видаленням помилок і невідповідностей в даних з метою поліпшення якості даних.

Проблеми з якістю зустрічаються в окремих наборах даних - таких як файли і бази даних. Коли інтеграції підлягає безліч джерел даних (наприклад в Сховищах, інтегрованих системах баз даних або глобальних інформаційних Інтернет-системах), необхідність в очищенні даних істотно зростає. Це відбувається тому, що джерела часто містять розрізнені дані в різному поданні. Для забезпечення доступу до точних і узгодженим даними необхідна консолідація різних уявлень даних і виключення дублюється інформації. Спеціальні засоби очищення зазвичай мають справу з конкретними областями - в основному це імена і адреси - або ж з виключенням дублікатів.

Перетворення забезпечуються або у формі бібліотеки правил, або користувачем в інтерактивному режимі. Перетворення даних можуть бути автоматично отримані за допомогою засобів узгодження схеми.

Метод очищення даних повинен задовольняти ряду критеріїв.

1. Він повинен виявляти і видаляти всі основні помилки і невідповідності, як в окремих джерелах даних, так і при інтеграції декількох джерел.

2. Метод повинен підтримуватися певними інструментами, щоб скоротити обсяги ручної перевірки і програмування, і бути гнучким в плані роботи з додатковими джерелами.

3. Очищення даних не повинна проводитися у відриві від пов'язаних зі схемою перетворення даних, які виконуються на основі складних метаданих.

4. Функції маппірованія для очищення і інших перетворень даних повинні бути визначені декларативним чином і підходити для використання в інших джерелах даних і в обробці запитів.

5. Інфраструктура технологічного процесу повинна особливо інтенсивно підтримуватися для сховищ даних, забезпечуючи ефективне і надійне

виконання всіх етапів перетворення для безлічі джерел і великих наборів даних.

На сьогоднішній день інтерес до очищення даних зростає. Цілий ряд дослідницьких груп займається загальними проблемами, пов'язаними з очищенням даних, в тому числі, зі специфічними підходами до Data Mining і перетворенню даних на підставі зіставлення схеми. Останнім часом деякі дослідження торкнулися єдиного, більш складного підходу до очищення даних, що включає ряд аспектів перетворення даних, специфічних операторів і їх реалізації.

1.1.2 Метод зміни (суперечки) даних (wrangling data)

Wrangling Data (також відомий як Data Munging) - це процес перетворення даних з початкової «сирої» форми у більш засвоюваний формат та організація наборів з різних джерел у єдине цілісне ціле для подальшої обробки. "Необроблені дані" це будь-які дані сховища (тексти, зображення, записи баз даних), які документально підтверджені, але ще підлягають обробці та повністю інтегровані в систему.

Процес суперечки може бути описаний як "перетравлення" даних (часто їх називають "перетворенням даних", таким чином, альтернативним терміном "обмін даними") і роблячи його корисним (він же корисний) для системи. Це можна описати як етап підготовки до кожної іншої операції, пов'язаної з даними.

Складання даних зазвичай супроводжується картографуванням. Термін "картографування даних" відноситься до елементу процесу складання, який включає ідентифікацію полів вихідних даних до відповідних цільових полів даних. Хоча Wrangling присвячений трансформації даних, Mapping - це з'єднання крапок між різними елементами.

Основна мета складання даних може бути описана як отримання даних у узгодженій формі. Іншими словами, це робить необроблені дані корисними.

Таким чином, Data Wrangling виступає як етап підготовки до операції з вилучення даних. Ці дві операції об'єднані між собою, оскільки ви не можете зробити одну без іншої.

Загалом, складання даних охоплює такі процеси:

- Отримання даних з різних джерел в одне місце
- Складання даних разом відповідно до визначених параметрів
- Очищення даних від шуму чи помилок, відсутніх елементів

Слід зазначити, що Data Wrangling є дещо вимогливою та трудомісткою операцією як з обчислювальних можливостей, так і з людських ресурсів. Складання даних займає половину того, що робить вчений.

Зрештою, прямий результат цього процесу є міцною основою для подальшої обробки даних.

Ця операція включає послідовність наступних процесів:

1. Попередня обробка - початковий стан, що виникає відразу після отримання даних.

2. Стандартизація даних у зрозумілому форматі.

3. Очищення даних від шуму, відсутніх або помилкових елементів.

4. Консолідація даних з різних джерел чи наборів даних у цілісне ціле.

5 Узгодження даних із наявними наборами даних.

6 Фільтрування даних через визначені настройки для обробки.

1.2 Огляд алгоритмів вирішення задачі сентиментального аналізу

Нижче приведено розгляд різновиди алгоритмів вирішення задачі сентиментального аналізу.

1.2.1 Алгоритми навчання з учителем.

Алгоритми машинного навчання з учителем вимагають підготовлені та розмічені набори даних для навчання та тестування моделі. Нижче наведені

основні методи машинного навчання з учителем, що використовуються для сентиментального аналізу.

1.2.1.1 Наївний баєсів класифікатор

Наївний баєсів класифікатор ґрунтується на так званій байєсівській теоремі і особливо підходить, коли розмірність вхідних параметрів велика. Наївний баєсів класифікатор використовує формулу Баєса (формула 1.1), що описує ймовірність події, виходячи з попереднього знання умов, які можуть бути пов'язані з цією подією.

$$P(A|B) = \frac{P(B|A)\,P(A)}{P(B)} \qquad 1.1$$

Для оцінки параметрів використовується метод максимальної правдоподібності, що задає значення параметрів моделі шляхом максимізації функції правдоподібності. Часто для тренування моделі у методі наївного баєсіва класифікатора потрібна менша кількість тренувать, ніж для інших моделей. На рисунку 1.1 зображено приклад застосування наївного баєсіва класифікатора для оцінки ймовірності позитивної оцінки фільму.

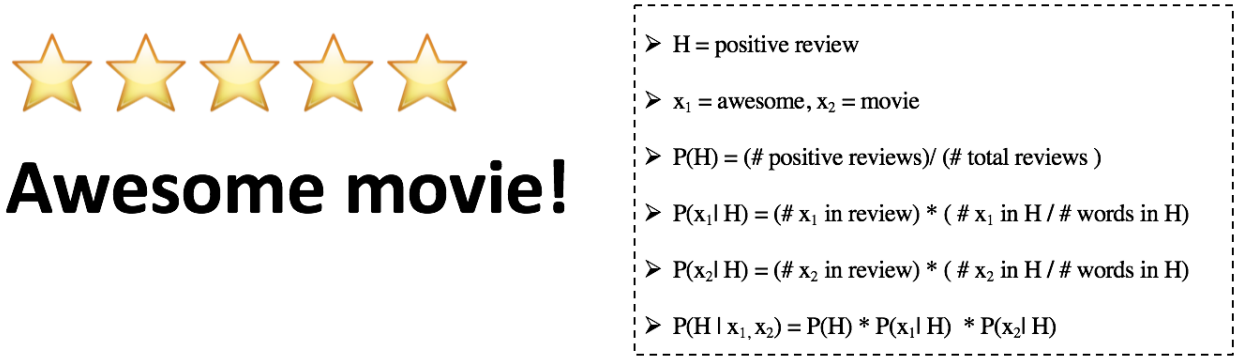


Рисунок 1.1 – Приклад застосування наївного баєсіва класифікатора для оцінки ймовірності позитивної оцінки фільму [1]

Прикладом реалізації сентиментального аналізу із застосуванням наївного баєсіва класифікатора може слугувати проект Dragon Sentiment Classifier (рисунок 1.2), який використовується для оцінки тональності документа як позитивної або негативної.

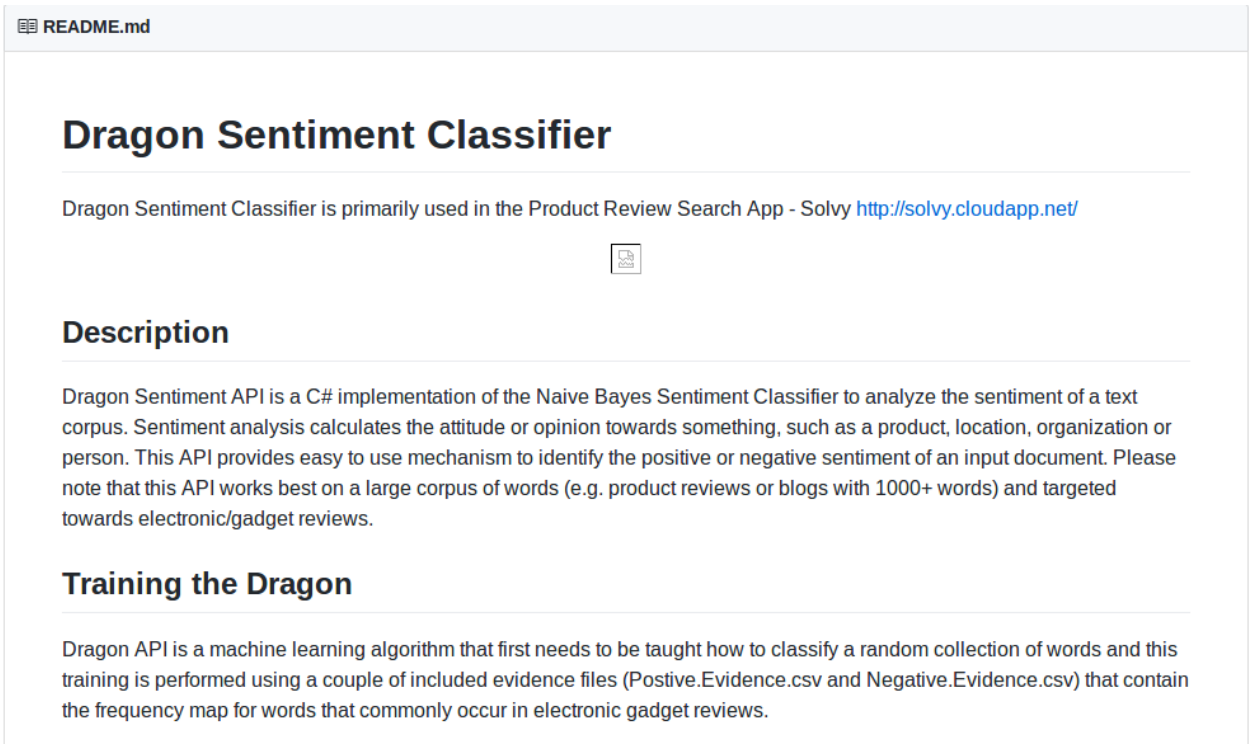


Рисунок 1.2 – Головна сторінка проекту Dragon Sentiment Classifier на сайті Github [2]

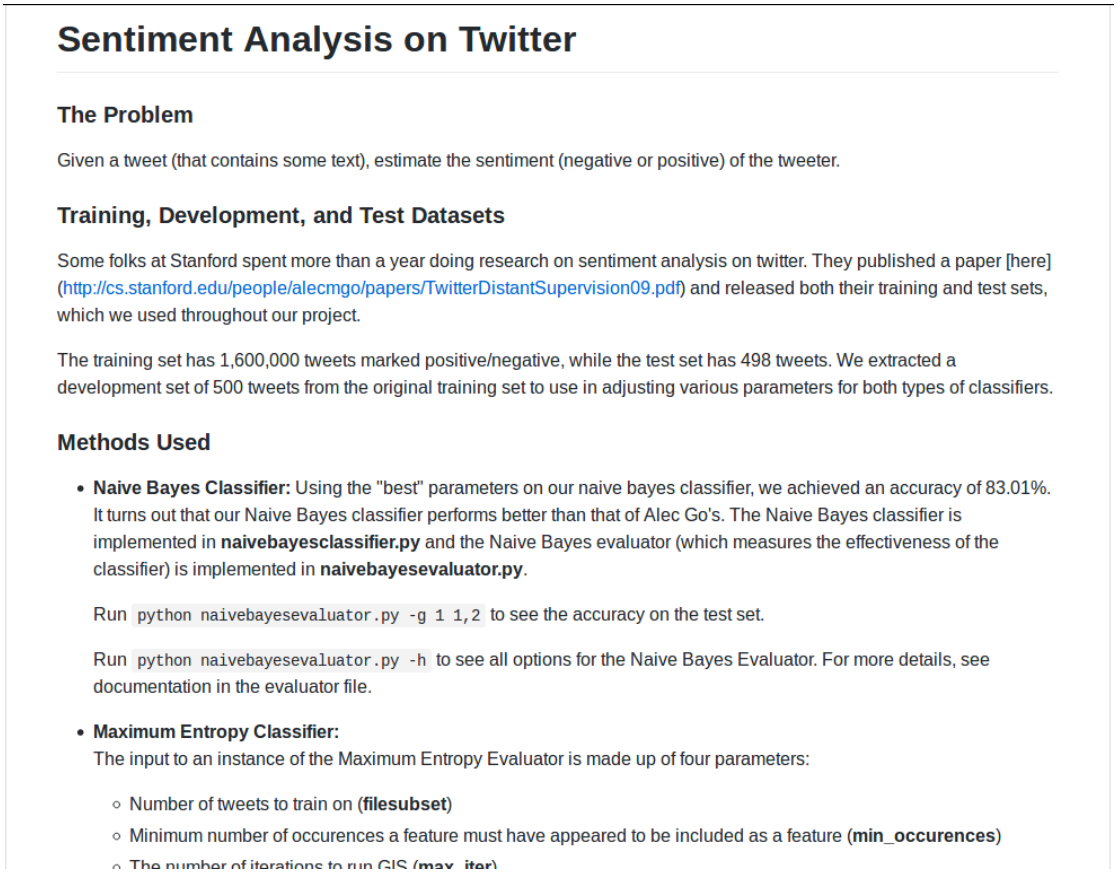
Загалом, незважаючи на свою простоту, наївний баєсів класифікатор може досить часто перевершувати більш складні методи класифікації, що відносяться до методів навчання з учителем, проте є занадто простим для застосування на багатьох наборах даних.

1.2.1.2 Метод максимальної ентропії

У ряді досліджень [3] було відмічено, що метод максимальної ентропії дає кращі результати у задачах порівняння та класифікації тексту, ніж наївний баєсів класифікатор.

Принцип максимальної ентропії - це пошук найкращого розподілу ймовірності серед попередніх тестових даних. Це дає інформацію максимальної ентропії, яка дає належний розподіл. Класифікатори максимальної ентропії зазвичай використовуються як альтернативи наївним баєсівим класифікаторам, оскільки вони не вимагають статистичну незалежність між випадковими величинами (також відомих як features), що слугують незалежними змінними. Проте навчання в такій моделі відбувається повільніше, ніж для наївного баєсіва класифікатора, і, таким чином, може бути недоцільним при великій вибірці даних для навчання моделі. Зокрема, навчання в наївному баєсівому класифікаторі можно звести до підрахунку кількості сполучень незалежних змінних і класів, тоді як у класифікаторі максимальної ентропії ваги, які, як правило, максимізуються за допомогою максимізації апостеріорної ймовірності, можуть бути знайдені тільки використовуючи ітераційну процедуру.

Прикладом реалізації сентиментального аналізу тексту, використовуючи принцип максимума ентропії, може слугувати проект Sentimentstwitter (рисунок 1.3).



Sentiment Analysis on Twitter

The Problem

Given a tweet (that contains some text), estimate the sentiment (negative or positive) of the tweeter.

Training, Development, and Test Datasets

Some folks at Stanford spent more than a year doing research on sentiment analysis on twitter. They published a paper [here] (http://cs.stanford.edu/people/alecmgo/papers/TwitterDistantSupervision09.pdf) and released both their training and test sets, which we used throughout our project.

The training set has 1,600,000 tweets marked positive/negative, while the test set has 498 tweets. We extracted a development set of 500 tweets from the original training set to use in adjusting various parameters for both types of classifiers.

Methods Used

- **Naive Bayes Classifier:** Using the "best" parameters on our naive bayes classifier, we achieved an accuracy of 83.01%. It turns out that our Naive Bayes classifier performs better than that of Alec Go's. The Naive Bayes classifier is implemented in **naivebayesclassifier.py** and the Naive Bayes evaluator (which measures the effectiveness of the classifier) is implemented in **naivebayesevaluator.py**.

  Run `python naivebayesevaluator.py -g 1 1,2` to see the accuracy on the test set.

  Run `python naivebayesevaluator.py -h` to see all options for the Naive Bayes Evaluator. For more details, see documentation in the evaluator file.

- **Maximum Entropy Classifier:**
  The input to an instance of the Maximum Entropy Evaluator is made up of four parameters:

  - Number of tweets to train on (**filesubset**)
  - Minimum number of occurences a feature must have appeared to be included as a feature (**min_occurences**)
  - The number of iterations to run GIS (**max_iter**)

Рисунок 1.3 – Опис проекту Sentimentstwitter на сайті Github [4]

Цей проект був розроблений для аналізу тональності текстів, що були написані користувачами соціальної мережі Twitter (рисунок 1.4).

Рисунок 1.4 – Приклад текстового контента на сайті Twitter [5]

На нижче приведена формула 1.2, що використовується для знайдення ймовірності приналежності об’єкта до певного классу при побудові класифікаційної моделі з використанням принципа максимальної ентропії.

$$p(y = y_i \mid x) = \frac{\exp(\sum_k \lambda_k f_k(y_i, x))}{\sum_{y_j} \exp(\sum_k \lambda_k f_k(y_j, x))} \quad 1.2$$

Ми підсумовували, що метод максимальної ентропії не робить жодних припущень щодо взаємозв'язку між незалежними змінними, і тому потенційно може поліпшити результати модулювання у порівнянні з наївним баєсівим класифікатором, якщо припущення про незалежність не буде виконане.

1.2.1.3 Метод опорних векторів

Метод опорних векторів - це модель навчання з учителем, основна ідея якої полягає в тому, щоб знайти гіперплощину, що розділяла б подані дані на класи з максимальною відстанню між об'єктами цих класів та площиною. Принцип роботи даного методу добре проілюстровано на рисунку 1.5. Цей метод добре знаходить оптимальні рішення задач тектової класифікації. Є декілька різних варіацій алгоритмів опорних векторів: лінійні алгоритми, алгоритми з м'яким зазором, нелінійні алгоритми.

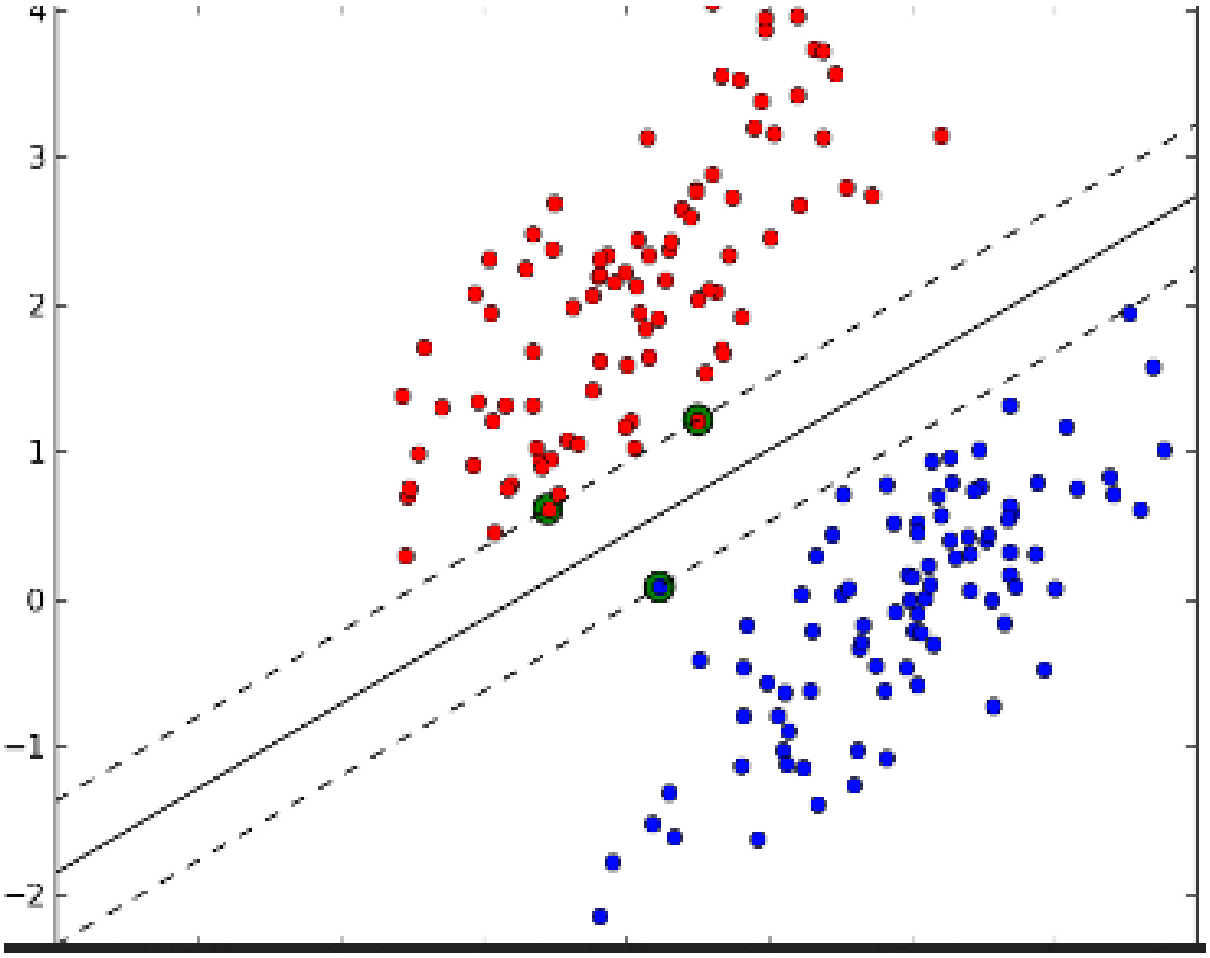
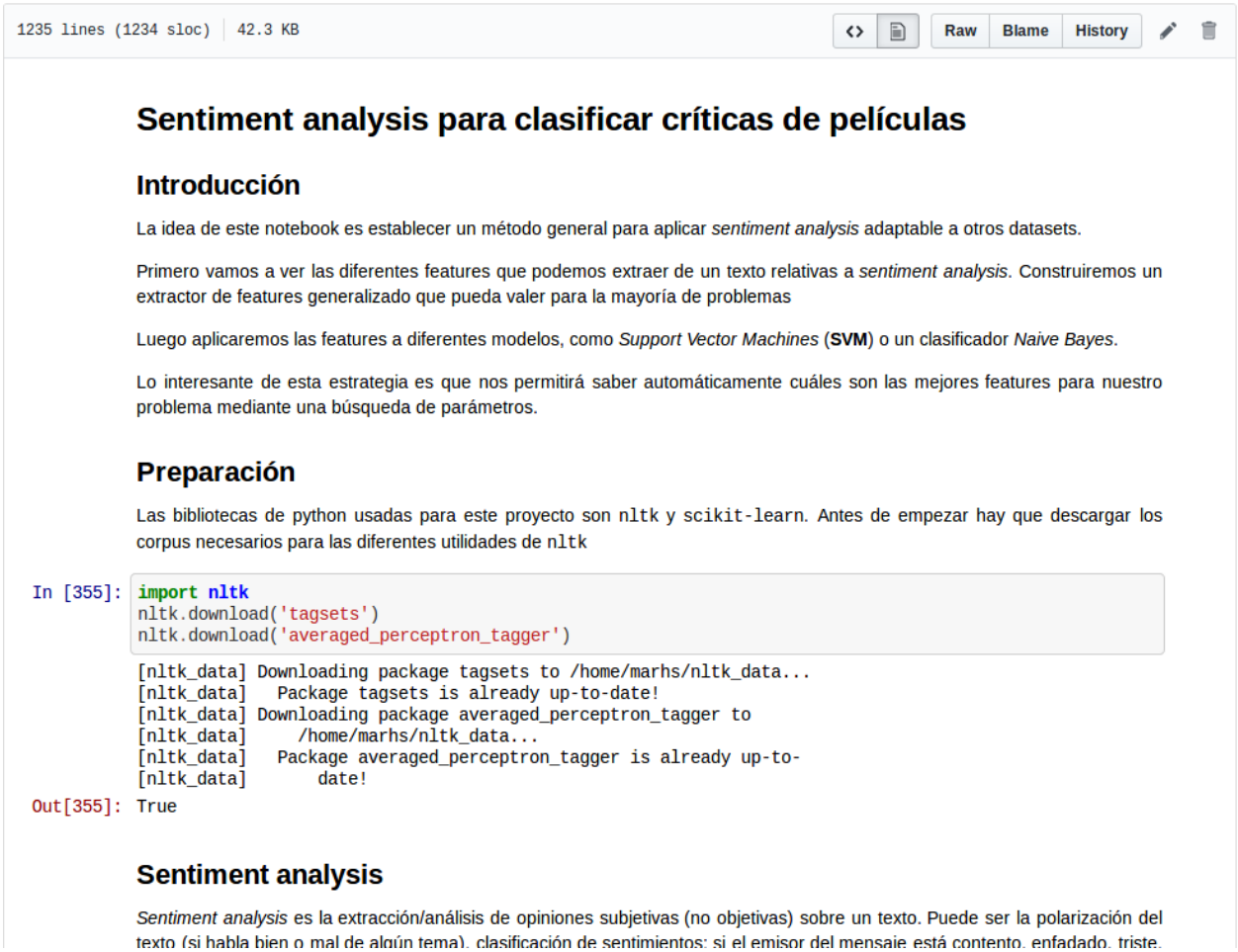


Рисунок 1.5 – ілюстрація принципу класифікації об’єктів методом опорних векторів

Метод опорних векторів в даний час є одним з кращих методів для ряду класифікаційних завдань, починаючи від текстових до геномних даних. У ряді досліджень [3] було виявлено, що він ефективний, точний і добре працює при невеликій кількості навчальних даних. Таким чином можна зробити висновок, що метод опорних векторів перевершує методи наївного баєсового класифікатора та метод максимума ентропії для стандартної текстової класифікації.

Прикладом реалізації даного метода на практиці може слугувати проект Sentimental Analysis, головну сторінку опису якого зображено на рисунку 1.6.



1235 lines (1234 sloc) | 42.3 KB

Raw Blame History

Sentiment analysis para clasificar críticas de películas

Introducción

La idea de este notebook es establecer un método general para aplicar *sentiment analysis* adaptable a otros datasets.

Primero vamos a ver las diferentes features que podemos extraer de un texto relativas a *sentiment analysis*. Construiremos un extractor de features generalizado que pueda valer para la mayoría de problemas

Luego aplicaremos las features a diferentes modelos, como *Support Vector Machines* (**SVM**) o un clasificador *Naive Bayes*.

Lo interesante de esta estrategia es que nos permitirá saber automáticamente cuáles son las mejores features para nuestro problema mediante una búsqueda de parámetros.

Preparación

Las bibliotecas de python usadas para este proyecto son `nltk` y `scikit-learn`. Antes de empezar hay que descargar los corpus necesarios para las diferentes utilidades de `nltk`

In [355]:
```
import nltk
nltk.download('tagsets')
nltk.download('averaged_perceptron_tagger')
```

```
[nltk_data] Downloading package tagsets to /home/marhs/nltk_data...
[nltk_data]   Package tagsets is already up-to-date!
[nltk_data] Downloading package averaged_perceptron_tagger to
[nltk_data]     /home/marhs/nltk_data...
[nltk_data]   Package averaged_perceptron_tagger is already up-to-
[nltk_data]       date!
```

Out[355]: True

Sentiment analysis

Sentiment analysis es la extracción/análisis de opiniones subjetivas (no objetivas) sobre un texto. Puede ser la polarización del texto (si habla bien o mal de algún tema), clasificación de sentimientos: si el emisor del mensaje está contento, enfadado, triste,

Рисунок 1.6 — Приклад реалізації метода SVM на практиці [6]

1.2.1.4 Баєсові мережі

Баєсова мережа є імовірнісною моделлю і є спрямованим ациклічним графом, в якому вузли є змінними (дискретними або неперервними), а ребра вказують на залежність між змінними. Приклад структури даної мережі зображено на рисунку 1.7. Баєсові мережі мають три основні особливості. Вони передбачають латентні змінні, а також параметричне та структурне навчання. У поєднанні з деревами рішень баєсові мережі дають досить хороші результати у класифікації емоцій, зокрема в області фільмів. Баєсова мережа моделює взаємозв'язок між функціями в дуже загальному вигляді.

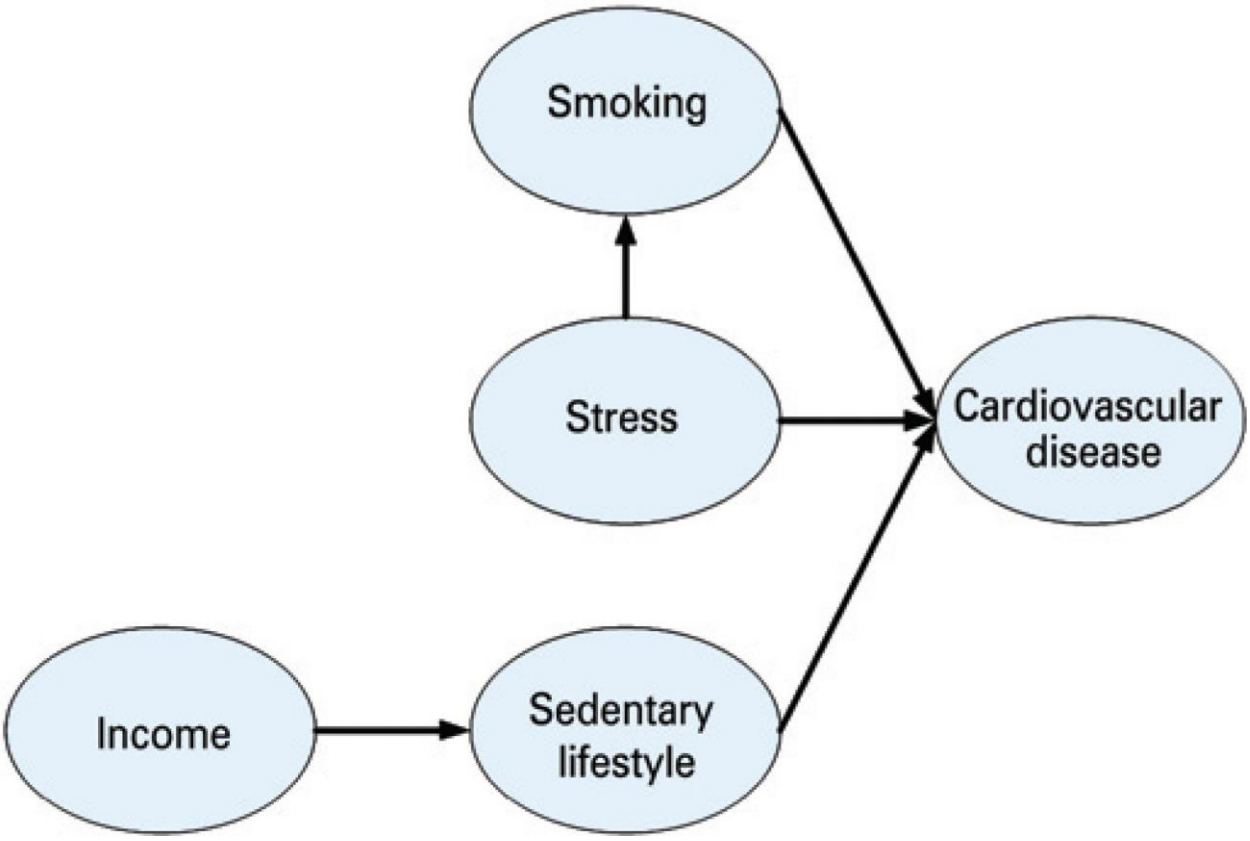


Рисунок 1.7 — Приклад структури баєсової мережі

Якщо зв'язки між змінними моделі добре відомі, або ж є достатнью даних для їх виявлення, то може бути доцільним використовувати байєсівську мережу. Наївний баєсів класифікатор - це проста модель, яка являє собою певний клас баєсової мережі, де всі змінні є умовно незалежними. Через це існує клас задач, які наївний баєсів класифікатор не може вирішити.

1.2.1.5 Нейронні мережі

Нейронні мережі - це сукупність природних або штучних нейронів, які використовуються для математичного та обчислювального аналізу моделей (рисунок 1.8). Нейромережі мають можливість адаптуватися до перемінних вхідних даних, тому мережа виробляє найкращий результат без необхідності переробки вихідних критеріїв. Сьогодні нейронні мережі активно продовжують набирати популярність у вирішенні дуже шикорого спектру

задач, зокрема такі, як гра в шахи або го на професіональному рівні, генерація людського голосу або зображень, а також у різноманітних задачах класифікації, таких як класифікація тональності тексту.

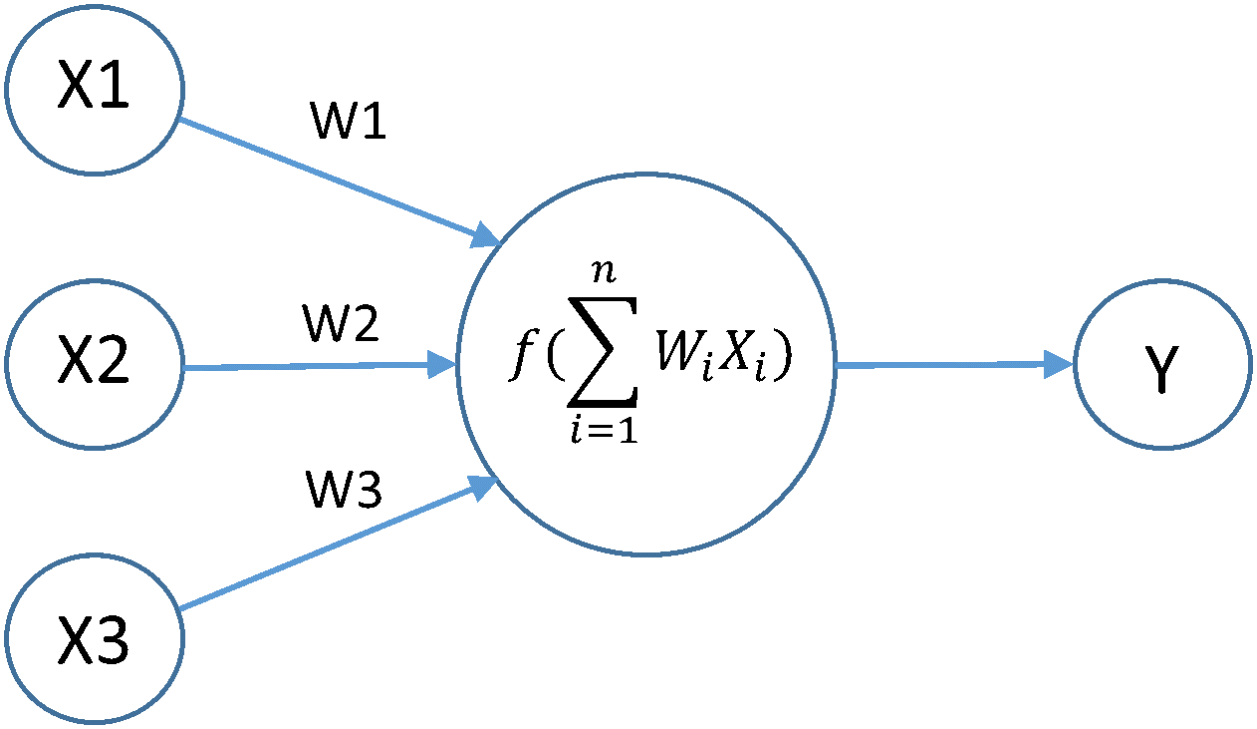


Рисунок 1.8 — Математична модель перцептрона, однієї з перших моделей нейронних мереж

Найбільш поширені алгоритми в нейронних мережах це мережі зворотнього поширення помилки (відомі також як багатошарові перцептрони (рисунок 1.9)) та мережі радіальних базисних функцій. Обидва класи нейронних мереж можуть будувати моделі, пов'язані як з розпізнванням образів, так і з класифікацією довільних об'єктів. Крім того, входи (і виходи) можуть мати будь-які дійсні значення.

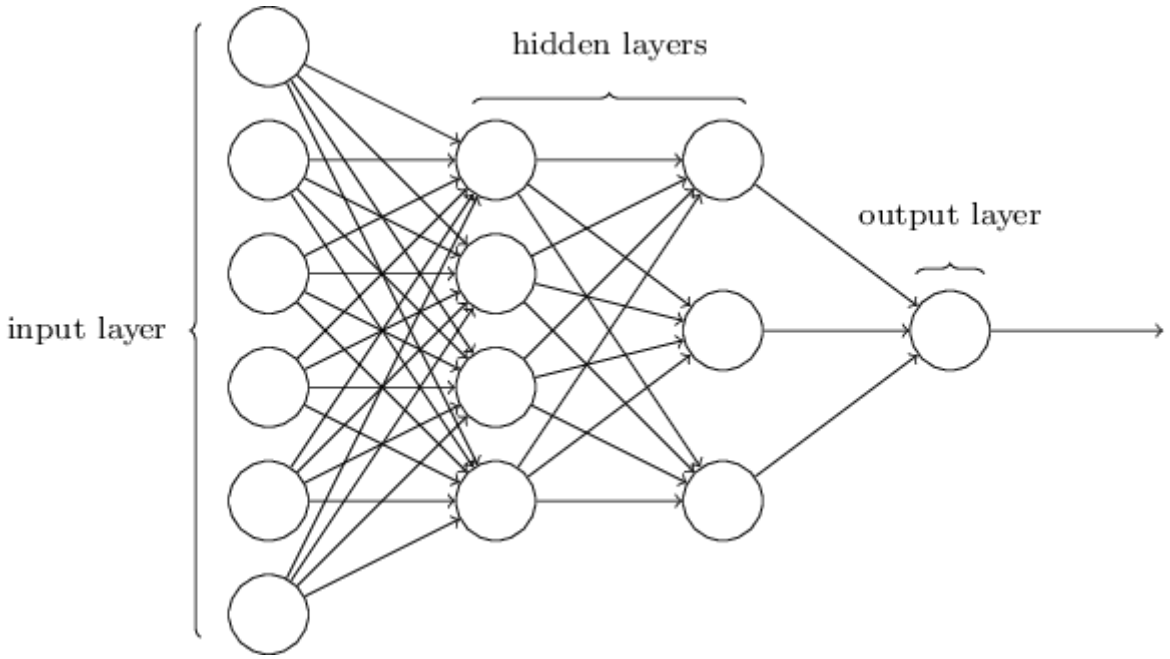


Рисунок 1.9 – структура багатошарового перцептрону

Прикладом реалізації сентиментального аналізу, використовуючи нейронні мережі, може слугувати проект CS291K, головна сторінка якого зображена на рисунку 1.10. Даний проект використовує пости у соціальній мережі Twitter у якості даних для навчання.

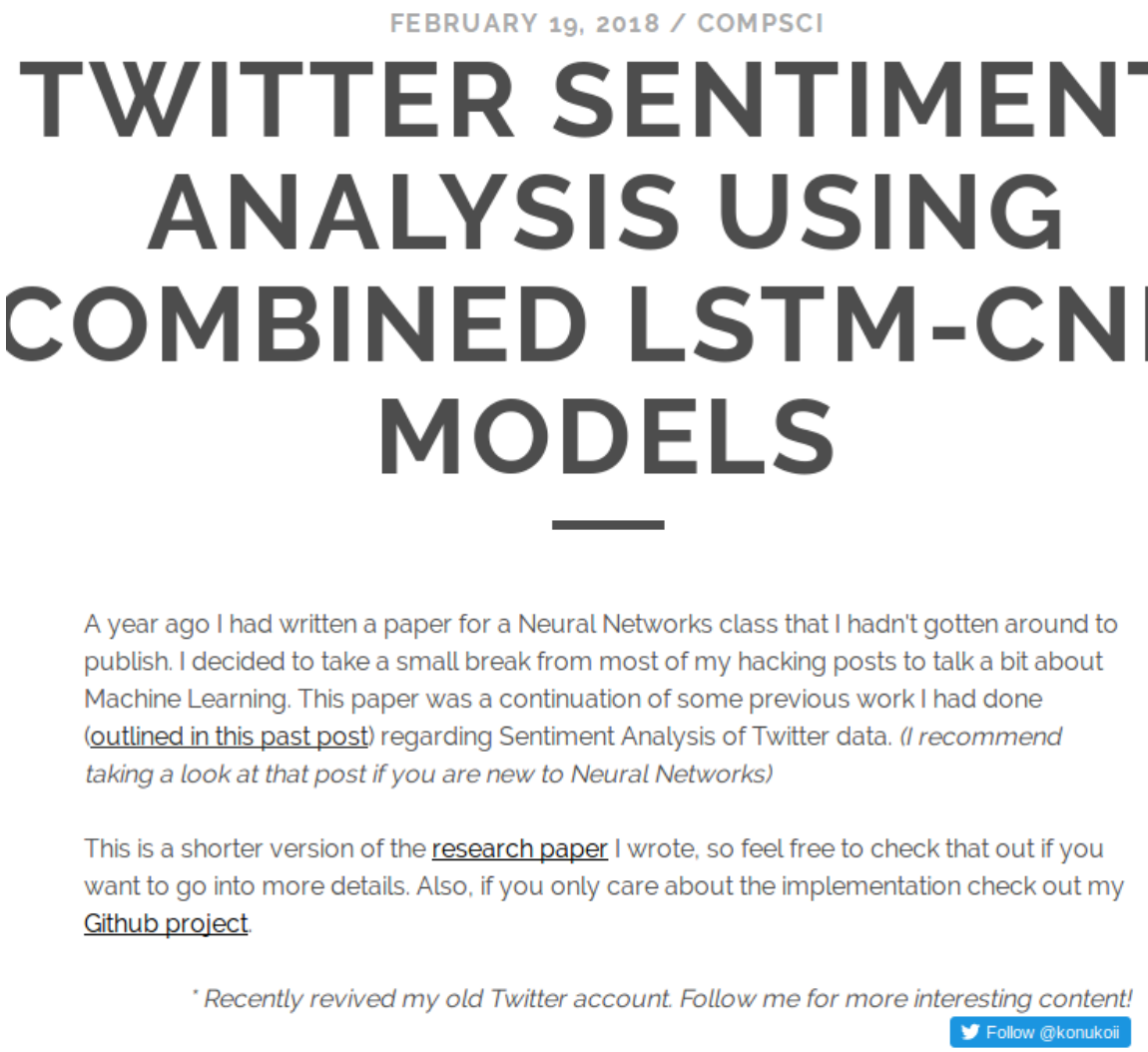


Рисунок 1.10 – Головна сторінка проекту CS291K [7]

1.2.2 Алгоритми навчання без учителя.

Машинне навчання без учителя має таку назву, тому що не потребує навчальних та тренувальних даних для побудови моделі. Також ці алгоритми у контексті сентиметнального аналізу можуть називатися семантично орієнтованими. Основними методами навчання без учителя, що використовуються у сентиментальному аналізі, є методи, що базуються на словниках, а також корпусна лінгвістика.

1.2.2.1 Методи, що базуються на словниках

Підхід, оснований на словниках (рисунок 1.11) - це метод, в якому слова у тексті послідовно оцінюються як позитивні або негативні, з використанням попереднью створеного словника співвідношень слів та їхніх оцінок.

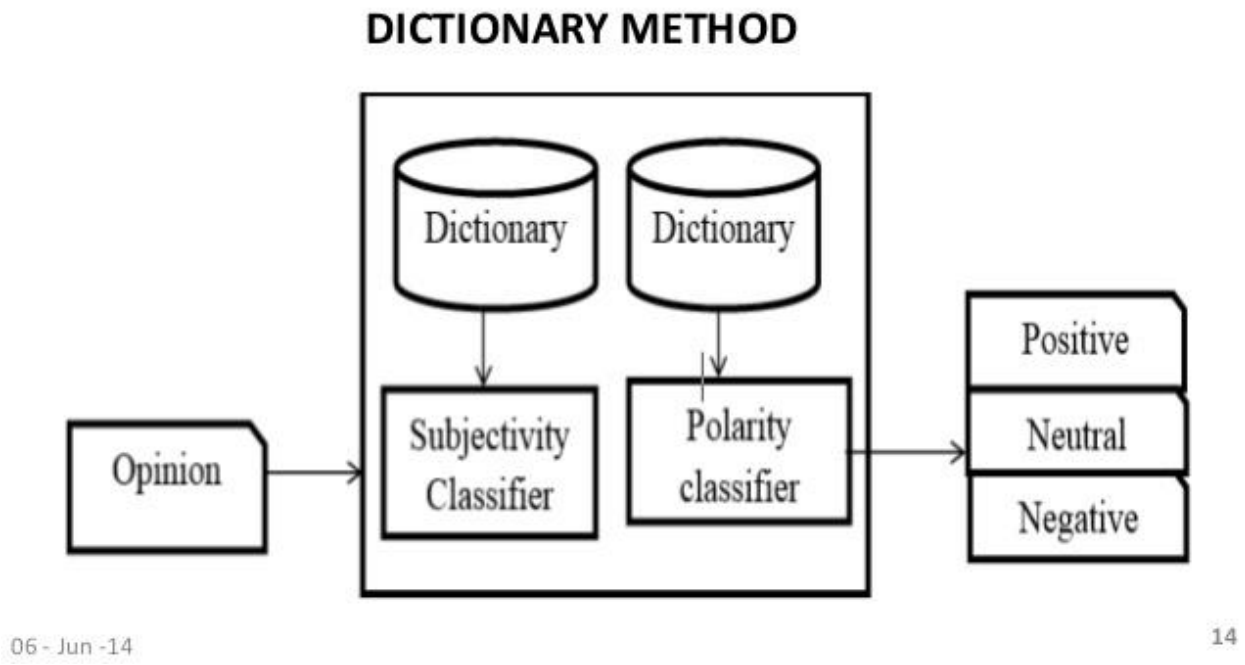


Рисунок 1.11 – Схема роботи моделі, побудованої з підходом, заснованим на словниках

Цей метод має ряд обмежень. Для того, щоб словникові методи добре працювали, оцінки, додані до слів, повинні тісно збігатися з тим, як ці слова використовуються в певному контексті. Якщо словник розроблено для певної програми, то це припущення повинно бути легко виправдати. Але коли словники створюються в одній предметній області, а потім застосовуються до інших проблем, можуть виникати серйозні помилки.

На рисунку 1.12 приведено приклад реалізації методу, заснованого на словниках.

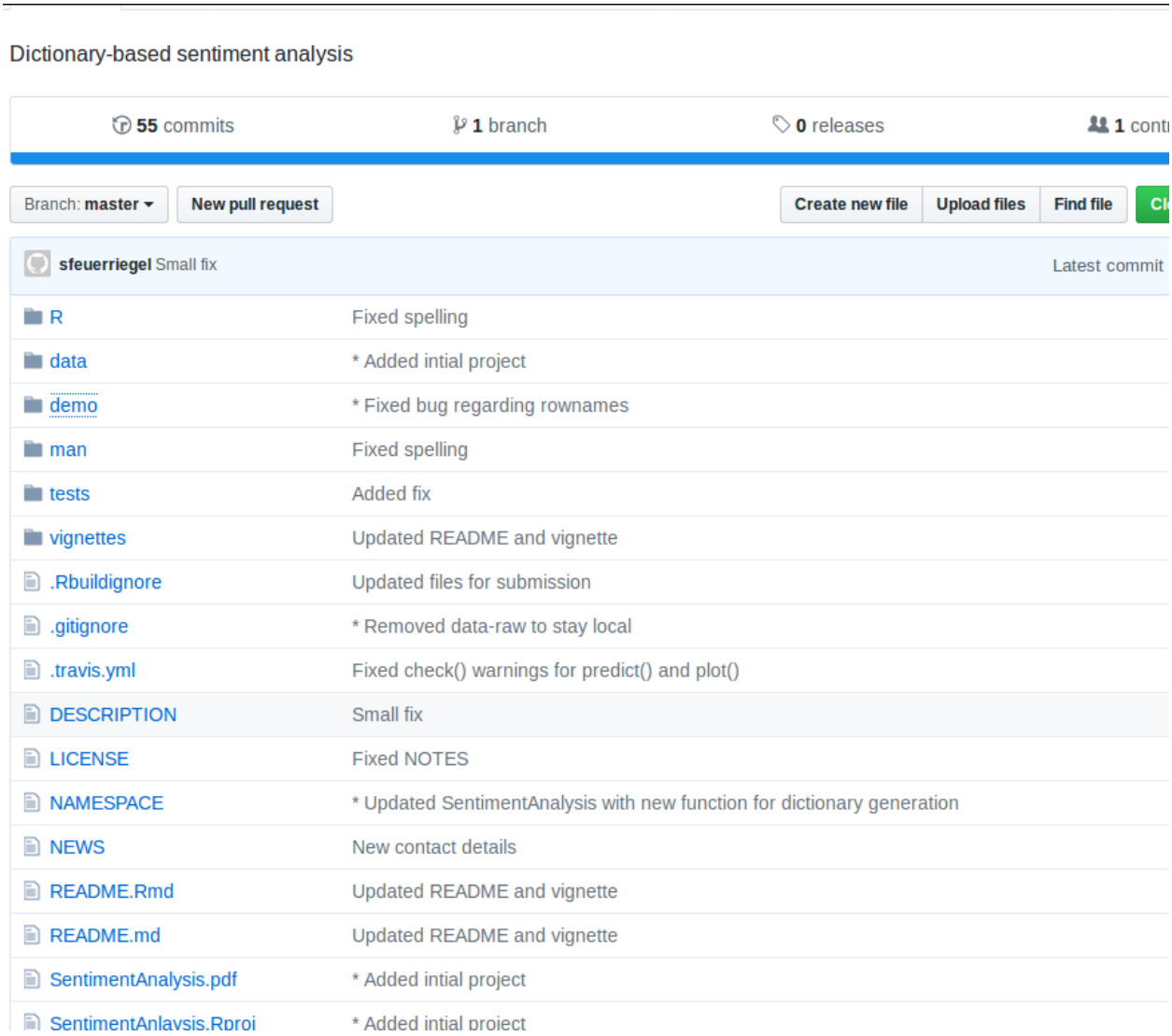


Рисунок 1.12 – Головна сторінка реалізації сентиментального аналізу, використовуючи метод, заснований на словниках [8]

1.2.2.2 Корпусна лінгвістика

Корпусна лінгвістика (рисунок 1.13), на відміну від метода, заснованого на словниках, оцінює близькість текста до груп текстів, об’єднаних у певний клас. Даний підхід добре себе проявив у сентиментальній класифікації як текстових, так і голосових даних.

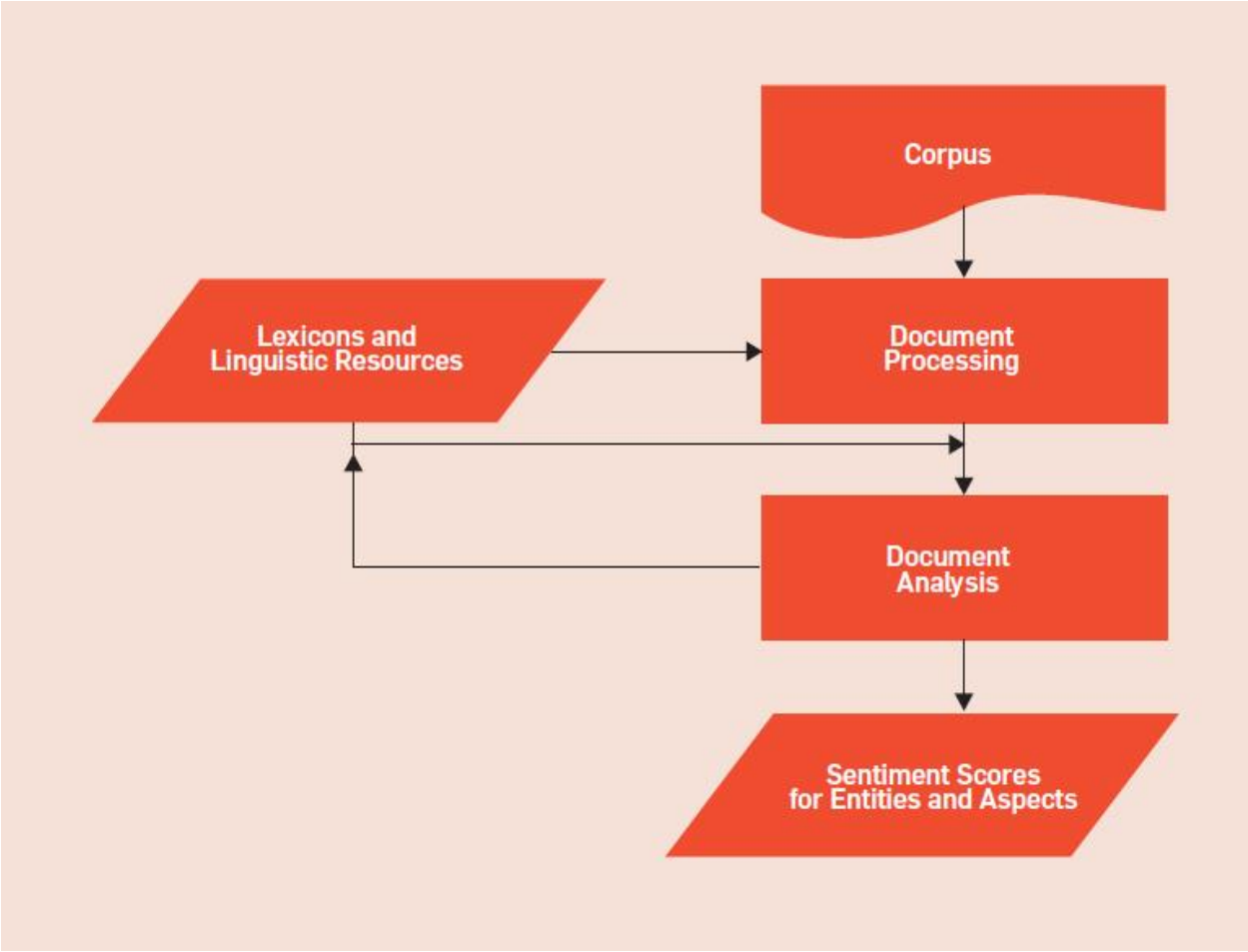


Рисунок 1.13 – Алгоритм сентиментального аналізу тексту методом корпусної лінгвістики

Вище були розглянуті основні методи машинного навчання, що використовуються для сентиментального аналізу. Нижче приведені основні переваги й недоліки машинного навчання з учителем.

Основні переваги навчання з учителем:

– модель може бути побудована таким чином, щоб отримати дуже специфічну інформацію про досліджуваний об'єкт, що дозволить досягти великої точності у рішенні поставленої задачі;
– програміст може самостійно визначити кількість класів, що він хоче мати у моделі;
– після завершення тренування моделі не обов'язково зберігати приклади тренувань у пам'яті. Отримана модель може бути збережена у вигляді

готової математичної формули, і цього буде достатньо для класифікації майбутніх вхідних даних.

Недоліки навчання з учителем:

- побудована модель може бути перенавчена. Це означає, що якщо навчальна вибірка не включає деякі приклади, які можуть бути в класі, тоу разі класифікації цих прикладів після тренування, вони можуть бути неправильно класифіковані;
- якщо є вхідні дані, які не можуть бути співставлені із жодним з існуючих класів, ці дані також можуть бути неправильно класифіковані;
- для побудови моделі потрібно мати великий обсяг розмічених прикладів з кожного класу під час навчання класифікатора. Якщо розглядати класифікацію великих даних, що може бути проблемою;
- тренування моделі може займати багато часу, що може бути особливо помітно у разі швидкої зміни даних.

Також можна виділити основні недоліки та переваги машинного навчання без учителя.

Переваги машинного навчання без учителя:

- відсутність необхідності витрачати час та ресурси на тренування моделі;
- відсутність необхідності у попередньо підготовлених даних про об’єкти класифікації.

Недоліки:

- менша точність моделювання у порівнянні з методами машинного навчання з учителем;
- відсутність фази тренування веде за собою необхідність кожен раз проводити усі розрахунки.

У таблиці 1 наведена порівняльна характеристика методів машинного навчання з учителем та без учителя для сентиментального аналізу.

Таблиця 1 – Порівняння методів машинного навчання для сентиментального аналізу тексту

Розділ машинного навчання	Машинне навчання з учителем	Машинне навчання без учителя
Необхідність розмічених даних про об’єкти	Так	Ні
Необхідність тренування моделі	Так	Ні
Точність моделювання	Висока	Висока, проте зазвичай менша за точність моделювання методами машинного навчання з учителем
Коли краще використовувати	За наявності підготовлених даних або можливості їх підготувати	За відсутності підготовлених даних та можливості їх підготувати

1.3 Огляд різновидів сентиментальної класифікації

Можна виділити різновиди сентиментальної класифікації за наступними критеріями:

- за використаним алгоритмом класифікації (розглянуто у розділі 1.1);
- за наявністю ієрархічності;
- за рівнем класифікації (документ, речення, властивість);

Вибір алгоритма сентиментальної класифікації розглянуто у розділі 1.1. Інші критерії більш детально розглянуто нижче.

1.3.1 Наявність ієрархічності

Ієрархічна класифікація застосовується у тому випадку, коли деякі класи об’єктів можуть включати до себе інші класи. Приклад структури класів у сентиментальному аналізі наведено на рисунку 1.14.

Рисунок 1.14 – Ієрархічна класифікація тональності тексту

Плоска класифікація – це класифікація у якій ієрархія відсутня, тобто немає класів, що включали б до себе інші класи. Приклад класів плоскої класифікації при оцінці тональності тексту наведено на рисунку 1.15.

Рисунок 1.15 – Плоска класифікація тональності тексту

1.3.2 Рівень класифікації

Аналіз тональності рівня документу класифікує весь поданий на вхід документ як єдине ціле. Цей рівень, наприклад, може класифікувати тональність документу як позитивну, негативну або нейтральну.

Аналіз тональності рівня речення визначає, чи кожне речення виражає позитивну, негативну чи нейтральну думку щодо товару чи послуги. Даний тип сентиментальної класифікації використовується для відгуків та коментарів, які містять одне речення і написані користувачем, або ж у разі, коли є наобхідність відокремити одне речення від іншого.

Аналіз тональності рівня властивості використовується у тому випадку, коли у поданному тексті необхідно виокремити відношення користувача до тих чи інших властивостей товару або послуги. Наприклад, користувач, описуючи свої враження від фільму, може окремо оцінювати його сценарій, гру актерів та спецефекти.

Порівнюючи розглянуті у розділі 1.2 різновиди сентиментальної класифікації не можна сказати, що якісь з них краще за інші для будь-яких задач моделювання. Навпаки, кожен з методів може виявитися кращим за інші у вирішенні окремої проблеми. Тож вибір різновиду класифікації та його адекватність залежить від розробника програми та його спосібності правильно оцінити проблему, що необхідно вирішити.

1.4 Огляд мов та технологій програмування

Далеко не всі мови програмування добре підходять для розробки программ, що реалізують методи машинного навчання. Адже деякі з мов програмування мають велику кількість бібліотек, модулів та допоміжних засобів, що дуже спрощують розробку.

На рисунку 1.16 наведено статистику популярності мов програмування серед роботодавців.

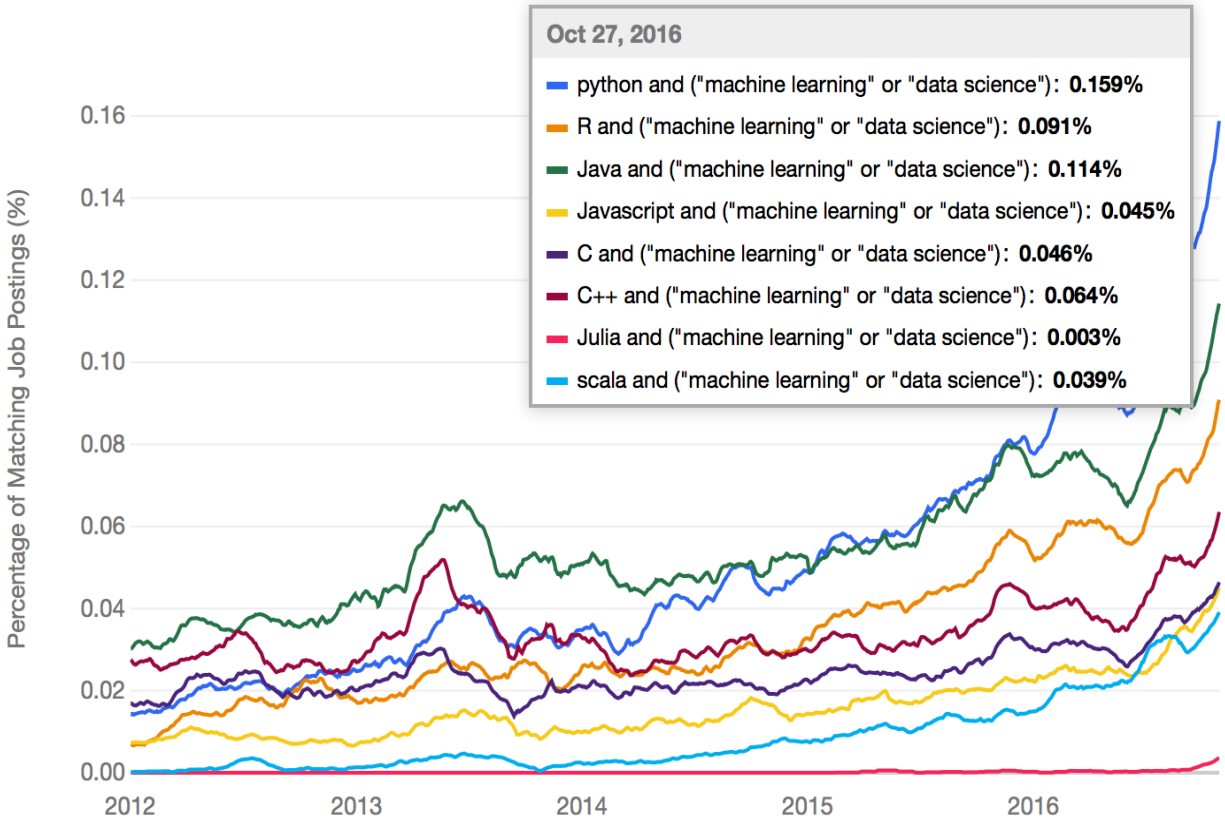


Рисунок 1.16 – Найбільш популярні мови програмування у сфері машинного навчання [9]

Також до розгляду мов та засобів програмування, що використовуються у машинному навчанні можна додати середовища, що не так популярні серед роботодавців, але використовуються у академічній сфері. Це такі середовища, як Matlab, LabVIEW. Також, сьогодні важливе місце серед засобів розробки програм у сфері машинного навчання займає Tensorflow – розроблена компанією Google бібліотека для вирішення задач машинного навчання, що містить у собі реалізацію багатьох низькорівневих проблем й дозволяє розробнику зконцентруватися безпосередньо на розробці алгоритму, а також надає можливість розробляти програми машинного навчання, використовуючи широкий вибір популярних мов програмування.

Нижче кожна з наведених мов та технологій розглянута більш детально.

1.4.1 Python

Python на сьогодняшній день є стандартом у сфері машинного навчання. Ця мова програмування має безліч бібліотек, що спрощують розробку программ у сфері штучного інтелекту. Python є повноцінною мовою програмування, і багато організацій використовують його у своїх виробничих системах. Мова Python це високорівнева мова програмування, що має простий синтаксис (рисунок 1.17) та низький поріг входження й вивчення, що дозволяє науковцям, що мало пов’язані з програмуванням, використовувати її у своїх дослідженнях. Дана мова програмування через свою високорівневість є досить повільнюю, однак усі найбільш затратні операції машинного навчання, такі як алгоритми навчання моделей, що використовуються у Python, реалізовані на низькорівневих мовах програмування, як правило, на мові програмування C. Тож, дана мова програмування поєднує у собі зручність роботи високорівневих мов програмування, швидкість низькорівневих, а також має за плечима десятки років розробок бібліотек для реалізації алгоритмів машинного навчання.

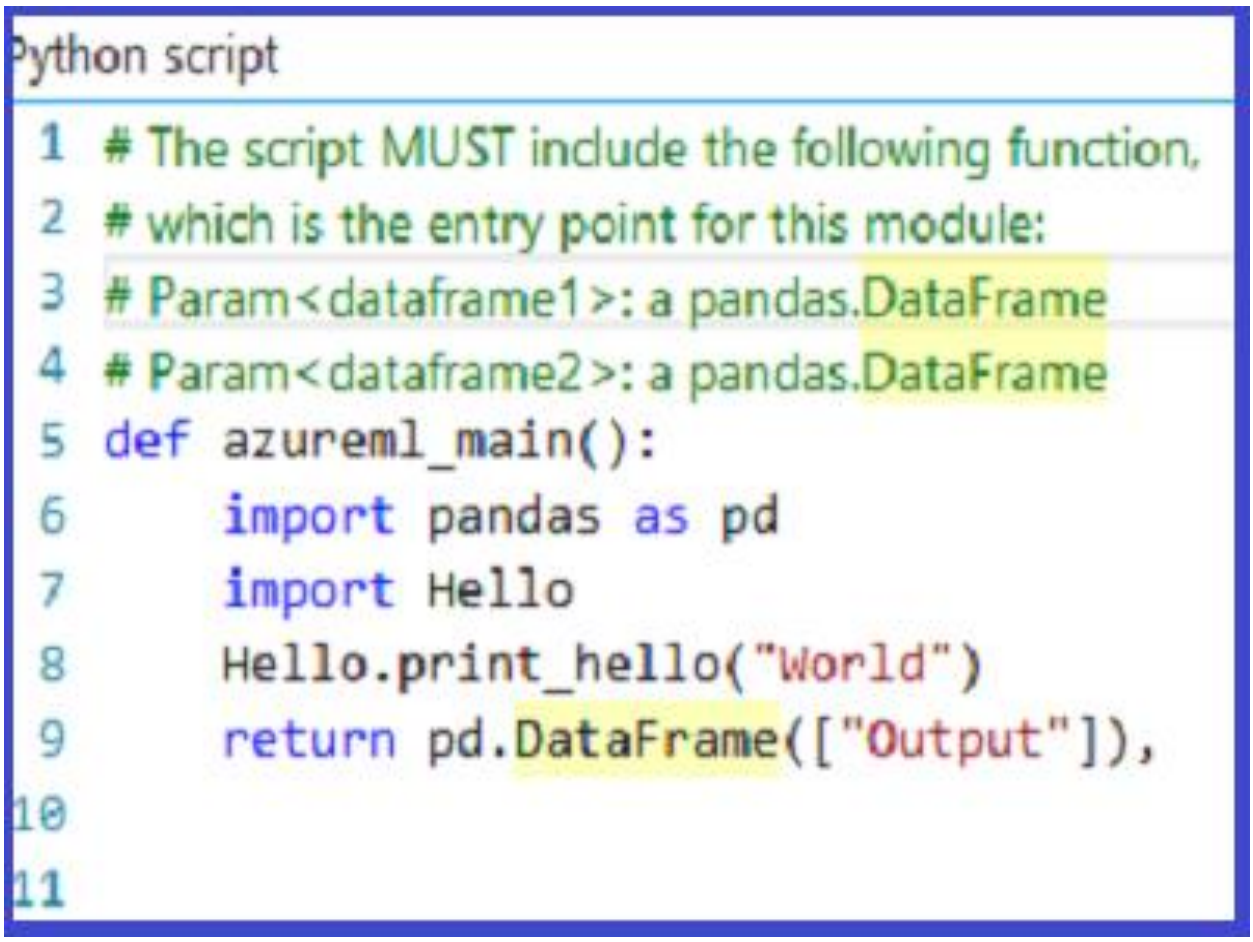


Рисунок 1.17 – Приклад програми, написаної мовою Python

Переваги мови програмування Python:

– безкоштовість;
– синтаксис програмування. Мову програмування Python було створено схожою на природню мову, яку можна легко читати, в той час як такі мови програмування, як Matlab, орієнтовані здебільшого на вирішення математичних проблем, а не на легкість читання коду;
– потужність. Python поставляється з великими стандартними бібліотеками і має потужні типи даних, такі як списки, масиви та словники;
– простори імен. Python працює з модулями, які потрібно імпортувати для використання. Використання просторів імен дає структуру програмі і залишає її чистою та зрозумілою. У Python все є об'єктом, тому кожен

об'єкт має власний простір імен. Це одна з причин, через яку код, написаний на мові Python, легко піддається відладці;

- відладка. Це випливає з об'єктно-орієнтованого характеру Python. Оскільки програма має чітку структуру, відлака проста. Приватні змінні існують лише умовно, тому розробник може отримати доступ до будь-якої частини програми, включаючи деякі внутрішні компоненти Python;
- портативність. Оскільки Python є цілком безкоштовним, код, написаний данною мовою програмування, може працювати скрізь. Крім того, він працює на операційних системах Windows, Linux і OS X;
- широкі інструменти GUI. З Python можна створити інтерфейс для програми, яка виглядає і працює добре. Програміст має вибрати будь-який з основних інструментів GUI, таких як Wx або Qt.

1.4.2 R

Мова програмування R (рисунок 1.18) є більш молодою, ніж Python, і на сьогодняшній день є популярною у задачах, пов’язаних зі статистикою. R віддають перевагу у академічному середовищі, і внаслідок зростання популярності R у наукових колах, а також за рахунок доступності бібліотек та відкритого вихідного коду, компанії також почали використовувати R.

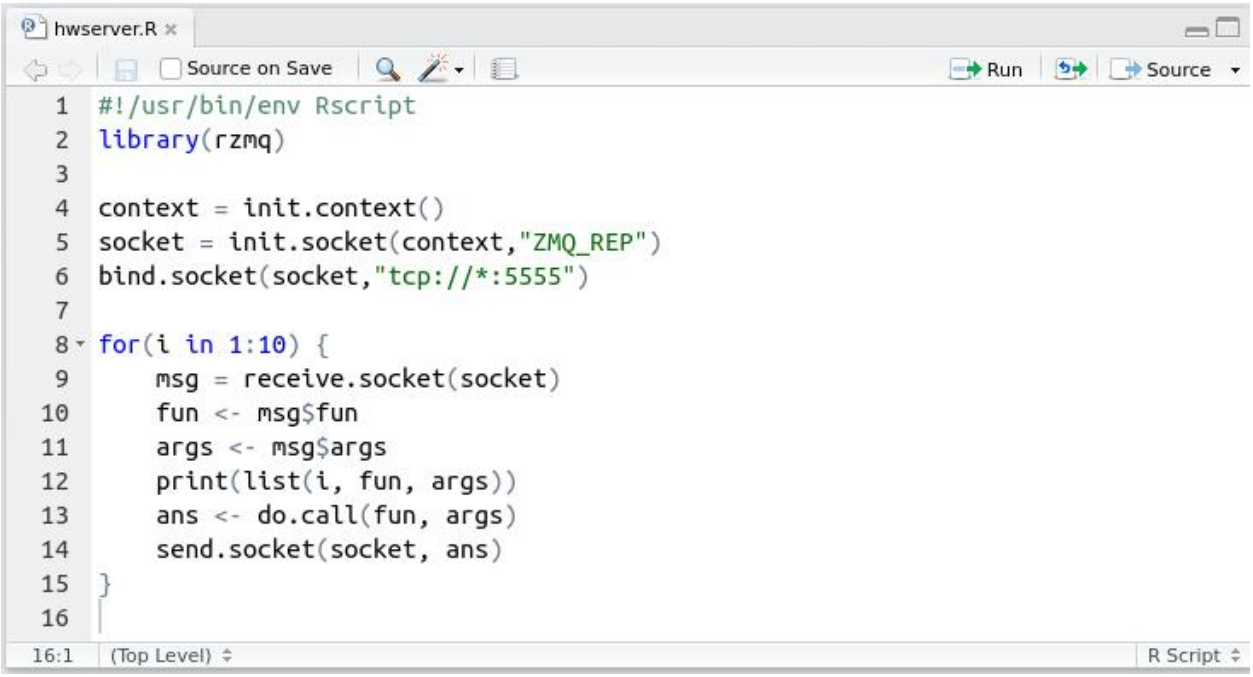


Рисунок 1.18 – Приклад програми, написаною мовою програмування R

R має деякі переваги над мовою Python, наприклад, його засобами набагато зручніше візуалізувати дані (рисунок 1.19). Однак, у порівнянні з Python це рішення все ще має такої ж кількості бібліотек та засобів для використання у комерційних проектах.

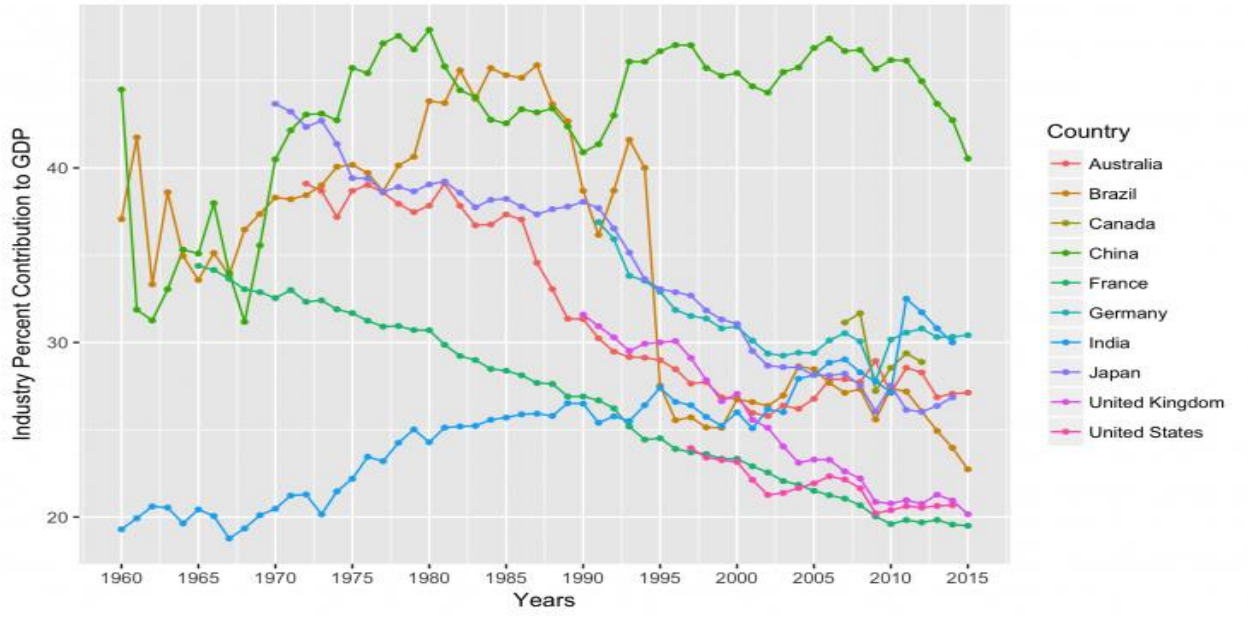


Рисунок 1.19 – Приклад візуалізації статистичних даних мовою програмування R

1.4.3 Matlab

Matlab – це середовище розробки, що надає користувачу разом з мовою програмування також багато інших корисних інструментів (рисунок 1.20).

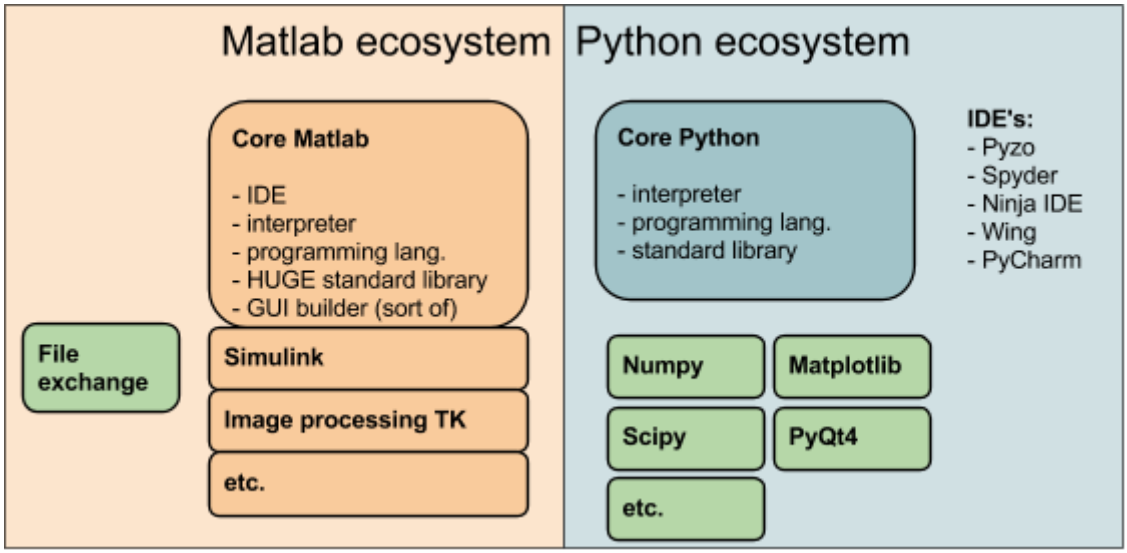


Рисунок 1.20 – Порівняння екосистем Python та Matlab [10]

Переваги середовища Matlab:

– має широкий вибір функцій для реалізації будь-яких алгоритмів;
– Simulink це продукт, для якого ще немає хорошої альтернативи;
– може бути простішим для початківців, оскільки пакет Matlab містить все необхідне, тоді як у Python потрібно встановити додаткові пакети та IDE (Pyzo намагається вирішити це питання);
– має велику наукову спільноту;
– використовується в багатьох університетах (хоча у невеликих компаній немає гроші на придбання ліцензії Matlab);

Водночас, дане середовище розробки має ряд недоліків:

– алгоритми є закритими, а це означає, що ви не бачите коду більшості алгоритмів, які ви використовуєте, і не можете перевірити, чи були вони правильно написані;
– Matlab досить дорогий, що означає, що код, написаний у Matlab, може використовуватись лише для людей, які мають достатньо коштів для придбання ліцензії;
– зазвичай Mathworks накладає обмеження на переносимість коду, тобто на здатність запускати код іншими людьми. Розробник може запустити свою "скомпільовану" програму, використовуючи Matlab Component Runtime (MCR), але прикладний додаток має точно відповідати версії встановленого MCR, що може бути незручно, оскільки Matlab випускає нову версію кожні 6 місяців;
– закритий вихідний код також не дає можливості стороннім розробникам розширити функціональність Matlab.

1.4.4 LabVIEW

LabVIEW – це середовище розробки, що досить схоже на Matlab, однак має порівняно меншу спільноту, а також менший набір бібліотек з готовою реалізацією тих чи інших алгоритмів, а також спрямований, порівняно з Matlab, в більшому степені на тестування. Зовнішній вигляд даного середовища розробки представлено на рисунку 1.21.

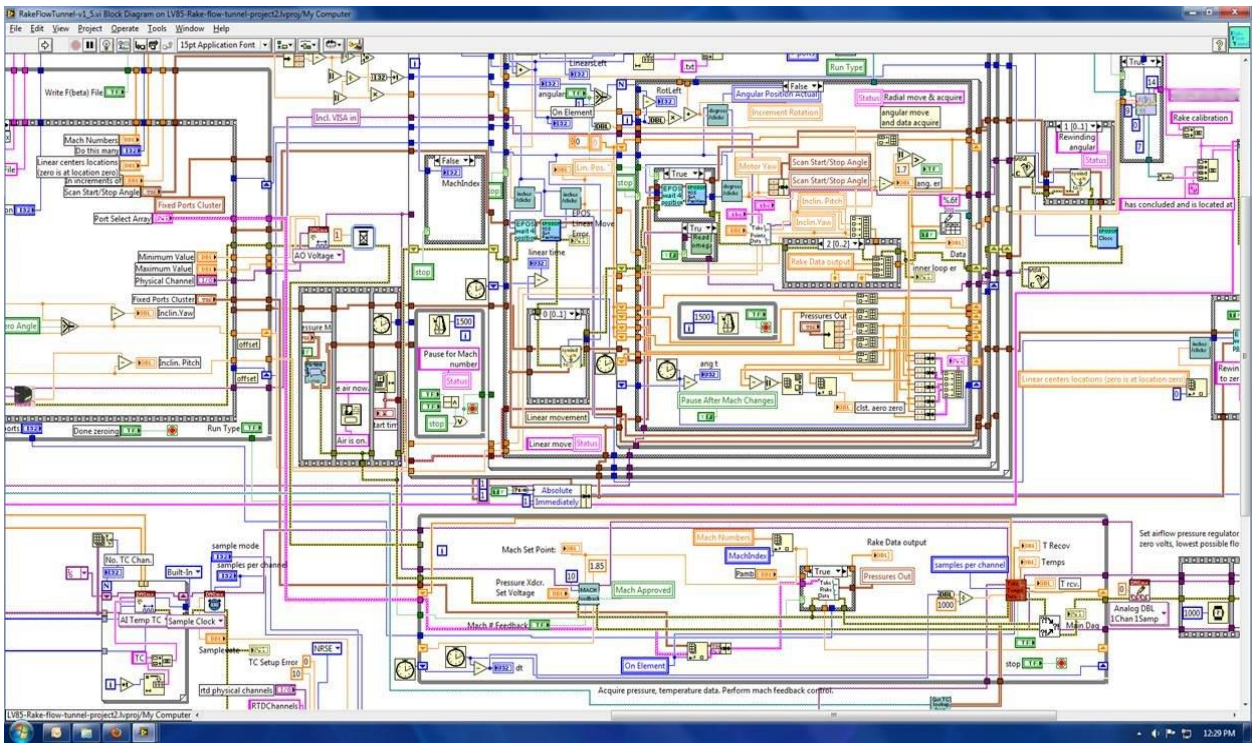


Рисунок 1.21 – Зовнішній вигляд програми LabVIEW

1.4.5 TensorFlow

TensorFlow - бібліотека програмного забезпечення з відкритим кодом для чисельного обчислення з використанням графів потоку даних. Вузли на графу являють собою математичні операції, а грані графа представляють собою багатовимірні масиви даних (тензиси), що передаються між ними. Гнучка архітектура дозволяє розгорнути обчислення для одного або декількох процесорів або графічних процесорів на настільному комп'ютері, сервері або мобільному пристрої за допомогою одного API. На рисунку 1.22 зображено приклад візуалізації потоку даних, використовуючи Tensorflow.

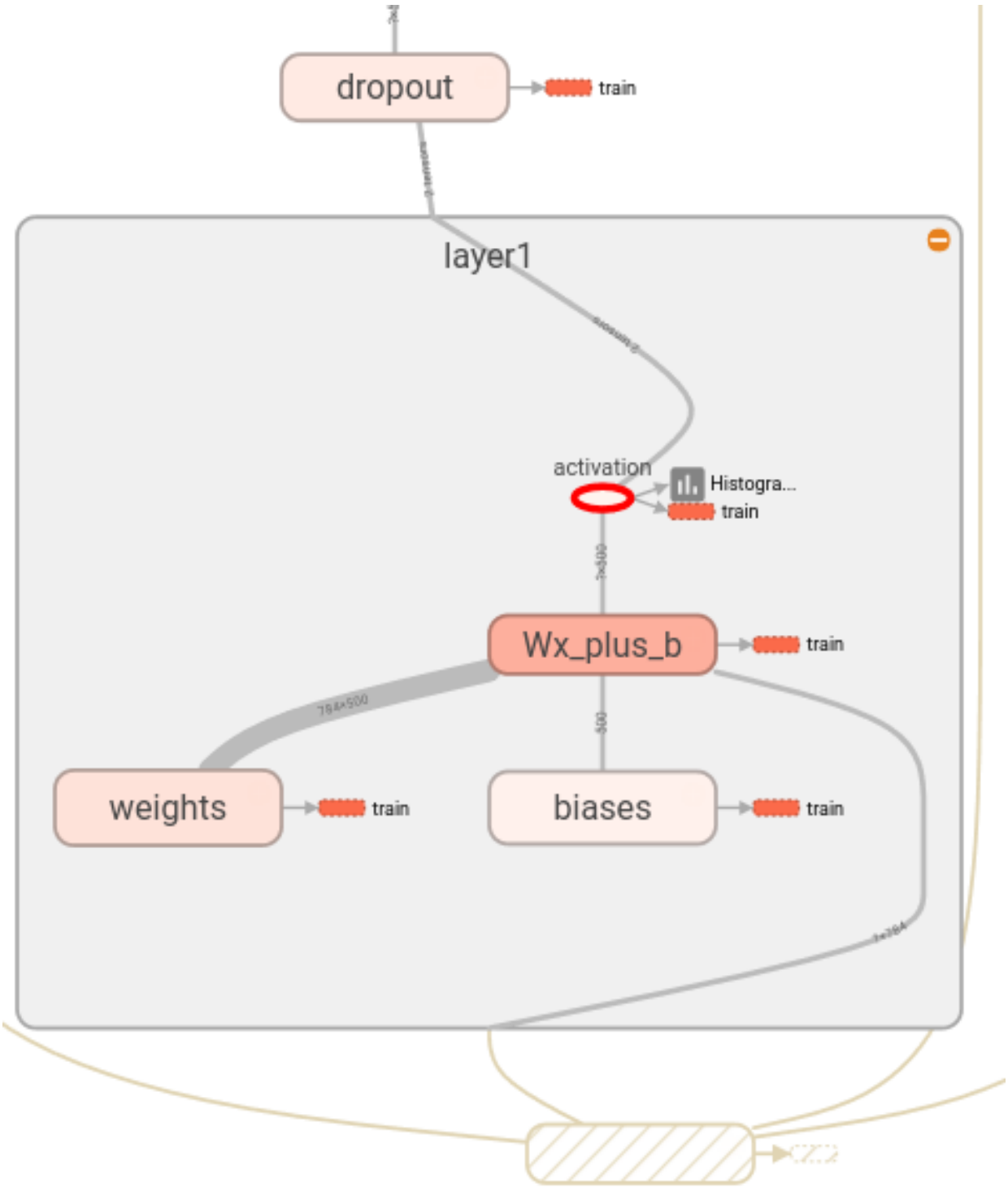


Рисунок 1.22 – Приклад візуалізації потоку даних, використовуючи Tensorflow

TensorFlow був спочатку розроблений дослідниками та інженерами, що працюють у групі Google Brain Team в дослідницькій організації Google

Machine Intelligence з метою проведення машинного навчання та дослідження глибинних нейронних мереж, однак ця система є достатньо загальною, щоб бути з успіхом застосованою у багатьох інших областях.

У таблиці 2 наведена порівняльна характеристика розглянутих вище технологій та мов програмування.

Таблиця 2 – Порівняння мов та технологій програмування для сентиментального аналізу

Мова програмування	Python	R	Matlab	LabVIEW	Tensorflow
Широкий технологій для сентиментального аналізу	Так	Так	Так	Ні	Так
Наявність великої спільноти	Так	Так	Так	Ні	Так
Відкритий вихідний код	Так	Так	Ні	Ні	Так
Наявність графічного	Ні	Ні	Так	Так	Ні

інтерфейсу					
Низький поріг вивчення	Так	Ні	Так	Так	Ні
Безкоштовність	Так	Так	Ні	Ні	Так
Проста переносимість коду	Так	Так	Ні	Ні	Так
Універсальність мови програмування	Ні	Ні	Ні	Ні	Так
Добре підходить до застосування у комерційних рішеннях	Так	Ні	Ні	Ні	Так

У даному розлілі були розглянуті та проаналізовані основні алгоритми машинного навчання, що можуть бути використаня для сентиментального аналізу тексту, а також основні види класифікації

тексту й мови та середовища, що краще підходять для написання програми. Можна зробити наступні висновки:

- оскільки мова програмуання Python має більше переваг, ніж інші, а фреймворк Tensorflow може ще більше спростити реалізацію програми й може бути використаний у поєднанні з мовою Python, для написання программного додатку до данного дипломного проекту добре підходить мова програмування Python у поєднанні з фреймворком Tensorflow;
- оскільки нейронні мережі дозволяють досягти гарних результатів у текстовій класифікації, а також отримали широку популярність в останні роки, вони добре підходять для написання программного додатку до данного дипломного проекту;
- оскільки стоїть задача класифікувати відгуки на два класи (позитивна або негативна тональність), то програмний додаток буде реалізовувати плоску класифікацію рівня документу.

2 ОГЛЯД АРХІТЕКТУР НЕЙРОННИХ МЕРЕЖ

Враховуючи результати, що досягаються за допомогою нейронних мереж у вирішенні різних задач машинного навчання (у тому числі в розв’язанні задач сентиментального аналізу), універсальність нейронних мереж, а також їхню популярність у останні роки та схожість деяких реалізацій на роботу біологічних систем, велику увагу у даному дипломному проекті приділено саме данному классу алгоритмів.

2.1 Детальний розгляд загальної архітектури штучних нейронних мереж

Штучна нейронна мережа (або англійською – artificial neural network, скорочено ANN) - це обчислювальна нелінійна модель, заснована на нейронній структурі мозку, яка здатна навчитися виконувати такі задачі, як класифікація, прогнозування, прийняття рішень, візуалізація та інші, просто тренуючись на прикладах.

Штучна нейронна мережа складається з штучних нейронів або елементів обробки, що як правило поділені на шари, що поєднані між собою послідовно (рисунок 2.1). Є три класи шарів нейронної мережі: вхідний шар, прихований шар, і вихідний шар. Нейронна мережа може мати більше одного прихованого шару, й у такому випадку процесс тренування даної нейронної мережі носить назву глибокого навчання (Deep Learning).

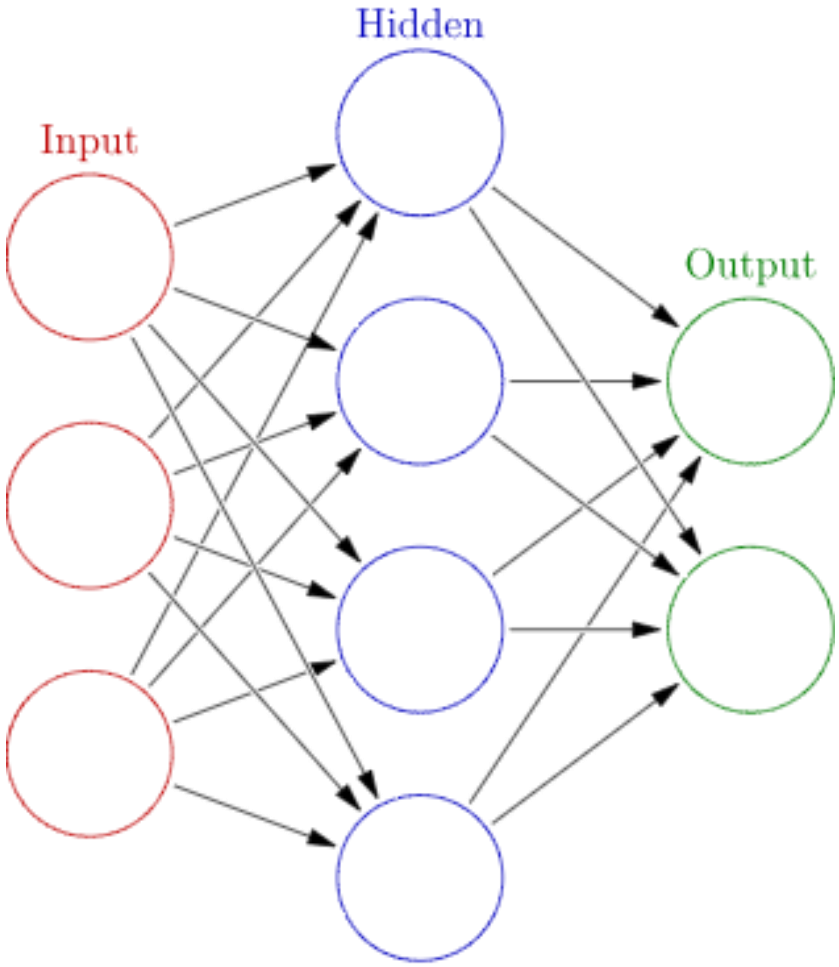


Рисунок 2.1 – Зображення загальної структури та типів шарів нейронної мережі [11]

Вхідний шар містить вхідні нейрони, які надсилають інформацію до прихованого шару. Прихований шар передає дані до наступного прихованого шару або на вихідний шар. Кожен нейрон має зважені входи (синапси), функцію активації (деяким чином перетворює вихід нейрона згідно заданих обмежень), і один вихід. Синапси - це регульовані параметри, які перетворюють нейронну мережу на параметризовану систему.

На рисунку 2.2 зображена нейронна мереж з чотирма вхідними нейронами.

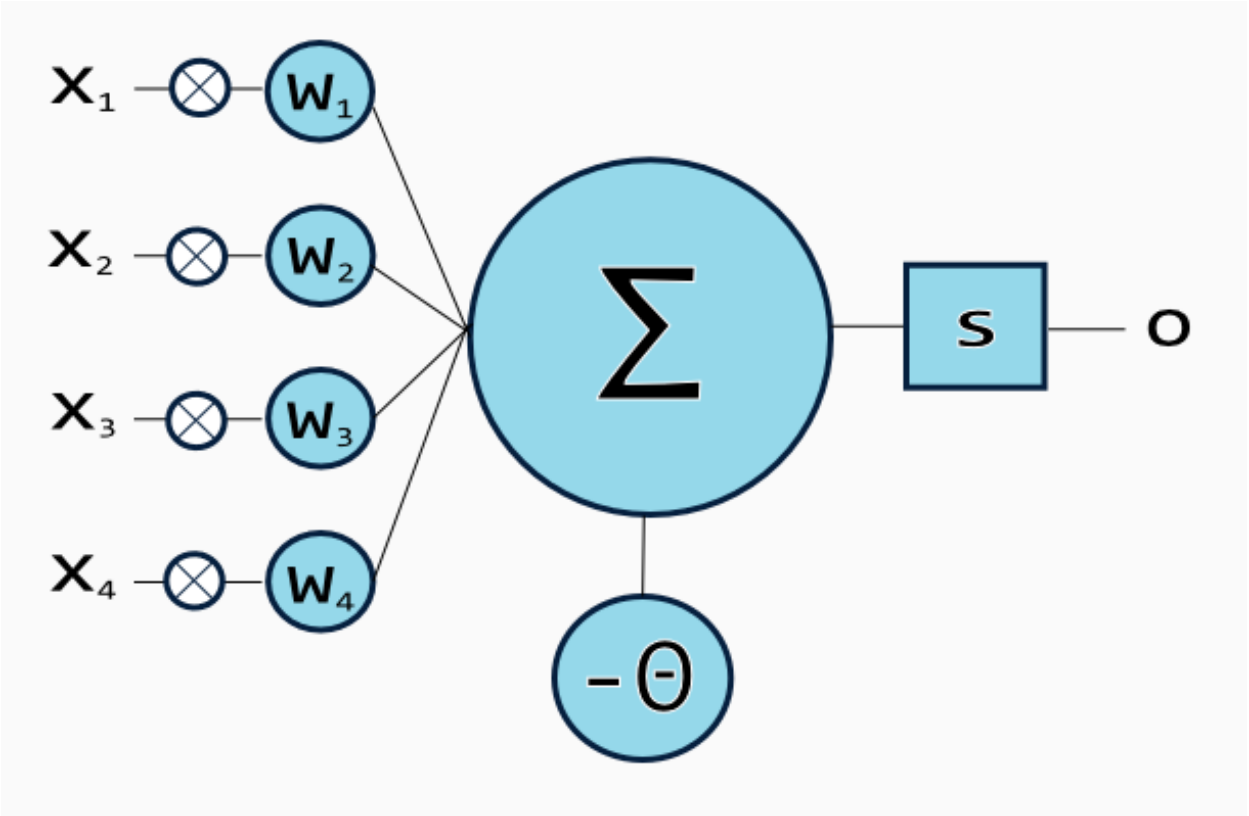


Рисунок 2.2 – Приклад нейронної мережі з чотирма вхідними нейронами [12]

Зважена сума входів виробляє сигнал активації, який передається в функцію активації для одержання одного виходу з нейрона. Загальноприйнятими функціями активації є лінійна, крокова, сигмоїдальна, тангенційна та лінійна випрямляюча (rectified linear unit, або ReLU).

На рисунку 2.3 зображено формулу та графік крокової функції активації.

$$f(x) = \begin{cases} 0 & \text{for} \quad x < 0 \\ 1 & \text{for} \quad x \geq 0 \end{cases}$$

Рисунок 2.3 – Графік та формула крокової функції активації

На рисунку 2.4 зображено формулу та графік сигмоїдної функції активації.

$$f(x) = \frac{1}{1 + e^{-x}}$$

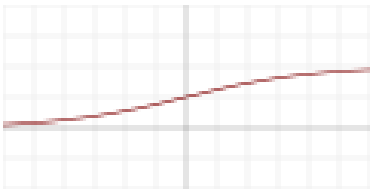


Рисунок 2.4 – Графік та формула сигмоїдальної функції активації

Сьогодні сигмоїда використовується рідше, ніж раніше. Ця функція має два серйозні недоліки:

– насичення сигмоид призводить до загасання градієнтів;

– вихід сигмоїди не центрований відносно нуля.

На рисунку 2.5 зображено формулу та графік тангенційної функції активації.

$$f(x) = \tanh(x) = \frac{2}{1 + e^{-2x}} - 1$$

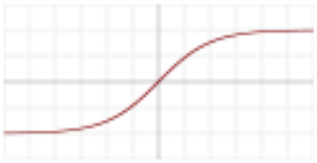


Рисунок 2.5 – Графік та формула тангенційної функції активації

Подібно до сигмоїди, гіперболічний тангенс може насичуватись. Однак, на відміну від сигмоїди, вихід даної функції центрований відносно нуля. Отже, на практиці завжди краще використовувати гіперболічний тангенс, а не сигмоїду.

На рисунку 2.6 зображено формулу та графік ReLU функції активації.

$$f(x) = \begin{cases} 0 & \text{for} \quad x < 0 \\ x & \text{for} \quad x \geq 0 \end{cases}$$

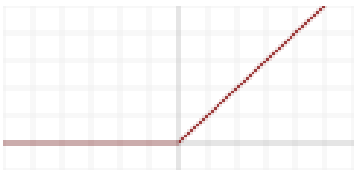


Рисунок 2.6 – Графік та формула ReLU функції активації

Розглянемо позитивні і негативні сторони ReLU.

Позитивні:

– обчислення сигмоїди і гіперболічного тангенса вимагає виконання ресурсномістких операцій, таких як піднесення до степеня, в той час як ReLU може бути реалізований за допомогою простого порогового перетворення матриці активацій в нулі. Крім того, ReLU не схильний до насичення;
– застосування ReLU істотно підвищує швидкість збіжності стохастичного градієнтного спуску (в деяких випадках до 6 разів [13]).

Негативна сторона - ReLU не завжди достатньо надійні і в процесі навчання можуть виходити з ладу.

Нижче приведено формулу softmax функції активації (формула 2.1).

$$\sigma(\mathbf{z})_j = \frac{e^{z_j}}{\sum_{k=1}^{K} e^{z_k}} \quad \text{for } j = 1, \ldots, K. \tag{2.1}$$

Softmax функція активації має наступну особливість: сума усіх виходів нейронів шару після застосування даної функції буде дорівнювати одиниці. Тому ця фунція активації часто застосовується на вихідному шарі при класифікації. Це дозволяє отримати на виході для кожного класу імовірність приналежності об’єкта до нього.

Загалом, ефективність використання тієї чи іншої функції активації шару залежить від задачі моделючання та розміщення шару у структурі нейронної мережі.

Тренування – це процес оптимізації вільних коефіцієнтів нейронної мережі (ваг), в якому мінімізується прогностична помилка, і нейронна мережа досягає заданого рівня точності. Метод, який в основному використовується для визначення вагів кожного нейрона, називається методом зворотного пошерення помилки. Даний метод мінімізує помилку моделі шляхом обчислення градієнту функції втрат. Це досягається шляхом ітеративної зміни

значень функції, рухаючись у напрямку, протилежному до градієнта функції втрат (рисунок 2.7).

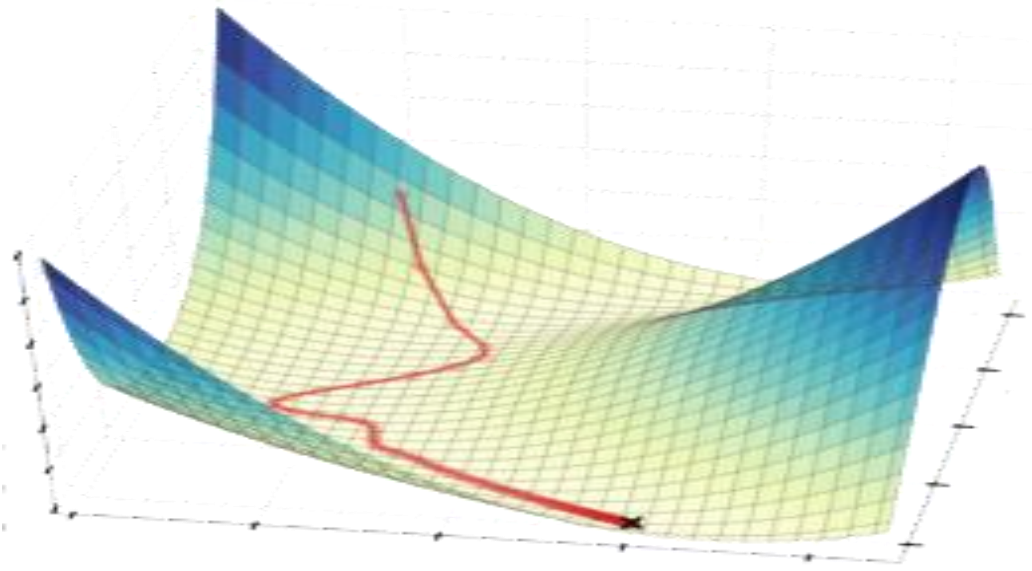


Рисунок 2.7 – Ілюстрація ітеративного зменшення функції втрат шляхом методу градієнтного спуску

Нижче приведено формули (2.2 – 2.5), що пов’язують зменшення значення функції на кожній наступній ітерації з градієнтом даної функції. Оскільки часткова похідна функції втрат буде завжди мати від’ємне значення, можна гарантувати зменшення функції втрат.

$$\Delta C \approx \nabla C \cdot \Delta v. \tag{2.2}$$

$$\Delta v = -\eta \nabla C, \tag{2.3}$$

$$v \to v' = v - \eta \nabla C. \tag{2.4}$$

$$\Delta C \approx -\eta \nabla C \cdot \nabla C = -\eta \|\nabla C\|^2 \quad 2.5$$

2.2 Детальний огляд архітектур нейронних мереж

Нижче розглянуті основні типии архітектур нейронних мереж, що можуть бути використані для сентиментального аналізу та визначено їхню придатність до вирішення даної проблеми.

2.2.1 Багатошаровий перцептрон

Багатошаровий персептрон (рисунок 2.8) може мати три або більше шарів. Він використовує нелінійну функцію активації (переважно гіперболічну дотичну або сигмоїдальну функцію), яка дозволяє класифікувати дані, які не є лінійними. Кожен нейрон в шарі з'єднується з кожним нейроном у наступному шарі, що робить мережу повнозв’язною. Застосовуватись, багатошаровий персептрон може, наприклад, для обробки природних мов: розпізнавання мовлення та машинного перекладу.

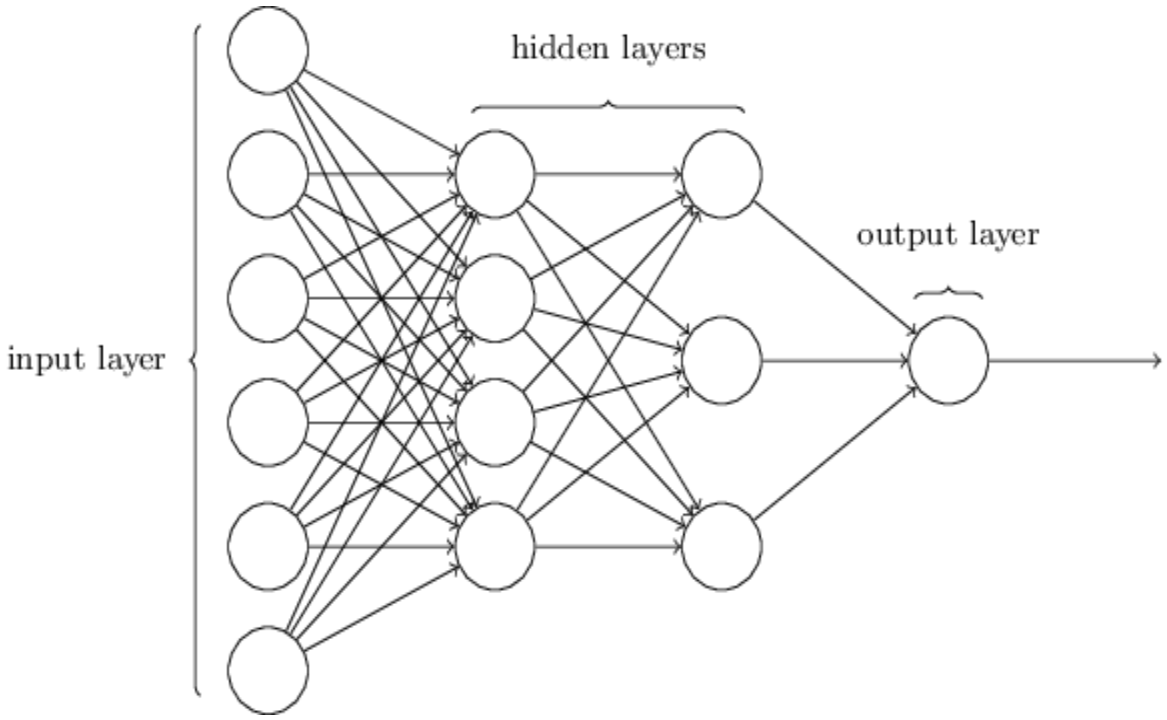


Рисунок 2.8 – структура багатошарового перцептрону [14]

Даний тип нейронних мереж може бути з успіхом застосовано для текстової класифікації [15].

2.2.2 Згорткова нейронна мережа

Згорткова нейронна мережа (Convolutional Neural network, або CNN) містить один або декілька згорткових шарів, частково або повністю з'вязаних, і є удосконаленною варіацією багатошарових персептронів, описаних вище. Структура згорткової мережі зображено на рисунку 2.9.

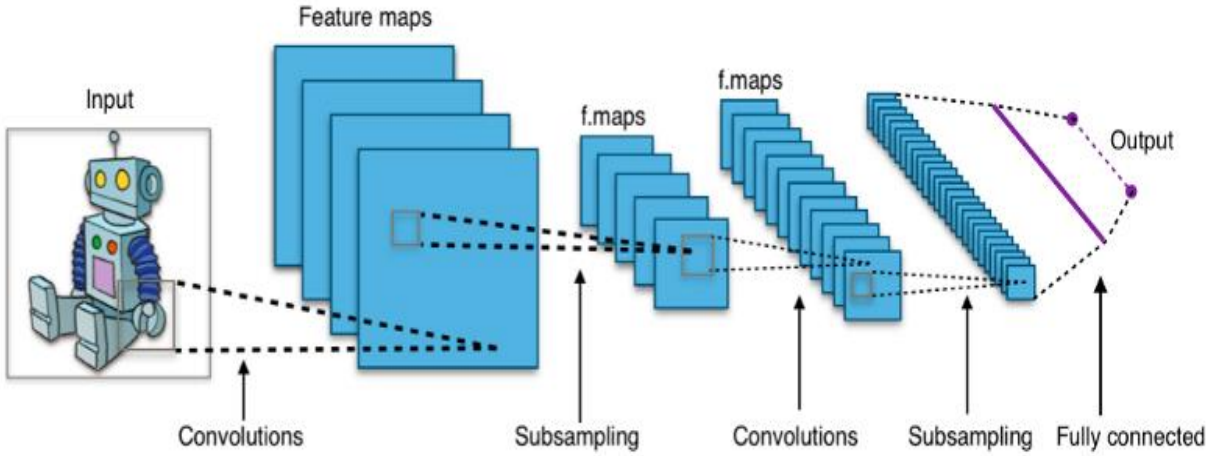


Рисунок 2.9 – Структура згортковоъ мережі

Згорткові шари застосовують операцію згортки до вхідних даних, що преедаються до наступного шару. Ця операція дозволяє нейнронній мережі ставати глибшою з набагато меншою кількістю параметрів.

Згорткові нейронні мережі демонструють видатні результати в програмах для обробки зображень та мовлення. Згорткові нейронні мережі можуть досягти видатної продуктивності без знання слів, фраз, речень та будь-яких інших синтаксичних чи семантичних структур стосовно людської мови.

Даний тип нейронних мереж добре підходить та широко використовується для задач класифікації зображень, а також для задач, пов’язаних з обробкою тексту [16].

2.2.3 Рекурсивна нейронна мережа

Рекурсивна нейронна мережа (РНН) - це тип глибокої нейронної мережі (нейронної мережі, що має більше одного прихованого шару), сформований шляхом рекурсивного застосування одного і того ж набору вагів у структурі, щоб зробити структуроване передбачення для вхідних даних змінної величини або скалярне передбачення на цих даних шляхом переміщення даної структури в топологічному порядку. У найпростішій архітектурі використовуються нелінійна функція активації, наприклад тангенційна, та

матриця вагів, яка є спільною для всієї мережі. Приклад найростішої архітектури рекурсивної нейронної мережі зображено на малюнку 2.10.

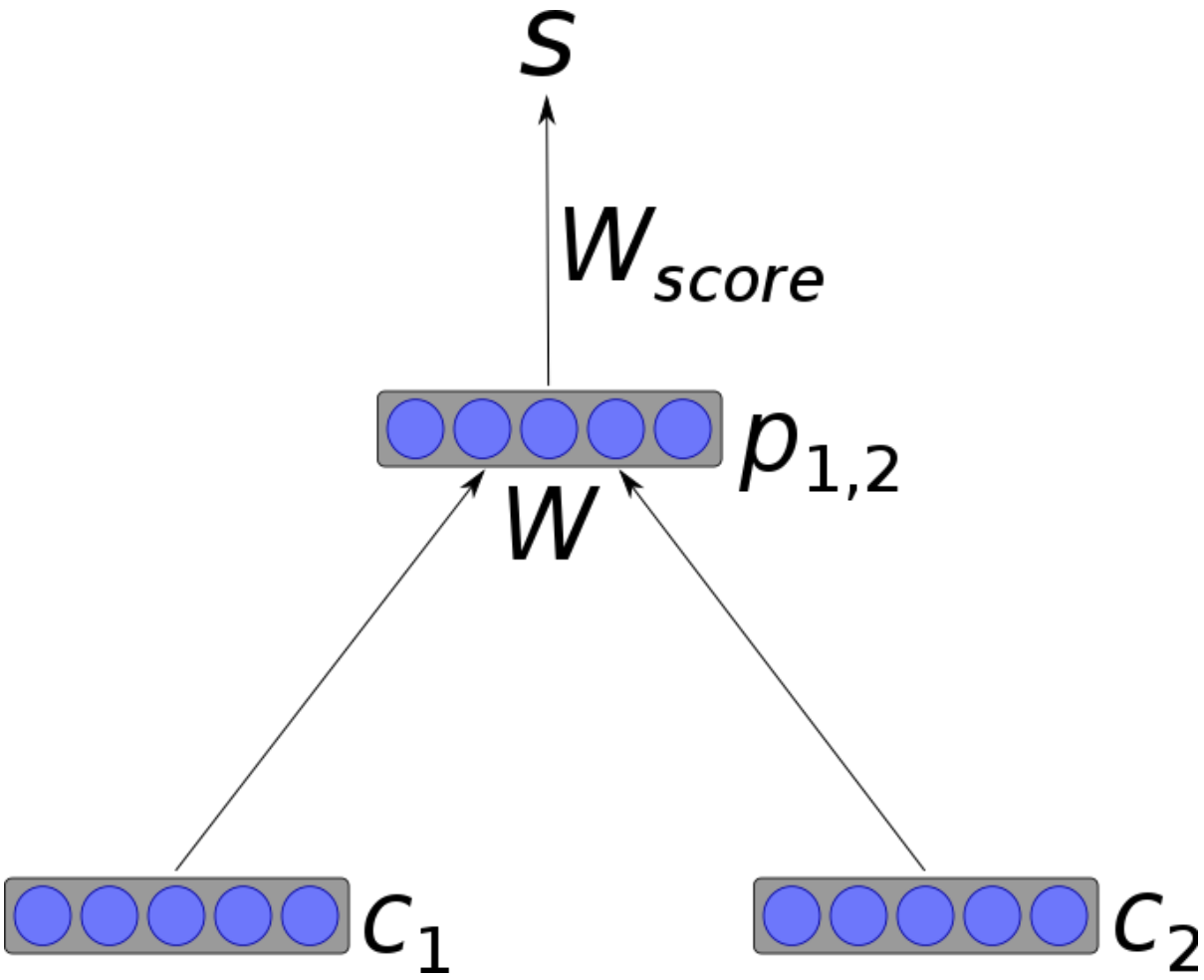


Рисунок 2.10 – Проста архітектура рекурсивнох нейронної мережі [17]

2.2.4 Рекурентна нейронна мережа

Рекурентна нейронна мережа (рисунок 2.10), на відміну від нейронних мереж прямого розвовсюдження, є варіантом рекурсивної штучної нейронної мережі, в якій зв'язки нейронів роблять спрямований цикл. Це означає, що вихід залежить не тільки від поточних входів, але і від стану нейронів на попередньому кроці. Ця пам'ять дозволяє користувачам покращити результати у вирішені таких проблем, як розпізнавання рукописного тексту або розпізнавання мовлення.

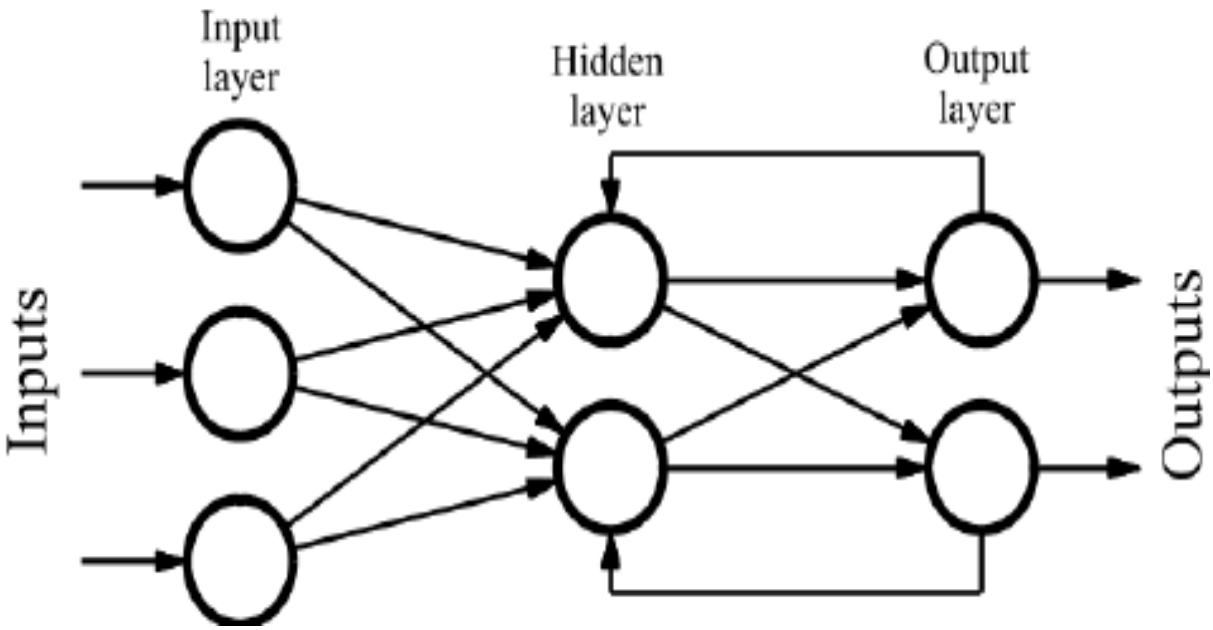


Рисунок 2.10 – Приклад архітектури рекурентної нейронної мережі

Даний тип нейронних мереж дозволяє досягти добрих результатів у вирішенні задач генерації текстів [18].

2.2.5 Нейронні мережі з довгою короткочасною пам’яттю

Довга короткострокова пам'ять - це специфічна рекурентна архітектура рекурсиних нейронних мереж, яка була розроблена для моделювання часових рядів більш точно, ніж звичайні рекурсивні нейронні мережі. Довга

короткострокова пам'ять не використовує функцію активації у своїх рекурентних компонентах, збережені значення не змінюються, і градієнт не схильний до зникнення під час тренувань. Зазвичай, нейрони у даній архітектурі поділені на блоки, що включають по декілька нейронів. Дані блоки містять три або чотири "вентиля", що слугують для упраління потоками даних на входах і виходах блоків даних. Довга короткострокова пам’ять забезпечує найсучаснішу продуктивність для масштабного акустичного моделювання.

Даний тип нейронних мереж добре проявляє себе у задачах акустичного моделювання [19]. На рисунку 2.11 зображено приклад архітектури нейронної мережі з довгою короткочасною пам’яттю.

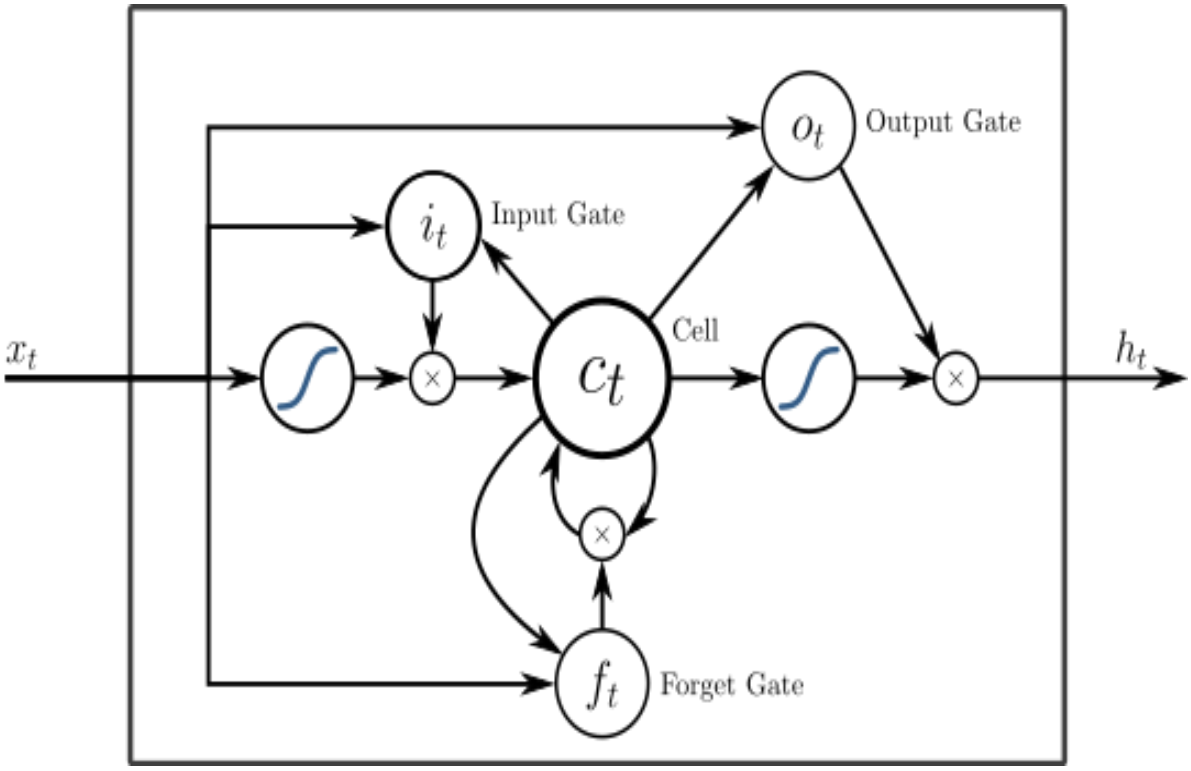


Рисунок 2.11 – Приклад архітектури нейронної мережі з довгою короткочасною пам’яттю [20]

В даному розділі були проаналізовані основні архітектури нейронних мереж та обрано архітектуру, що буде використано для сентиментального аналізу відгуків на фільми. Для застосування було обрано багатошаровий перцептрон, оскільки він дозволяє досягти досить високих показників точності (приблизно 90 відсотків) та є простим у реалізації.

3. МЕТОДИ ОПТИМІЗАЦІЇ МОДЕЛЮВАННЯ

3.1 Підготовка даних для нейронної мережі

У машинному навчанні з учителем дуже велику роль мають правильно підготовлені дані, оскільки при їх неправильній підготовці модель буде навчена хибним чином й давати результати, набагато гірші, ніж з даними, що пройшли коректну підготовку. Також ті чи інші алгоритми машинного навчання використовують певні математичні методи для створення моделі передбачення. Й для подання даних на вхід цих математичних методів дані мають відповідати певним вимогам. У разі нейронної мережі дані мають відповідати наступним вимогам:

- фіксована довжина вхідних даних;
- фіксована довжина вихідних даних;
- числова форма вхідних даних;
- числова форма вихідних даних.

Непідготовлені текстові дані не задовольняють даним вимогам, адже вони не є ані представленими у числовій формі, ані фіксованими за довжиною. Тож необхідно перетворити текстові відгуки на числові вектори фіксованого розміру. Це можна зробити, якщо співставити з кожним словом, що є у вхідних даних, індекс, тобто зробити словник з усіх слів, що є у вхідних даних. Після присвоєння індекса кожному слову, дані будуть мати фіксовану довжину та числову форму: кількість вхідних нейронів дорівнює розміру словника, а значення n-го вхідного нейрона дорівнює нулю, якщо n-го слова немає у вхідному тексті й дорівнює одиниці, якщо слово присутнє у тексті, що подається на вхід (рисунок 3.1).

Рисунок 3.1 – Перетворення текстових даних змінної довжини до числових даних фіксованої довжини

Підготовка ж даних на виході більш проста, ніж підготовка вхідних. Оскільки класів відгуків тільки два (позитивна або негативна оцінка), нейронна мережа бути мати два нейрони на вихідному шарі. Кожному вихідному нейрону відповідає клас відгука, і той клас, на виході відповідного шару якого найбільше значення, буде розцінюватись як передбачений.

3.2 Вплив методу навчання нейронної мережі на швидкість навчання

Широко застосовуваним методом для навчання нейронної мережі є метод градієнтного спуску та зворотнього розповсюдження помилки. Цей метод має суттєвий недолік, що полягає в тому, що обчислення градієнта функції багатьох змінних це дуже затратна операція, оскільки на кожній ітерації проводяться обчислення з урахуванням усієї вибірки даних. Рішення даної проблеми полягає в тому, щоб зменшити кількість змінних при обчисленні градієнта функції втрат.

Дана проблема вирішується вдосконаленням методу градієнтного спуску. В алгоритмі так званого mini-batch градієнтного спуску на кожній ітерації

обчислюється градієнт функції втрат, що залежить не від усієї вибірки даних, а лише від її частини, або партії (mini-batch), з фіксованим розміром цієї партії й з випадковими елементами вибірки. Це значно покращує швидкість тренування й не веде до суттєвого погіршення точності, однак веде до стрибкоподібної зміни значень функції. На рисунку 3.2 зображено графік зменшення значення функції втрат в залежності від номера ітерації при використанні звичайного градієнтного спуску.

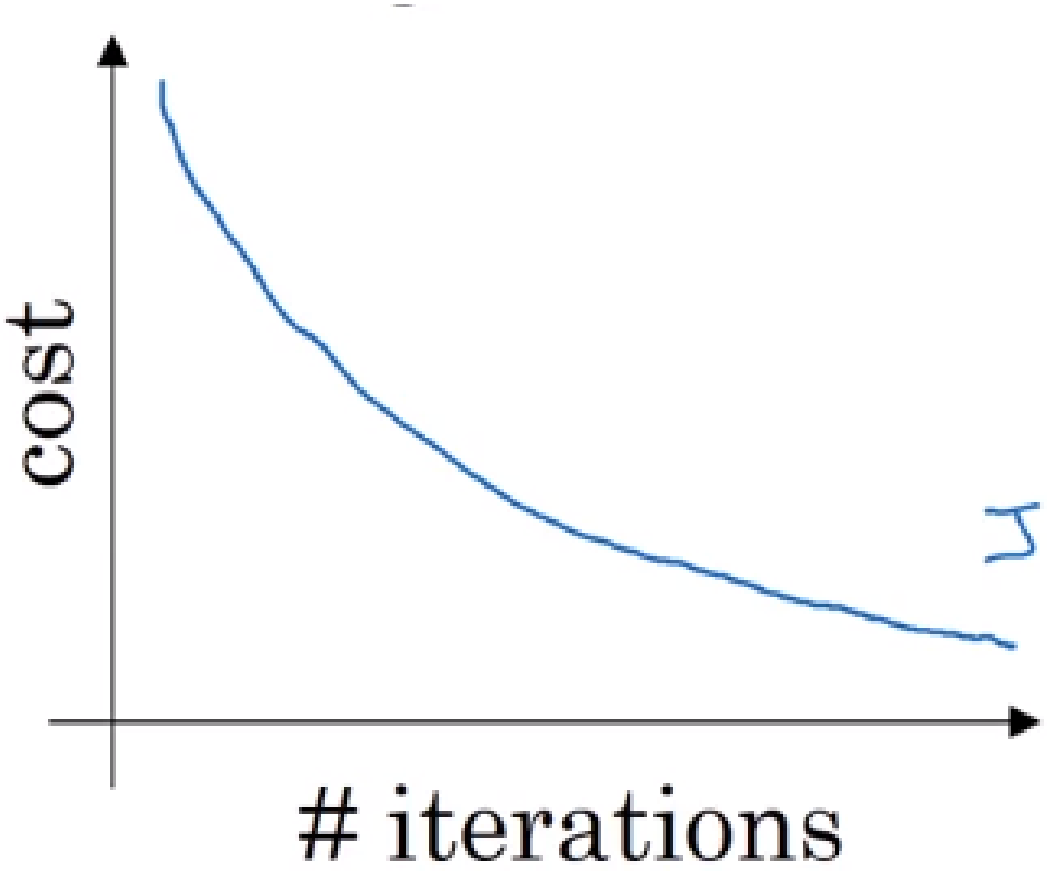


Рисунок 3.2 – Графік зменшення значення функції втрат в залежності від номера ітерації при використанні звичайного градієнтного спуску

На рисунку 3.3 зображено графік зменшення значення функції втрат в залежності від номера ітерації при використанні mini-batch градієнтного спуску. Mini-batch градієнтний спуск не впливає на досягнення мінімума значним чином, але робить процес зменшення функції менш стійким.

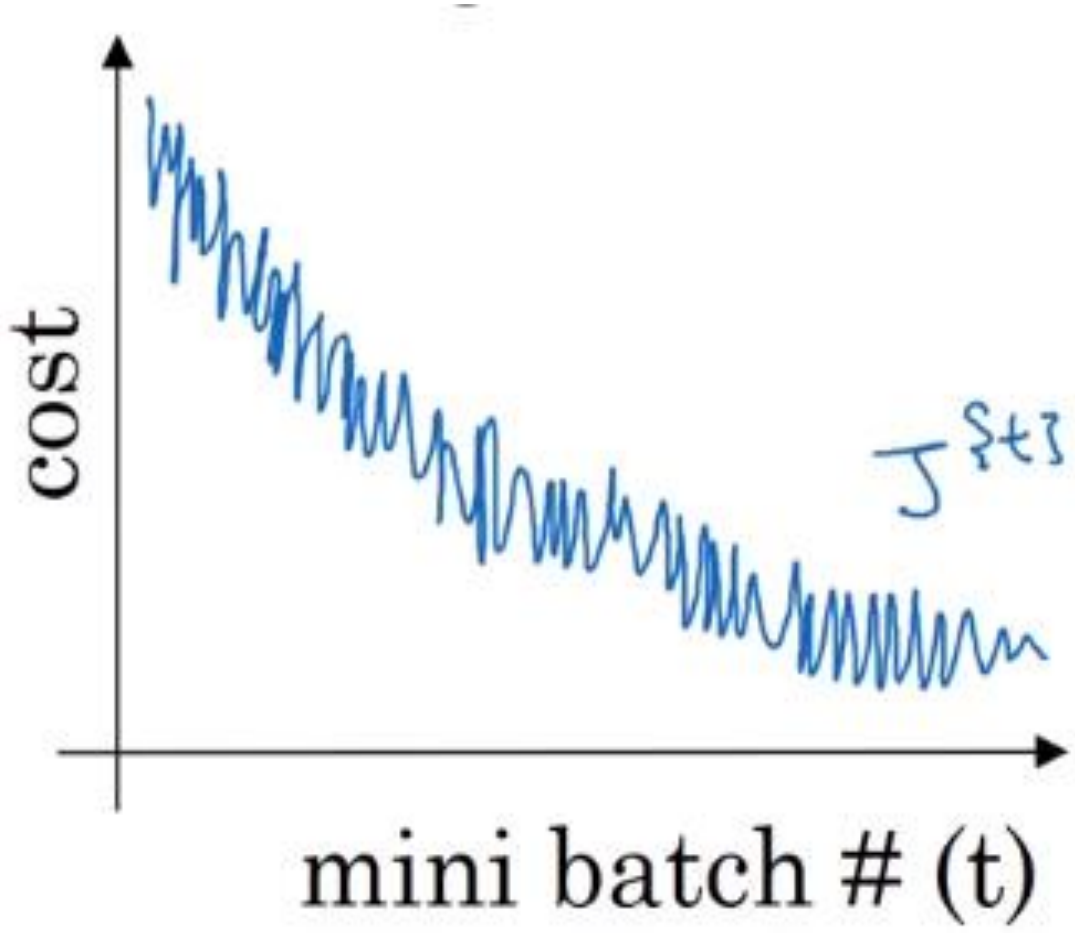


Рисунок 3.3 – Графік зменшення значення функції втрат в залежності від номера ітерації при використанні mini-batch градієнтного спуску

3.3 Вибір функції втрат

Функція втрат – функція, що у математичній формі виражає величину помилки моделі й, відповідно, відстань знайденого рішення від ідеального. На етапі тренування моделі у методах машинного навчання з учителем функція втрат використовується для корегування змінних коефіцієнтів моделі таким чином, щоб функція втрат зменшувалася. Для оцінки моделі за допомогою функції втрат у методах машинного навчання з учителем використовують тестові та валідаційні дані.

Валідаційні дані використовуються протягом навчання моделі для оцінки точності моделі й можуть бути використані, наприклад, для зупинки навчання у разі досягнення необхідного для рішення задачі класу точності.

Тестові дані використовуються для перевірки моделі вже після завершення процессу навчання моделі.

Отже, від функції втрат залежить як якість навчання моделі, так і якість її оцінки. Сьогодні при побудові моделей класифікації та регресії з участю нейронних мереж в основному використовують декілька основних варіацій функції втрат:

– середньоквадратична помилка;
– помилка перехресної ентропії.

Нижче кожна з цих функцій втрат розглянута більш детально.

3.3.1 Середньоквадратична помилка

Для оцінка точності передбачення моделі середньоквадратична помилка знаходить середній квадрат відстані між відомим значенням й значенням, передбаченим моделлю. Формулу середньоквадратичної помилки представлено на формулі 3.1.

$$J = \frac{1}{N}\sum_{i=1}^{N}(y_i - h_\theta(x_i))^2 \qquad 3.1$$

Середньоквадратична помилка виходить з ймовірнісної інтерпретації регресійної проблеми й, отже, краще підгодить до вирішення задач регресії, ніж до задач класифікації [21].

3.3.2 Помилка перехресної ентропії

Помилка перехресної ентропії в контексті класифікації виходить з ймовірності отримати нульову або одиничну ймовірність приналежності об’єкта до классу, що відповідає ймовірності позитивної або негативної відповіді на питання про приналежність об’єкта до класу (формула 3.2).

$$p(y_i = 1|x_i) = h_\theta(x_i) \text{ and } p(y_i = 0|x_i) = 1 - h_\theta(x_i) \quad 3.2$$

При поєднані цих двох формул та розповсюдженні на усі об’єкти виборки отримується формула категоріальної перехресної ентропії (формули 3.3 – 3.4).

$$p(y_i|x_i) = [h_\theta(x_i)]^{(y_i)}[1 - h_\theta(x_i)]^{(1-y_i)} \quad 3.3$$

$$J = -\sum_{i=1}^{N} y_i \log(h_\theta(x_i)) + (1 - y_i) \log(1 - h_\theta(x_i)) \quad 3.4$$

Помилка перехресної ентропії виходить з ймовірнісної інтерпретації проблеми класифікації й, отже, краще підгодить до вирішення задач класифікації, ніж до задач регресії [21].

3.4 Вибір коефіцієнта швидкості навчання

Коефіцієнт швидкості навчання (leaning rate) визначає, наскільки швидко будуть змінюватися ваги нейронної мережі при переході на наступну ітерацію (формула 3.5).

$$v \to v' = v - \eta \nabla C. \tag{3.5}$$

При виборі замалого значення learning rate модель буде навчатися занадто повільно, при виборі ж занадто великого – буде погіршено якість навчання через те, що модель не буде залишатися у знайденому локальному мінімумі (рисунок 3.4).

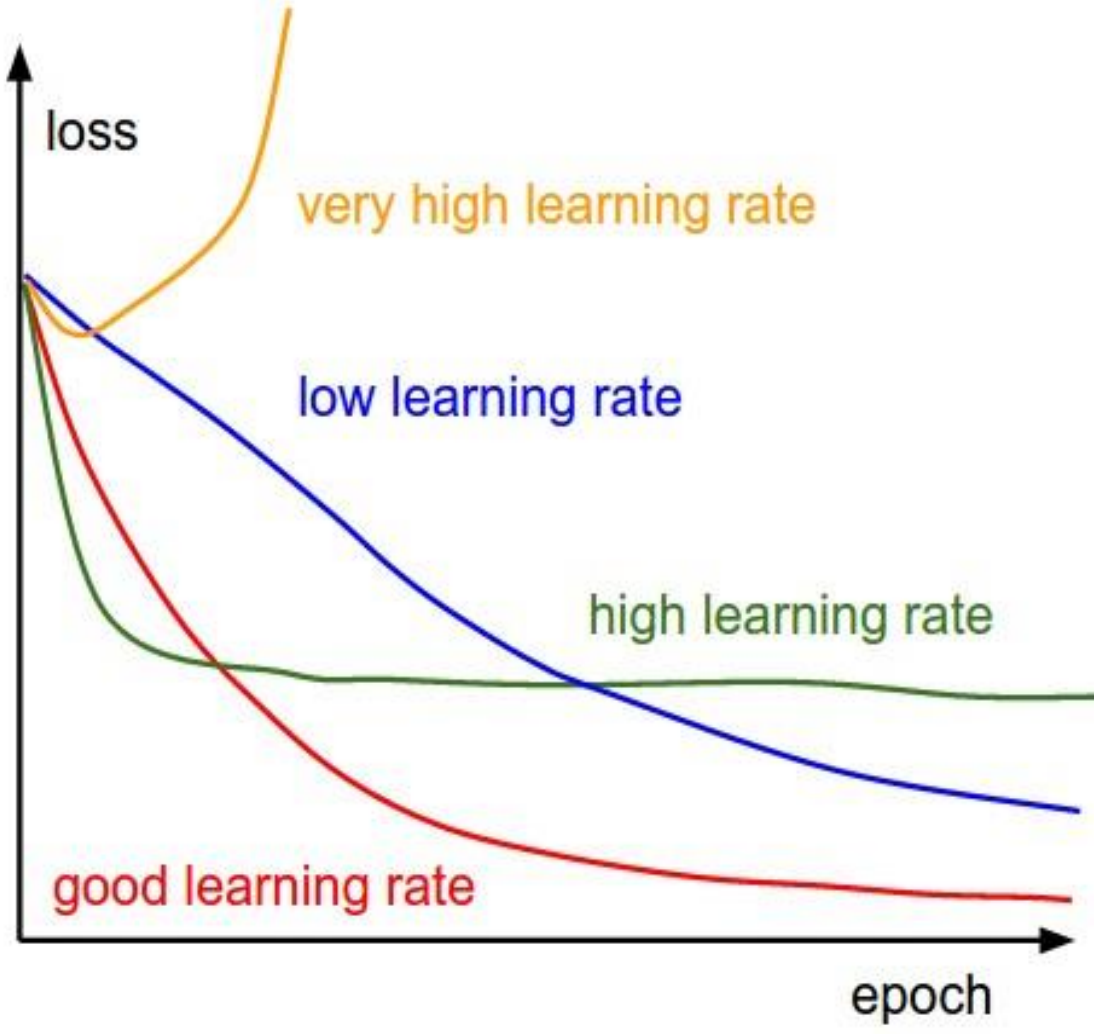


Рисунок 3.4 – Залежність зміни величини помилки при ітеративному навчанні моделі від правильності підбору коефіцієнту швидкості навчання

Загалом, не існує формули для підбору learning rate для кожного випадку, тож рекомендується підбирати його експериментально, поступово збільшуючи й відслідковуючи швидкість навчання та точність отриманої моделі.

Підсумовуючи даний розділ, можна зробити наступні висновки:

– для реалізації у програмному додатку обрано метод перетворення текстових даних змінної довжини до числових даних фіксованої довжини;
– для реалізції програмного додатку обрано mini-batch градієнтний спуск для навчання моделі;
– оскільки для класифікації краще підходить помилка перехресної ентропії, її обрано до залучення у навчанні нейронної мережі.

4. РОЗРОБКА ТА РЕЗУЛЬТАТИ

Для реалізації програмного додатку, що вирішує задачу сентиментальної класифікації з попередньою обробкою даних, було обрано алгоритм нейронних мереж прямого розповсюдження, методи очищення та метод зміни даних, мову Python та фреймворк TensorFlow.

4.1 Задачі реалізації й вибір інтерфейсу

Основними вимогами до вихідної програми є:

- можливість для користувача ініціалізувати тренування та тестування моделі із заданим алгоритмом й навчальними та тестовими даними;
- можливість зупинити процес навчання;
- можливість слідкувати за процесом навчання на кожній ітерації;
- можливість бачити кінцеві результати навчання та тестування для оцінки ефективності обраного алгоритму.

З даного списку видно, основну складність у реалізації программного додатку складає сааме алгоритм вирішення задачі, а не взаємодія з користувачем (рисунок 4.1). Отже, для даного програмного додатку було вирішено обмежитися консольним інтерфейсом взаємодії з користувачем, що задовольняє усім переліченим вимогам й добре підходить у разі необхідності реалізувати той чи інший алгоритм машинного навчання на практиці.

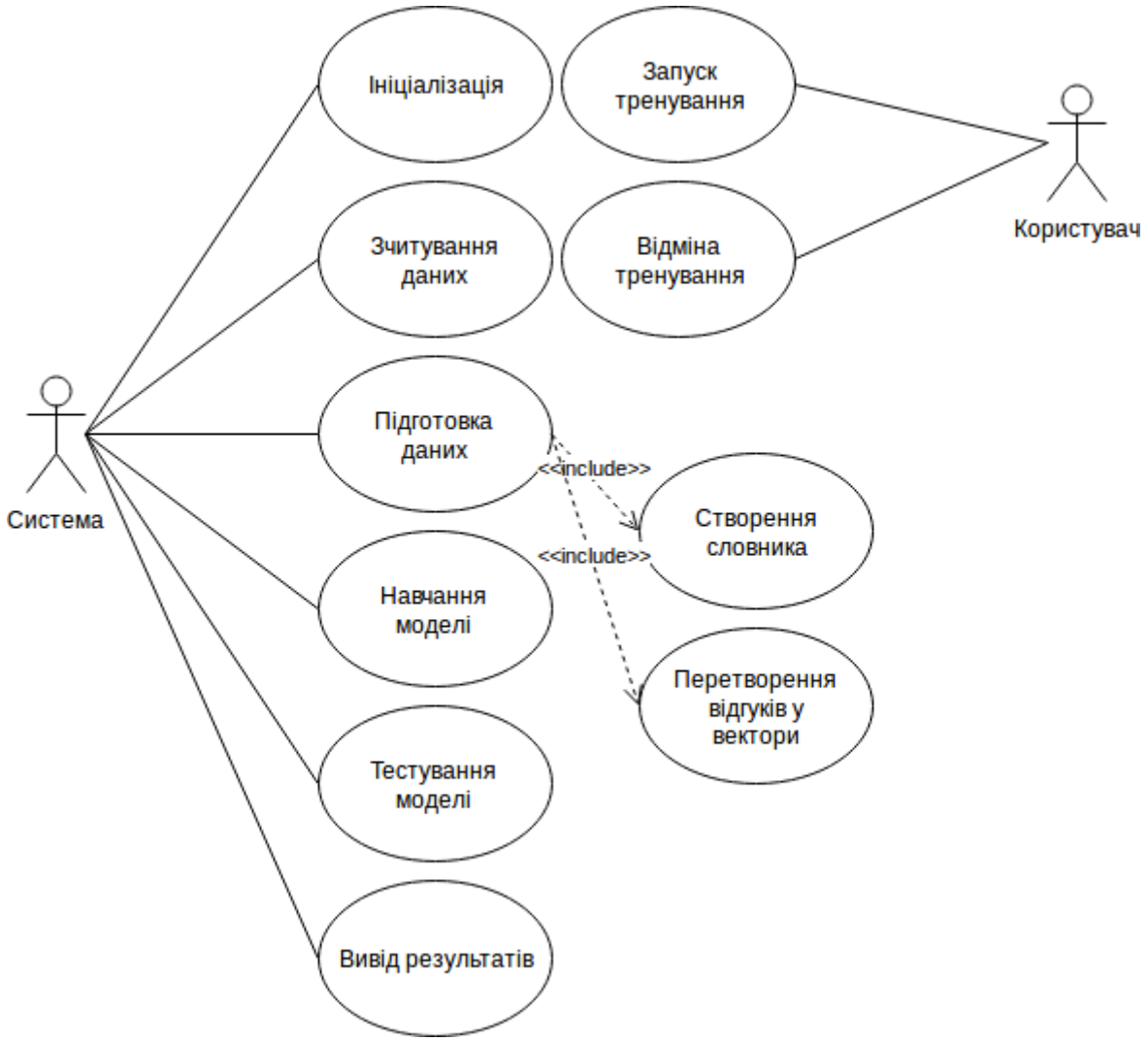


Рисунок 4.1 – Use Case діаграма програмного додатку

На рисунку 4.2 зображено вигляд консольного інтерфейсу програми з прикладом результатів перших ітерацій навчання.

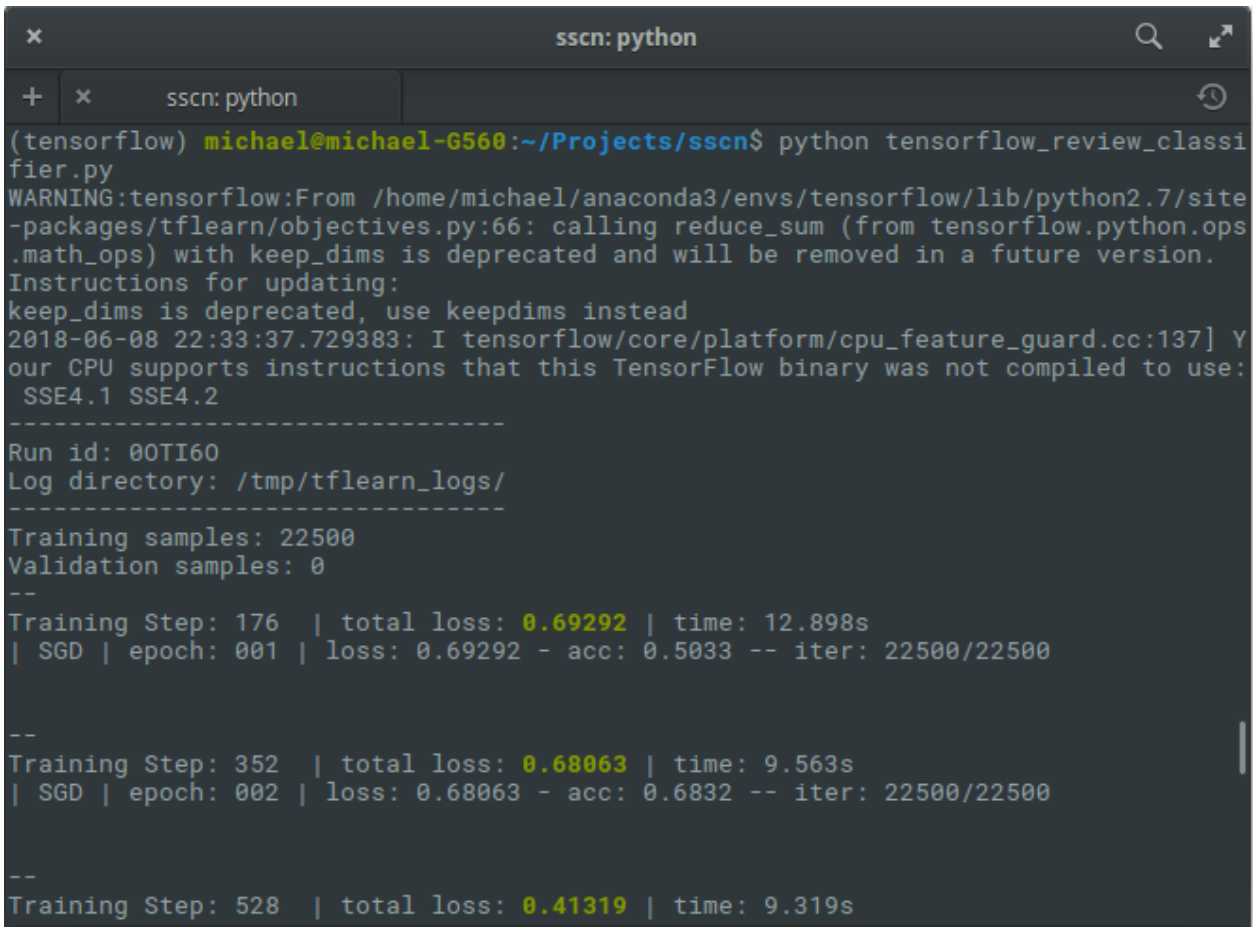


Рисунок 4.2 – Вигляд консольного інтерфейсу програми з прикладом результатів перших ітерацій навчання

4.2 Робота з даними

При навчанні моделі програма має взаємодіяти з великим обсягом даних (25 тисяч текстових відгуків). Для цього вона зчитує текстові дані безпосередньо з файлової системи. На рисунку 4.3 представлена діаграма послідовності роботи програмного додатку.

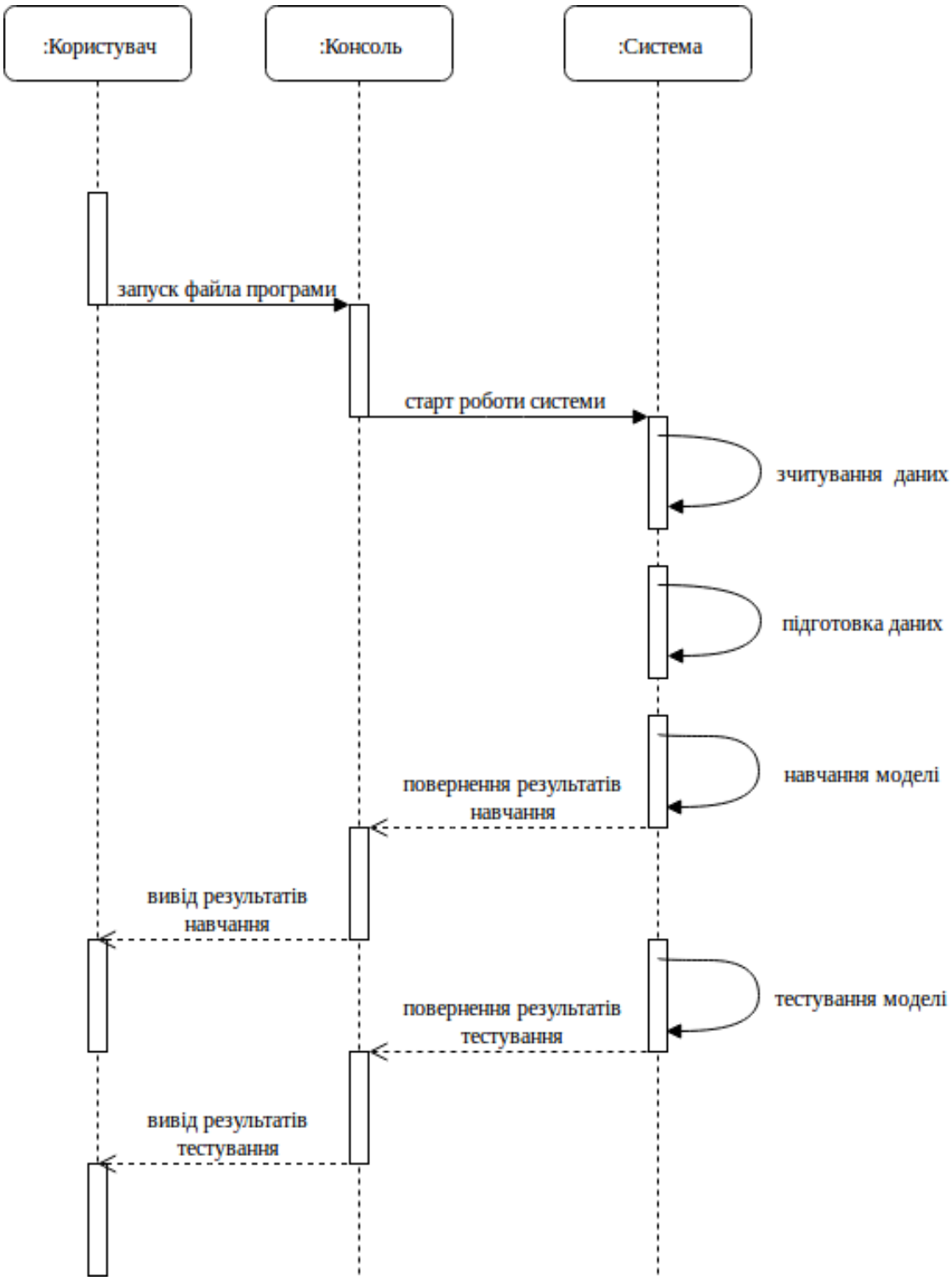


Рисунок 4.3 – Діаграма послідовності роботи програмного додатку

При цьому текстові відгуки конвертуються у числові вектори фіксованої довжини у два етапи. На першому етапі створюється словник з усіх слів, що входять до відгуків. Слова у словнику сортуються за кількістю входжень до відгуків. Після цього розмір словника обмежується 10000 словами, з

найбільшою кількістю входжень. За цю частину алгоритму відповідає функція prepare_input_data, код якої представлено на рисунку 4.4.

```
def prepare_input_data(reviews):
    word_counts = Counter()
    for _, row in reviews.iterrows():
        for word in row[0].split(' '):
            word_counts[word] += 1

    vocabulary = sorted(word_counts, key=word_counts.get, reverse=True)[:vocabulary_size]

    word_indexes = {}
    for index, word in enumerate(vocabulary):
        word_indexes[word] = index

    word_vectors = np.zeros((len(reviews), len(vocabulary)), dtype=np.int_)
    for word_vector_index, (_, text) in enumerate(reviews.iterrows()):
        word_vectors[word_vector_index] = review_to_vector(text[0], vocabulary, word_indexes)
    return word_vectors
```

Рисунок 4.4 – Код функції prepare_input_data

У коді даної функції викликається функція review_to_vector, що відповідає за другий етап конвертації відгуків до числових векторів – безпосередня генерація векторів, використовуючи словник. Код даного перетворення представлений на рисунку 4.5.

```
def review_to_vector(text, vocabulary, word_indexes):
    word_vector = np.zeros(len(vocabulary))
    for word in text.split(' '):
        if word in vocabulary:
            vector_index = word_indexes[word]
            word_vector[vector_index] = 1
    return word_vector
```

Рисунок 4.5 – Код функції review_to_vector

4.3 Результати моделювання

На рисунку 4.4 представлені результати моделювання. Як видно з результатів, реалізована программа дозволила досягти точності на тестових даних, що складає майже 90 відсотків (приблизно 87.56%).

Рисунок 4.4 – Результати моделювання (оцінка точності моделі на тестових даних)

4.4 Можливості вдосконалення алгоритму

Слід зазначити, що недоліком використаного у даному программному додатку перетворення текстових даних у вектор є те, що воно не враховує кількості входжень слова до тексту. Щоб вирішити дану проблему, можна записувати у вихідний вектор не лише одиниці у разі входження слова до тексту та нулі у разі відсутності слова у тексті, а деяке нормоване число між нулем та одиницею, величина якого б вказувала на число входжень слова до тексту.

Також використане перетворення не дозволяє врахувати порядку слів у тексті. Для цього можливо використовувати окремий вектор з нормованими індексами слів.

З результатів моделювання, що були отримані в результаті написання даного програмного додатку, можна зробити висновок про задовільнення написаною программою та її результатами поставлених цілей.

ВИСНОВКИ

У даному дипломному проекті здійснено аналіз алгоритмів та мов програмуання вирішення задачі попередньої обробки даних, типів сентиментальної текстової класифікації в її рамках. Було встановлено, що для попередньої обробки текстових даних у даній предметній області найкраще підходять методи зміни даних та очищення, використовуючи мову програмування Python та фреймворка Tensorflow.

За результатами проведеної роботи було розроблено додаток, що класифікує ставлення користувачів до переглянутих фільмів у текстових відгуках на позитивні та негативні.

ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Naive Bayes to Classify Movie Reviews Based on Sentiment [Електронний ресурс] / Suruchi Fialoke – Назва з екрана. Доступ : http://suruchifialoke.com/2017-06-10-sentiment-analysis-movie/
2. C# implementation for Naive Bayes sentiment classification engine [Електронний ресурс] – Назва з екрана. Доступ : https://github.com/amrish7
3. A Comparison of Event Models for Naive Bayes Text Classification In Proc [Електронний ресурс] / McCallum and Kamal Nigam – Назва з екрана. Доступ : www.cs.cmu.edu/~knigam/papers/multinomial-aaaiws98.pdf
4. Sentiment Analysis on Twitter Datasets [Електронний ресурс] – Назва з екрана. Доступ : https://github.com/alabid/sentimentstwitter
5. Saucy Kensington (@Book_Krazy) | Twitter [Електронний ресурс] – Назва з екрана. Доступ : https://twitter.com/book_krazy
6. Sentiment analysis para clasificar críticas de películas [Електронний ресурс] / Marco Herrero – Назва з екрана. Доступ : https://github.com/marhs/sentiment-analysis-svm/blob/master/sentiment_analysis.ipynb
7. Sentiment Analysis of Twitter data using combined CNN and LSTM Neural Network models [Електронний ресурс] / Pedro M Sosa – Назва з екрана. Доступ : https://github.com/pmsosa/CS291K
8. Dictionary-based sentiment analysis [Електронний ресурс] / Stefan Feuerriegel, Nicolas Pröllochs – Назва з екрана. Доступ : https://github.com/sfeuerriegel/SentimentAnalysis
9. The most popular language for machine learning analysis [Електронний ресурс] / Jean Francois Puget – Назва з екрана. Доступ : https://www.kdnuggets.com/2017/01/most-popular-language-machine-learning-data-science.html

10. Matlab vs Python [Електронний ресурс] – Назва з екрана. Доступ : http://www.pyzo.org/python_vs_matlab.html

11. Artificial neural network with layer coloring [Електронний ресурс] – Назва з екрана. Доступ : https://en.wikipedia.org/wiki/Artificial_neural_network#/media/File:Colored_neural_network.svg

12. Image of an artificial neuron [Електронний ресурс] – Назва з екрана. Доступ : http://en.citizendium.org/wiki/File:Artificialneuron.png

13. ImageNet Classification with Deep Convolutional Neural Networks [Електронний ресурс] / Alex Krizhevsky, Ilya Sutskever, Geoffrey E. Hinton – Назва з екрана. Доступ : http://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks.pdf

14. Структура багатошарового перцептрону [Електронний ресурс] – Назва з екрану. Доступ : https://upload.wikimedia.org/wikipedia/ru/d/de/Neuro.PNG

15. Text Classification using Neural Networks [Електронний ресурс] – Назва з екрану. Доступ : https://machinelearnings.co/text-classification-using-neural-networks-f5cd7b8765c6

16. Convolutional Neural Networks for Sentence Classification [Електронний ресурс] / Yoon Kim – Назва з екрану. Доступ : http://www.aclweb.org/anthology/D14-1181

17. Simple recursive neural network [Електронний ресурс] – Назва з екрану. Доступ : https://upload.wikimedia.org/wikipedia/commons/6/60/Simple_recursive_neural_network.svg

18. Генерация естественного языка, парафраз и автоматическое обобщение отзывов пользователей с помощью рекуррентных нейронных сетей [Електронний ресурс] / Тарасов Д. С. – Назва з екрану. Доступ : http://www.meanotek.ru/files/TarasovDS(2)2015-Dialogue.pdf

19. Neural Speech Recognizer: Acoustic-to-Word LSTM Model for Large Vocabulary Speech Recognition [Електронний ресурс] / Hagen Soltau, Hank Liao, Hasim Sak - – Назва з екрану. Доступ : https://arxiv.org/abs/1610.09975

20. Long Short-Term Memory Network [Електронний ресурс] – Назва з екрану. Доступ : https://upload.wikimedia.org/wikipedia/commons/5/53/Peephole_Long_Short-Term_Memory.svg

21. Picking Loss Functions - A comparison between MSE, Cross Entropy, and Hinge Loss [Електронний ресурс] / Rohan Varma – Назва з екрану. Доступ : http://rohanvarma.me/Loss-Functions/

Додаток 1. Специфікація

Позначення	Найменування	Примітки
Документація		
1	Пояснювальна записка	н/п
2	Додаток 2. Текст програмного модулю	н/п
3	Додаток 3. Опис програмного модулю	н/п
Компоненти		
1	Наївний баєсів класифікатор	Грунтується на так званій байєсівській теоремі і особливо підходить, коли розмірність вхідних параметрів велика. Наївний баєсів класифікатор використовує формулу Баєса, що описує ймовірність події, виходячи з попереднього знання умов, які можуть бути пов’язані з цією умовою.
2	Метод опорних векторів	Це модель навчання з учителем, основна ідея якої полягає в тому, щоб знайти гіперплощину, що розділяла б подані дані на класи з максимальною відстанню між об’єктами цих класів та площина.
3	Метод очищення даних	Метод полягає у виявленні і видаленні помилок і невідповідностей в даних з метою поліпшення якості даних.

4	Статистична міра TF-IDF	Використовується для оцінки важливості слова, як в контексті документа (TF), так і в контексті корпусу документів (IDF). Важливою особливістю TF-IDF є той факт, що набір даних не повинен змінюватися під час розрахунку.
7	Метод відображення текстових даних у вигляді хмари	Метод полягає у візуальному представленні списка категорій (слів у випадку даного програмного додатку)
8	Метод зміни даних	Метод полягає у перетворенні даних з початкової «сирої» форми у більш засвоюваний формат та організація наборів з різних джерел у єдине ціле для подальшої обробки.

Додаток 2. Текст програмного модулю

Імпорт основних бібліотек

"""

```
# Commented out IPython magic to ensure Python compatibility.
import re
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import string
import nltk
import warnings
warnings.filterwarnings("ignore", category=DeprecationWarning)

# %load_ext autoreload
# %autoreload 2
# %matplotlib inline
```

"""## **Завантаження бібліотеки функцій для попередньої обробки текстів**"""

```
# -*- coding: utf-8 -*-
import re
import numpy as np
from nltk.stem import SnowballStemmer


SUPPPORTED_LANG_STEMMER = {
    'SPA': SnowballStemmer('spanish'),
    'ENG': SnowballStemmer('english'),
    'PRT': SnowballStemmer('portuguese'),
}
```

```
#Видалення паттернів
def remove_pattern(input_txt, pattern):
    r = re.findall(pattern, input_txt)
    for i in r:
        input_txt = re.sub(i, '', input_txt)

    return input_txt

#Видалення символів пунктуації, специфічних для іспанської мови
def rm_pun_num_esp_cha(pandas_input):
  return pandas_input.str.replace("[^a-zA-Z#]", " ")

#Видалення літер, специфічних для іспанської мови
def rm_esp_cha(pandas_input):
  return pandas_input.str.replace("[^a-zA-Z0-9áéíóúÁÉÍÓÚâêîôÂÊÎÔãõÃÕñçÇ: ]", " ")

#Видалення неважливих слів
def rm_length_word(input_data, word_length=3):
    return input_data.apply(lambda x: ' '.join([w for w in x.split() if len(w) > word_length]))

#Токенізація даних
def tokenize(input_data):
    return input_data.apply(lambda x: x.split())

#Перевірка на мову
def _check_lang(lang):
  if lang in SUPPPORTED_LANG_STEMMER:
      return True
  else:
      return False

#Стеммінг для англійської мови
```

```
def stemmer(input_data, language='ENG'):
    if  _check_lang(language):
        stemmer = SUPPPORTED_LANG_STEMMER[language]
        return input_data.apply(lambda x: [stemmer.stem(i) for i in x])
    else:
        raise "Language {} not sopported for stemming".format(language)

#Об'єднання токеніхзованих даних
def join_tokenize(input_data, join_char=' '):
    return input_data.apply(lambda x: join_char.join(x))

#Екстракція хештегів
def hashtag_extract(input_data, flatten=True):
    hashtags = []
    for i in input_data:
        ht = re.findall(r"#(\w+)", i)
        if flatten:
            hashtags.append(ht)
        else:
            hashtags.append([ht])

    return sum(hashtags, [])

#Видалення хештегів
def hashtag_rm(input_data):
    return input_data.replace('#', '')

"""## **Завантаження бібліотеки функцій для перетворення та відображення у векторній формі**"""

# -*- coding: utf-8 -*-
import numpy as np
#from .error_rate import cer
#from functions import hashtag_rm
```

```
def corpus2vec(model, data, hashtag=True, debug=False, use_next=True, treshhold=0.5):
  # """Функція перетворення даних в векторну форму за допомогою моделей gensim word2vec.

  # Параметри
  # ----------
  # model:  gensim.models.KeyedVectors
  #     модель gensim word2vec
  # data: array of strings
  #     масив фраз
  # hashtag: bool
  #     прапорець необхідності видалення хештегів
  # debug: bool
  #     прапорець необхідності відладки
  #
  # Повертає
  # -------
  # дані у векторній формі за допомогою моделі gensim
  # """
   count = 1
   total_data = []
   length = len(data)
   for row in data:
       emb_frase = []
       for word_in_row in row:
           if hashtag:
               clean_word = hashtag_rm(word_in_row)
           else:
               clean_word = word_in_row
           try:
               emb_frase.append(model[clean_word])
           except Exception as exe:
```

```
            if use_next:
                total_cer = []
                for word in model.vocab:
                    total_cer.append(cer(word, clean_word))

                if np.asarray(total_cer).argmin() > treshhold:
                    find_word = model.index2word[np.asarray(total_cer).argmin()]
                else:
                    if debug:
                        print("{} not similar found in model".format(word_in_row))
                    emb_frase.append([0])

                if debug:
                    print("{} not found in vect, using distance, word used {}".format(
                        word_in_row,
                        find_word))
                emb_frase.append(model[find_word])

            else:
                if debug:
                    print(exe)
                emb_frase.append([0])

        total_data.append(np.asarray(emb_frase))

        if debug:
            prct = (count/length)*100
            print("Transform the {}% ".format(prct))
            count += 1

    return np.asarray(total_data)

def standard_word2vec_size(data, vec_size):
```

```
#"""Функція перетворення до стандартного розміру речень в тілі word2vec
#
#Параметри
#----------
#data:  numpy array
#    Масив речень з усіма вставками
#
#vec_size: int
#    Розмір вектору вставок
#
#Повертає
#-------
#масив numpy з даними, стандартизованими за одним розміром речення
#"""
empty = np.zeros((1, vec_size))
max_length = 0

for i in range(0, len(data)):
    if len(data[i]) > max_length:
        max_length = len(data[i])

full_empty = np.zeros((max_length, vec_size))

clean_data = []
for i in range(0, len(data)):
    if data[i].shape[0] <= 1:
        aux = full_empty
    elif data[i].shape[0] < max_length:
        value = max_length - data[i].shape[0]
        aux = data[i]
        for j in range(0, value):
            aux = np.append(aux, empty, axis=0)
    else:
```

```
            aux = data[i]
        clean_data.append(aux)

    return np.asarray(clean_data)

# Path to train and test files
train_path = 'train_dataset.csv'
test_path = 'test_tweets_dataset.csv'

train  = pd.read_csv(train_path)
test = pd.read_csv(test_path)

all_data = train.append(test, ignore_index=True)

train.head()

all_data['tidy_tweet'] = np.vectorize(remove_pattern)(all_data['tweet'], "@[\w]*")
all_data['tidy_tweet'] = rm_pun_num_esp_cha(all_data['tidy_tweet'])
all_data['tidy_tweet'] = rm_length_word(all_data['tidy_tweet'])

tokenized_tweet = tokenize(all_data['tidy_tweet'])
tokenized_tweet = stemmer(tokenized_tweet)
all_data['tidy_tweet'] = join_tokenize(tokenized_tweet)
all_data['hashtag'] = hashtag_extract(all_data['tidy_tweet'], flatten=False)
all_data['tidy_tweet'] = np.vectorize(remove_pattern)(all_data['tidy_tweet'], "#[\w]*")

tokenized_tweet = tokenize(all_data['tidy_tweet'])

all_data.head()

import math
for i in all_data['label'].unique():
    print(not math.isnan(i))
```

```
    data = all_data[all_data['label'] == i]['tidy_tweet']
    print(len(data))

"""## **Завантаження бібліотеки функцій для відображення даних у вигляді хмари**"""

# -*- coding: utf-8 -*-
import nltk
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from wordcloud import WordCloud


def _plot_wordcloud(all_words):
    """
    #Відображає хмару слів

    Параметри
    ----------
    all_words: string
        строка з словами для відображення, розділена пробілами
    """
    wordcloud = WordCloud(width=800,
                          height=500,
                          random_state=21,
                          max_font_size=110).generate(all_words)
    plt.figure(figsize=(10, 7))
    plt.imshow(wordcloud, interpolation="bilinear")
    plt.axis('off')
    plt.show()

def plot_labels_wordcloud(data_frame, data_name, label_name, add_all=False, only_all=False):
    """Функція обєднання токенізованих даних
```

```
    Параметри
    ----------
    data_frame: pandas data frame
        фрейм з хоча б наявними колонкою міток та колонкою з текстом
    data_name: string
        назва колонки з даними
    label_name: string
        назва колонки з мітками
    """
    if only_all:
        print("All data Wordcloud")
        _plot_wordcloud(' '.join([text for text in data_frame[data_name]]))
    else:
        if add_all:
            print("All data Wordcloud")
            _plot_wordcloud(' '.join([text for text in data_frame[data_name]]))

        for label in data_frame[label_name].unique():
            if label:
                data = data_frame[data_frame[label_name] == label][data_name]
                if len(data) > 0:
                    print("Label {} Wordcloud".format(label))
                    _plot_wordcloud(' '.join([text for text in data]))

def plot_hashtag_hist(hashtags):
    """Функція побудови гістограми тегів

    Параметри
    ----------
    hashtags: array of strings
        масив хештегів
    """
```

```
    hash_hist = nltk.FreqDist(hashtags)
    hash_pd = pd.DataFrame({'Hashtag': list(hash_hist.keys()),
                            'Count': list(hash_hist.values())})
    # selecting top 10 most frequent hashtags
    hash_pd = hash_pd.nlargest(columns="Count", n=10)
    plt.figure(figsize=(16, 5))
    ax = sns.barplot(data=hash_pd, x="Hashtag", y="Count")
    ax.set(ylabel='Count')
    plt.show()

plot_labels_wordcloud(all_data,'tidy_tweet', 'label', add_all=True, only_all=False)

"""# Машинне навчання"""

from sklearn.feature_extraction.text import (
    CountVectorizer,
    TfidfVectorizer
)
# будуємо BOW-матрицю
bow_vectorizer = CountVectorizer(max_df=0.90, min_df=2, max_features=1000, stop_words='english')
bow = bow_vectorizer.fit_transform(all_data['tidy_tweet'])

# будуємо TF-IDF матрицю
tfidf_vectorizer = TfidfVectorizer(max_df=0.90, min_df=2, max_features=1000, stop_words='english')
tfidf = tfidf_vectorizer.fit_transform(all_data['tidy_tweet'])

from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split
from sklearn.metrics import f1_score

train_bow = bow[:31962,:]
test_bow = bow[31962:,:]
```

```
# поділяємо дані на навчальну та тестову вибірки
xtrain_bow, xvalid_bow, ytrain, yvalid = train_test_split(train_bow, train['label'], random_state=42,
test_size=0.3)

# виконуємо логістичну регресію з BOW
lreg = LogisticRegression()
lreg.fit(xtrain_bow, ytrain) # тренуємо модель

prediction = lreg.predict_proba(xvalid_bow) # тестуємо на перевірочній вибірці
prediction_int = prediction[:,1] >= 0.3
prediction_int = prediction_int.astype(np.int)

print("Logistic Regression with BOW f1: {}".format(f1_score(yvalid, prediction_int)))


# Логістична регресія з використанням TF-IDF
train_tfidf = tfidf[:31962,:]
test_tfidf = tfidf[31962:,:]

xtrain_tfidf = train_tfidf[ytrain.index]
xvalid_tfidf = train_tfidf[yvalid.index]

lreg.fit(xtrain_tfidf, ytrain)

prediction = lreg.predict_proba(xvalid_tfidf)
prediction_int = prediction[:,1] >= 0.3
prediction_int = prediction_int.astype(np.int)

print("Logistic Regression with TF-IDF f1: {}".format(f1_score(yvalid, prediction_int)))

from sklearn.svm import LinearSVC
from sklearn import tree
from sklearn.naive_bayes import GaussianNB
```

```
from sklearn.neighbors import KNeighborsClassifier

# TF-IDF

"""KNeighbors
neigh = KNeighborsClassifier(n_neighbors=10)
neigh = neigh.fit(xtrain_tfidf.toarray(), ytrain)
y_pred = neigh.predict(xvalid_tfidf.toarray())
print("KNeighbors with TF-IDF f1: {}".format(f1_score(yvalid, y_pred)))
"""

"""Наївний баєс"""
gnb = GaussianNB()
gnb = gnb.fit(xtrain_tfidf.toarray(), ytrain)
y_pred = gnb.predict(xvalid_tfidf.toarray())
print("Naive Bayes Gaussian with TF-IDF f1: {}".format(f1_score(yvalid, y_pred)))


"""Класифікатор на базі дерева рішень"""
clf = tree.DecisionTreeClassifier()
clf = clf.fit(xtrain_tfidf, ytrain)
y_pred = clf.predict(xvalid_tfidf)
print("Decision Tree Classifier with TF-IDF f1: {}".format(f1_score(yvalid, y_pred)))


"""SVM"""
svm = LinearSVC()
svm.fit(xtrain_tfidf, ytrain)

y_pred = svm.predict(xvalid_tfidf)
print("SVM with TF-IDF f1: {}".format(f1_score(yvalid, y_pred)))

# BOW
```

```
"""KNeighbors
neigh = KNeighborsClassifier(n_neighbors=10)
neigh = neigh.fit(xtrain_bow.toarray(), ytrain)
y_pred = neigh.predict(xvalid_bow.toarray())
print("KNeighbors with TF-IDF f1: {}".format(f1_score(yvalid, y_pred)))
"""

"""Наївний баєс"""
gnb = GaussianNB()
gnb = gnb.fit(xtrain_bow.toarray(), ytrain)
y_pred = gnb.predict(xvalid_bow.toarray())
print("Naive Bayes Gaussian with TF-IDF f1: {}".format(f1_score(yvalid, y_pred)))


"""Класифікатор на базі дерева рішень"""
clf = tree.DecisionTreeClassifier()
clf = clf.fit(xtrain_bow, ytrain)
y_pred = clf.predict(xvalid_bow)
print("Decision Tree Classifier with TF-IDF f1: {}".format(f1_score(yvalid, y_pred)))


"""SVM"""
svm = LinearSVC()
svm.fit(xtrain_bow, ytrain)

y_pred = svm.predict(xvalid_bow)
print("SVM with TF-IDF f1: {}".format(f1_score(yvalid, y_pred)))

from scipy.sparse import hstack

# Stack BOW and TF-IDF
x_val = hstack([xvalid_tfidf,xvalid_bow])
```

```
x_train = hstack([xtrain_tfidf,xtrain_bow])

"""Логістична регресія"""
lreg = LogisticRegression()
lreg.fit(x_train, ytrain)
prediction = lreg.predict_proba(x_val)
prediction_int = prediction[:,1] >= 0.3
prediction_int = prediction_int.astype(np.int)

print("Logistic Regression with TF-IDF and BOW f1: {}".format(f1_score(yvalid, prediction_int)))

"""SVM"""
svm = LinearSVC()
svm.fit(x_train, ytrain)

y_pred = svm.predict(x_val)
print("SVM with TF-IDF and BOW  f1: {}".format(f1_score(yvalid, y_pred)))


"""Класифікатор на базі дерева рішень"""
clf = tree.DecisionTreeClassifier()
clf = clf.fit(x_train, ytrain)
y_pred = clf.predict(x_val)
print("Decision Tree Classifier with TF-IDF and BOW f1: {}".format(f1_score(yvalid, y_pred)))
```

Додаток 3. Опис програмного модуля

АНОТАЦІЯ

Волков О.О. «Попередня обробка даних (в тому числі заповнення відсутніх даних, видалення аномальних, згладжування даних для вирішення завдання зниження випадкових шумів, стиснення і нормалізація даних)». КПІ ім. Ігоря Сікорського, Київ, 2020.

В ході проведеної роботи було розроблено додаток, що класифікує ставлення користувачів до переглянутих фільмів у текстових відгуках користувачів на придбані товари на позитивні та негативні.

Реалізована система вирішує проблему сентиментального аналізу великого обсягу (25000) текстових відгуків на фільми з використанням відповідного методу попередньої обробки даних.

ЗМІСТ

1 ЗАГАЛЬНІ ВІДОМОСТІ ТА ФУНКЦІОНАЛЬНЕ ПРИЗНАЧЕННЯ 4
2 ОПИС ЛОГІЧНОЇ СТРУКТУРИ 5
3 ВИКОРИСТАНІ ТЕХНІЧНІ ЗАСОБИ.....9
4 ВХІДНІ ТА ВИХІДНІ ДАНІ 9

1. ЗАГАЛЬНІ ВІДОМОСТІ ТА ФУНКЦІОНАЛЬНЕ ПРИЗНАЧЕННЯ

Python на 2 у сфері машинного навчання. Ця мова програмування має безліч бібліотек, що спрощують розробку программ у сфері штучного інтелекту. Python є повноцінною мовою програмування, і багато організацій використовують його у своїх виробничих системах. Дана мова програмування поєднує у собі зручність роботи високорівневих мов програмування, швидкість низькорівневих, а також має за плечима десятки років розробок бібліотек для реалізації алгоритмів машинного навчання.

TensorFlow - бібліотека програмного забезпечення з відкритим кодом для чисельного обчислення з використанням графів потоку даних. Вузли на графу являють собою математичні операції, а грані графа представляють собою багатовимірні масиви даних (тензиси), що передаються між ними. TensorFlow був спочатку розроблений дослідниками та інженерами, що працюють у групі Google Brain Team в дослідницькій організації Google Machine Intelligence з метою проведення машинного навчання та дослідження глибинних нейронних мереж, однак ця система є достатньо загальною, щоб бути з успіхом застосованою у багатьох інших областях.

Оскільки мова програмуання Python має більше переваг, ніж інші, а фреймворк Tensorflow може ще більше спростити реалізацію програми й може бути використаний у поєднанні з мовою Python, для написання программного додатку до данного дипломного проекту добре підходить мова програмування Python у поєднанні з фреймворком Tensorflow. Також для реалізації програмного додатку, що вирішує задачу сентиментальної класифікації, було обрано алгоритм нейронних мереж прямого розповсюдження.

Оскільки нейронні мережі дозволяють досягти гарних результатів у текстовій класифікації, а також отримали широку популярність в останні роки, вони добре

підходять для написання программного додатку до данного дипломного проекту.

Оскільки стоїть задача класифікувати відгуки на два класи (позитивна або негативна тональність), то програмний додаток буде реалізовувати плоску класифікацію рівня документу.

2. ОПИС ЛОГІЧНОЇ СТРУКТУРИ

Основними вимогами до вихідної програми є:

- можливість для користувача ініціалізувати тренування та тестування моделі із заданим алгоритмом й навчальними та тестовими даними;
- можливість зупинити процес навчання;
- можливість слідкувати за процесом навчання на кожній ітерації;
- можливість бачити кінцеві результати навчання та тестування для оцінки ефективності обраного алгоритму.

З даного списку видно, основну складність у реалізації программного додатку складає сааме алгоритм вирішення задачі, а не взаємодія з користувачем (рисунок 4.1). Отже, для даного програмного додатку було вирішено обмежитися консольним інтерфейсом взаємодії з користувачем, що задовольняє усім переліченим вимогам й добре підходить у разі необхідності реалізувати той чи інший алгоритм машинного навчання на практиці.

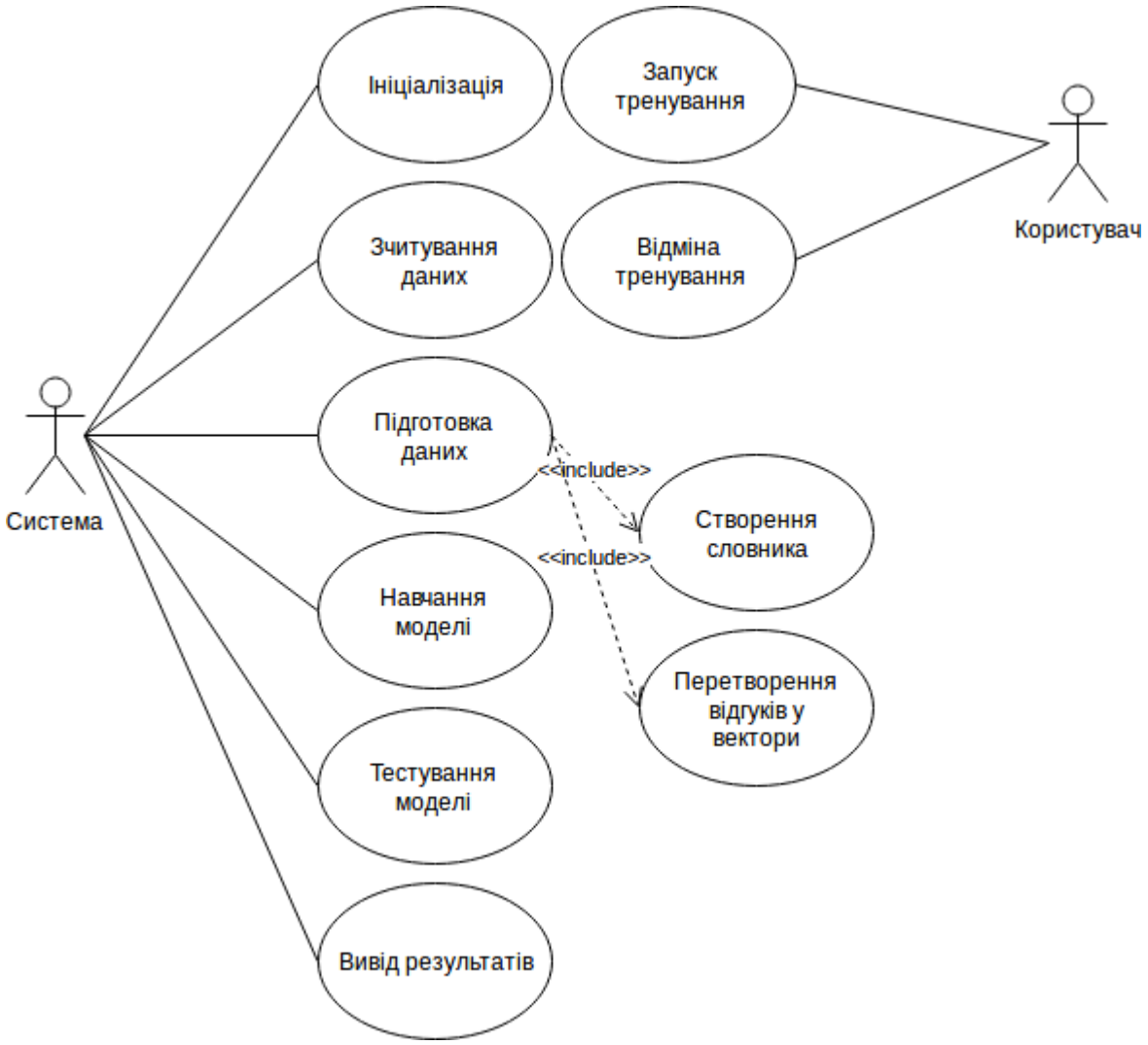
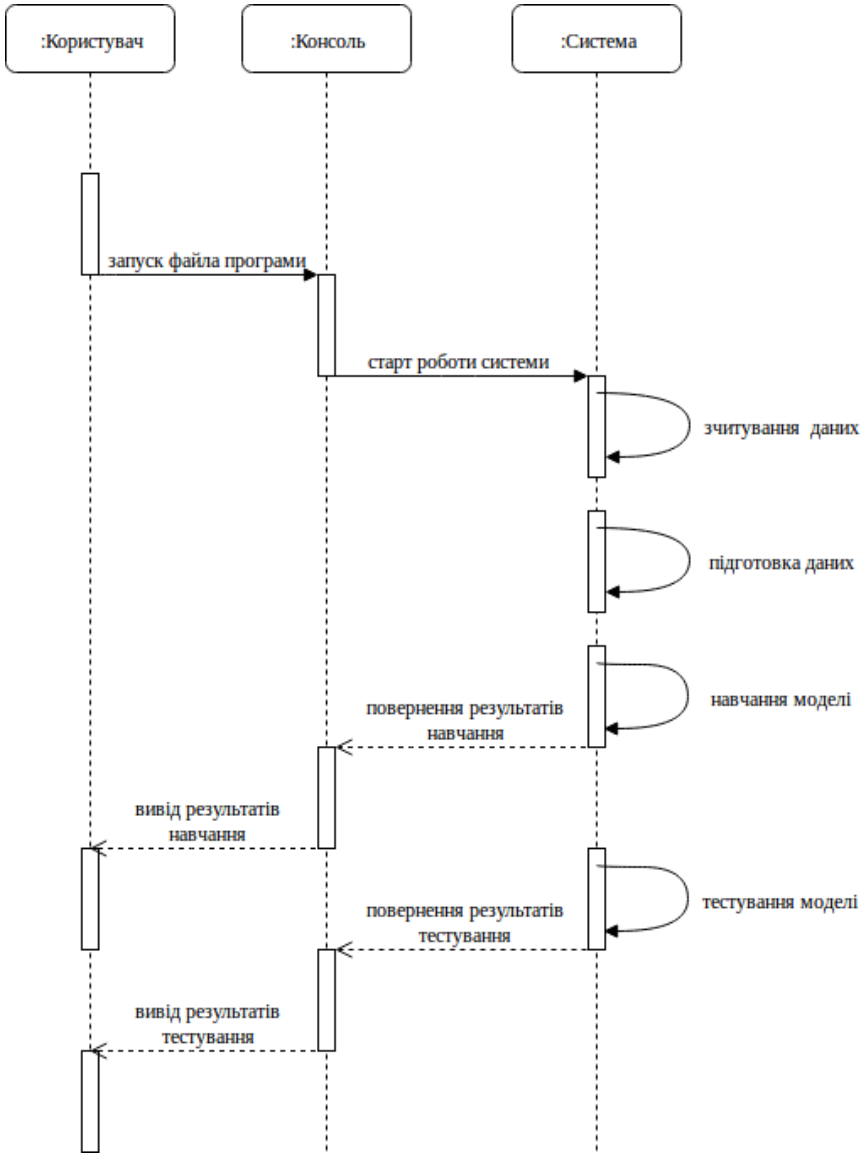


Рисунок 4.1 – Use Case діаграма програмного додатку

При навчанні моделі програма має взаємодіяти з великим обсягом даних (25 тисяч текстових відгуків). Для цього вона зчитує текстові дані безпосередньо з файлової системи. На рисунку представлена діаграма послідовності роботи програмного додатку:



При цьому текстові відгуки конвертуються у числові вектори фіксованої довжини у два етапи. На першому етапі створюється словник з усіх слів, що входять до відгуків. Слова у словнику сортуються за кількістю входжень до відгуків. Після цього розмір словника обмежується 10000 словами, з найбільшою кількістю входжень. За цю частину алгоритму відповідає функція prepare_input_data, код якої представлено на рисунку:

```
def prepare_input_data(reviews):
    word_counts = Counter()
    for _, row in reviews.iterrows():
        for word in row[0].split(' '):
            word_counts[word] += 1

    vocabulary = sorted(word_counts, key=word_counts.get, reverse=True)[:vocabulary_size]

    word_indexes = {}
    for index, word in enumerate(vocabulary):
        word_indexes[word] = index

    word_vectors = np.zeros((len(reviews), len(vocabulary)), dtype=np.int_)
    for word_vector_index, (_, text) in enumerate(reviews.iterrows()):
        word_vectors[word_vector_index] = review_to_vector(text[0], vocabulary, word_indexes)
    return word_vectors
```

У коді даної функції викликається функція review_to_vector, що відповідає за другий етап конвертації відгуків до числових векторів – безпосередня генерація векторів, використовуючи словник. Код даного перетворення представлений на рисунку:

```
def review_to_vector(text, vocabulary, word_indexes):
    word_vector = np.zeros(len(vocabulary))
    for word in text.split(' '):
        if word in vocabulary:
            vector_index = word_indexes[word]
            word_vector[vector_index] = 1
    return word_vector
```

3. ВИКОРИСТОВУВАНІ ТЕХНІЧНІ ЗАСОБИ

Для відтворення роботи даного дотатку було використувано стаціонарний комп’ютер з мікропроцесором Intel(R) Core(TM) i5-7500 CPU @ 3.40 GHz, з розміром RAM 16 гігабайтів та з 64-бітовою операційною системою Windows 10.

Також для підвищення швидкості роботи додатку можна використовувати технічні засоби з кращими характеристиками, аніж наведені вище.

4. ВХІДНІ ТА ВИХІДНІ ДАНІ

4.1 Вхідні дані

Вхідними даними для програмного додатку є текстові дані взяті з файлової системи (у випадку локального відтворення цим джерелом виступає файл з розширенням csv “train_dataset.csv”):

	id	label	tweet
0	1	0	@user when a father is dysfunctional and is s...
1	2	0	@user @user thanks for #lyft credit i can't us...
2	3	0	bihday your majesty
3	4	0	#model i love u take with u all the time in ...
4	5	0	factsguide: society now #motivation

4.2 Вихідні дані

На рисунку представлені результати моделювання (після тестування). Як видно з результатів (оцінка точності моделі на тестових даних), реалізована программа дозволила досягти точності на тестових даних, що складає майже 90 відсотків (приблизно 87.56%):

```
sscn: python
sscn: python
--
Training Step: 4576  | total loss: 0.25570 | time: 9.284s
| SGD | epoch: 026 | loss: 0.25570 - acc: 0.9072 -- iter: 22500/22500


--
Training Step: 4752  | total loss: 0.23436 | time: 9.084s
| SGD | epoch: 027 | loss: 0.23436 - acc: 0.9238 -- iter: 22500/22500


--
Training Step: 4928  | total loss: 0.06040 | time: 9.259s
| SGD | epoch: 028 | loss: 0.06040 - acc: 0.9943 -- iter: 22500/22500


--
Training Step: 5104  | total loss: 0.09502 | time: 9.036s
| SGD | epoch: 029 | loss: 0.09502 - acc: 0.9869 -- iter: 22500/22500


--
Training Step: 5280  | total loss: 0.38014 | time: 9.178s
| SGD | epoch: 030 | loss: 0.38014 - acc: 0.9592 -- iter: 22500/22500


--
('Test accuracy factor: ', 0.87560000000000004)
(tensorflow) michael@michael-G560:~/Projects/sscn$
```

Результат виконання методу відображення текстових даних у вигляді хмари зображений на наступному рисунку. Метод полягає у візуальному представленні списка категорій (слів у випадку даного програмного додатку):

```
plot_labels_wordcloud(all_data,'tidy_tweet', 'label', add_all=True, only_all=False)
```

All data Wordcloud

Статичстична міра TF-IDF використовується для оцінки важливості слова, як в контексті документа (TF), так і в контексті корпусу документів (IDF). Важливою особливістю TF-IDF є той факт, що набір даних не повинен змінюватися під час розрахунку. Вихідні дані функції що виконує логістичну регресію з TF-IDF:

```
prediction = lreg.predict_proba(xvalid_tfidf)
prediction_int = prediction[:,1] >= 0.3
prediction_int = prediction_int.astype(np.int)

print("Logistic Regression with TF-IDF f1: {}".format(f1_score(yvalid, prediction_int)))
```

```
Logistic Regression with BOW f1: 0.4303571428571428
Logistic Regression with TF-IDF f1: 0.40917782026768645
```

Метод опорних векторів (SVM) це модель навчання з учителем, основна ідея якої полягає в тому, щоб знайти гіперплощину, що розділяла б подані дані на класи з максимальною відстанню між об'єктами.

Наївний баєсів класифікатор грунтується на так званій байєсівській теоремі і особливо підходить, коли розмірність вхідних параметрів велика. Наївний баєсів класифікатор використовує формулу Баєса, що описує ймовірність події, виходячи з попереднього знання умов, які можуть бути пов'язані з цією умовою.

Вихідні дані функції що виконує наївний баєс, класифікатор на базі дерева рішень та SVM (метод опорних векторів) з мірою TF-IDF:

```
y_pred = svm.predict(xvalid_bow)
print("SVM with TF-IDF f1: {}".format(f1_score(yvalid, y_pred)))
```

```
Naive Bayes Gaussian with TF-IDF f1: 0.17205526278307934
Decision Tree Classifier with TF-IDF f1: 0.4493865030674847
SVM with TF-IDF f1: 0.38453713123092575
```

Вихідні дані класифікатору на базі дерева рішень, методу наївних векторів, логістичної регресії (TF-IDF та BOW) :

```
y_pred = clf.predict(x_val)
print("Decision Tree Classifier with TF-IDF and BOW f1: {}".format(f1_score(yvalid, y_pred)))
```

```
Logistic Regression with TF-IDF and BOW f1: 0.4349315068493151
SVM with TF-IDF and BOW  f1: 0.40784313725490196
Decision Tree Classifier with TF-IDF and BOW f1: 0.4678260869565218
```