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

Кафедра автоматизації проектування енергетичних процесів і систем

До захисту допущено:

Завідувач кафедри

_____Олександр КОВАЛЬ

«___»_____2020 р.

Дипломна робота

на здобуття ступеня бакалавра

за освітньо-професійною програмою «Програмне забезпечення розподілених систем»

спеціальності 121 «Інженерія програмного забезпечення»

на тему: «Web-сервіс з передбаченням жанру і тональності тексту»

Виконала:
Студентка IV курсу, групи ТВ-61
Єрохіна Анна Олексіївна _____

Керівник:
доцент, кандидат технічних наук
Кублій Лариса Іванівна _____

Рецензент:
доцент, кандидат технічних наук
Кублій Лариса Іванівна _____

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

Київ – 2020 року

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

Спеціалізація: Програмне забезпечення розподілених систем

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ Олександр КОВАЛЬ
(підпис)
”___”_____2020р.

ЗАВДАННЯ
на дипломну роботу студенту
Єрохіна Анна Олексіївна
(прізвище, ім’я, по батькові)

1. Тема роботи: Web-сервіс з передбаченням жанру і тональності тексту

керівник роботи: Кублій Лариса Іванівна, кандидат технічних наук, доцент
(прізвище, ім’я, по батькові науковий ступінь, вчене звання)

затверджена наказом вищого навчального закладу від “25” травня 2020 р. № 1168-с

2. Строк подання студентом роботи: 14 червня 2020 р.

3. Вихідні дані до роботи: мова програмування Python, середовище розробки Anaconda Jupyter Notebook

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити): розробити програмний продукт з передбачення жанру і тональності тексту; проаналізувати існуючі методи та підходи розв’язання проблеми багатоміткової класифікації; порівняти і реалізувати складніші підходи з використанням нейронних мереж; впровадити існуючий підхід передбачення тональності.

5. Перелік ілюстративного матеріалу: «Завдання розробки web-сервісу з передбаченням жанру і тональності тексту», «Аналіз існуючих підходів

класифікації тексту», «Засоби розробки», «Опис програмної реалізації», «Робота користувача з програмною системою», «Висновки».

6. Дата видачі завдання «11» жовтня 2019 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів виконання дипломної роботи	Термін виконання етапів роботи	Примітки
1.	Затвердження теми роботи	11 жовтня 2019 р.	
2.	Вивчення та аналіз задачі	13 квітня 2020 р.	
3.	Розробка архітектури та загальної структури системи	15 квітня 2020 р.	
4.	Розробка структур окремих підсистем	20 квітня 2020 р.	
5.	Програмна реалізація системи	27 квітня 2020 р.	
6.	Оформлення пояснювальної записки	10 травня 2020 р.	
7.	Захист програмного продукту	10 червня 2020 р.	
8.	Передзахист	10 червня 2020 р.	
9.	Захист	15 червня 2020 р.	

Студентка _____ _____Єрохіна А. О._____
(підпис) (прізвище та ініціали,)

Керівник роботи _____ _____Кублій Л. І._____
(підпис) (прізвище та ініціали,)

АНОТАЦІЯ

Метою дипломної роботи є дослідження і розробка програмного забезпечення для передбачення жанру за текстовими даними та визначення їх тональності з використанням технологій машинного навчання. Аналіз особливостей, недоліків та переваг використаних прийомів класифікації, обґрунтування обраного кінцевого методу. Результат роботи програми може бути використаний як підказка людині-оператору при введенні тексту, що вимагає визначення певної категорії. Модель побудована на прикладі передбачення жанрів фільму.

Під час виконання роботи було використано такі методи багатоміткової класифікації: Binary Relevance, Classifier Chains, Label Powerset і RAkelD. Також розроблено 7 варіантів нейронних мереж з «пам’яттю». Найкращий результат дала мережа з шарами GRU і LSTM — 0,91 за метрикою roc-micro.

Дипломна робота має обсяг 70 аркушів, містить 4 додатки і 29 посилань. Також наведено 28 рисунків і 3 таблиці.

Ключові слова: машинне навчання, багатоміткова класифікація, тональність тексту, нейронна мережа, дані.

ABSTRACT

The purpose of the thesis is to study and develop software to predict the genre of the textual description of the film and determine its tone using machine learning technologies. Analysis of features, disadvantages and advantages of the used classification methods, substantiation of the chosen final method. The result of the program can be used when filling in the information about the film by the administrator of the service for watching movies.

The following multi-label classification methods were used during the work: Binary Relevance, Classifier Chains, Label Powerset and RAkelD. There are also 7 variants of neural networks with "memory". The best result was given by the network with the GRU layer — 0.53 on the accuracy metric.

Thesis has a volume of 70 sheets and contains 4 appendices and 29 references. There are also 28 figures and 3 tables.

Key words: machine learning, multilabel classification, sentiment analysis, neural network, data.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

UX — досвід користування (англ. User Experience)

GPU — графічний процесор (англ. Graphics Proccesing Unit)

CPU — центральний процесор (англ. Central processing unit)

TPU — тензорний блок обробки (англ. Tensor Processing Unit)

MSE — середньоквадратична помилка (англ. minimum square error)

TF-IDF — частота слова — обернена частота документа (англ. Term Frequency — Inverse Document Frequency)

TP — істинно-позитивно (англ. true positives)

FP — хибно-позитивно (англ. false positives)

FN — помилково-негативно (англ. false negatives)

TN — істинно-негативно (англ. true negatives)

ROC-AUC — площа під ROC-кривою (англ. area under receiver operating characteristic curve)

GRU — керований рекурентний блок (англ. Gated recurrent units)

LSTM — довга короткочасна пам'ять (англ. Long short-term memory)

NLP — обробка природної мови (англ. Natural-language processing)

RNN — рекурентні нейронні мережі (англ. Recurrent neural networks)

ООП — Об’єктно-орієнтоване програмування (англ. Object-oriented programming)

ВСТУП

Останнім часом актуальною є проблема автоматизації, оскільки її переваги над механічною і рутинною роботою суттєві, а саме: підвищення продуктивності, поліпшення якості, економія часу, збільшення прибутку. Особливих успіхів було досягнуто у сфері машинного навчання, що стрімко розвивається останні 10 років і показує високі результати й використовується у формуванні пошукової видачі, аналізі маркетингової інформації, управлінні хмарними серверами, машинному перекладі, розпізнаванні образів на фотографіях і відео, відловлюванні спаму в пошті і соціальних мережах, побудові маршрутів на картах і навіть при визначенні музики, яка грає поруч із користувачем.

Метою роботи є дослідження й програмна розробка підходів і методів класифікації текстового опису за жанрами та визначення тональності тексту з використанням технологій машинного навчання. А також — реалізувати існуючі підходи на наявних даних для порівняння результатів, розробити власні моделі нейронних мереж. Розкрити особливості, недоліки й переваги використаних прийомів класифікації, обґрунтувати обраний кінцевий метод.

Основним призначенням даного програмного продукту є оптимізація процесу ідентифікації жанрів шляхом розв’язання задачі класифікації на прикладі сюжету фільму в текстовій формі.

Розроблена система може бути застосована сервісами з перегляду фільмів, наприклад, для пришвидшення й полегшення роботи адміністратора сайту, оскільки замість пошуку необхідного жанру серед сотень найменувань, система класифікації підказує кілька найбільш вірогідних варіантів.

Завданнями роботи є вивчення способів класифікації тексту, визначення його тональності, аналіз існуючих підходів реалізації задачі, виявлення

найефективнішого методу розв’язання проблеми даної роботи й на його основі створити програмний продукт для класифікації опису тексту за жанрами, а також визначення тональності тексту.

Серед впроваджених систем багатоміткової класифікації можна виділити популярний серед розробників програмного забезпечення сайт «stackoverflow.com» [1]. Це платформа, яка дає можливість поставити запитання на технічну тему та отримати на нього відповідь від інших користувачів. Сервіс пропонує користувачам теги-технології, які стосуються створеного ними питання для їхньої класифікації. Це надає змогу користувачам знаходити питання у категоріях, на яких вони знаються, і відповідати на питання. Зрозуміло, що одного запитання можуть стосуватися декілька технологій. Проблемою є те, що такі системи є комерційними, тому неможливо дізнатися ні деталі реалізації, ні точності впровадженої моделі, які, до того ж, залежать від задачі, що розв’язується. Тому було прийнято рішення сфокусуватися на проблемі передбаченні жанрів фільмів через наявність відкритих даних, які можна використовувати для навчання моделі. Таку модель можна однозначно порівняти з підходами, описаними в наукових статтях.

Даний програмний продукт реалізовано мовою програмування Python 3.6 з використанням фреймворків Flask і TensorFlow і бібліотек sklearn, nltk, pandas, matplotlib. У роботі наводиться обґрунтування вибору метрики оцінювання результату передбачення, порівняння і застосування таких методів, як: Binary Relevance, Classifier Chains, Label Powerset та RAkelD, аналіз їхніх результатів. У підсумку, модель із найвищою точністю впроваджено у веб-сервіс, який за наданим текстом прогнозує жанри, до яких потенційно може належати цей текст, та оцінює його тональність.

У результаті створено сервіс із класифікації тексту, проаналізовано й реалізовано популярні підходи машинного навчання.

1. ЗАДАЧА КЛАСИФІКАЦІЇ ТЕКСТУ ЗА ЖАНРАМИ

Метою даної роботи є реалізація системи класифікації тексту за жанрами, визначення ступеню його емоційного забарвлення, проведення досліджень і порівнянь серед існуючих методів класифікації.

Класифікація текстів за жанрами проводиться на прикладі поданих англійською мовою розгорнутих текстових описів сюжетів фільмів. Елементами множини жанрів, до яких система відноситиме той чи інший фільм, є: «drama», «comedy», «action», «romance», «thriller», «crime», «horror», «western», «musical», «adventure». Тональність тексту, в свою чергу, матиме значення «позитивний» чи «негативний».

Завдання, які треба виконати для реалізації системи, такі:

— аналіз підходів до розв’язання задачі класифікації текстів за жанрами і визначення їхньої тональності;

— дослідження переваг і недоліків існуючих прийомів класифікації, обґрунтування вибору методу для реалізації;

— реалізація моделі багатоміткової класифікації. При цьому множину жанрів визначають 10 найменувань;

— застосування функції визначення тональності тексту; тональність тексту визначається як «позитивна» чи «негативна»;

— реалізація інтерфейсу і серверної частини.

Реалізована система повинна забезпечувати такі функції:

— багатоміткова класифікація текстової інформації за жанрами і мати показники точності від 0,5 за метрикою accuracy [2];

— визначення тональності тексту;

— мати інтерфейс користувача.

2. АНАЛІЗ ПРОБЛЕМИ КЛАСИФІКАЦІЇ ТЕКСТУ

Популярні Інтернет-платформи й сервіси зараз використовують технології машинного навчання, щоб вдосконалити функціонал і зробити контент орієнтованим індивідуально на кожного користувача і на конкретну проблему. Переваг у цього достатньо: підвищення ефективності й прибутку, у клієнта формується позитивний досвід користування сайтом (англ. User Experience, UX), поліпшення якості інформації.

2.1 Аналіз існуючих методів класифікації

Основним підходом для побудови моделі передбачення жанрів за текстом є багатоміткова класифікація, яка передбачає набір відповідних міток для нового екземпляра даних. Така тактика відрізняється від класичного підходу багатокласової класифікації. Якщо традиційний класифікатор поверне лише одне значення, багатомітковий повинен створити вектор вихідних значень. Тобто, метод машинного навчання багатоміткової класифікації використовується тоді, коли кожен об’єкт представлено одним екземпляром і асоціюється з декількома мітками класів одночасно.

Проблема багатоміткової класифікації неодноразово розв’язувалася, проте прикладів систем класифікації фільмів за жанрами не знайдено. Причиною цього є орієнтованість на закриту частину ресурсів, які дають можливість переглядати фільми, оскільки така система призначена передусім для адміністраторів і недоступна для звичайних користувачів. І все ж при дослідженні літератури було знайдено статті з різними підходами до розв’язання цієї проблеми [3].

Для класифікації фільмів за жанрами із сайту «imdb.com» розробники використали ансамблі бінарних класифікаторів, одним з яких виявився популярний метод Binary Relevance [4]. Цей метод є інтуїтивним рішенням для навчання на об’єктах з багатьма мітками. Він створює ряд незалежних бінарних класифікаторів окремо для кожної мітки (рисунок 2.1).

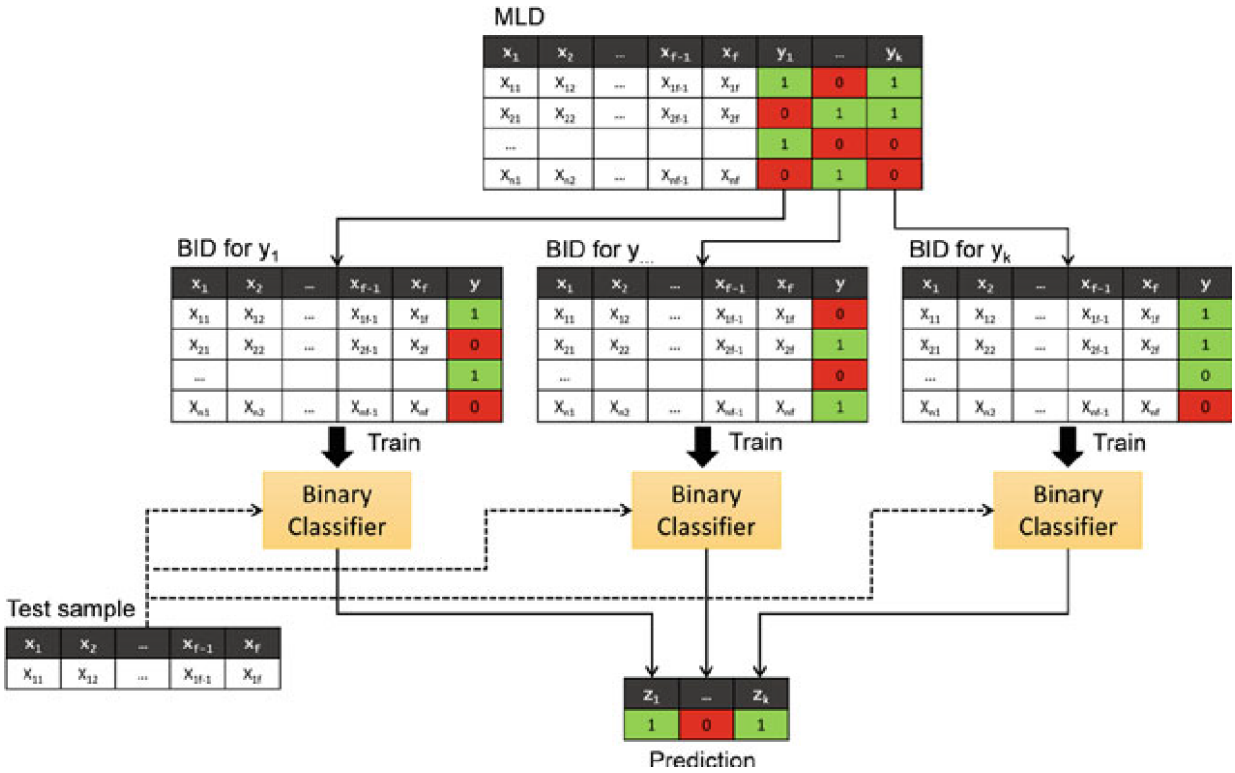

Рисунок 2.1 — Схема роботи методу Binary Relevance [4]

Як видно з рисунка 2.1, початкова задача багатоміткової класифікації означає наявність кількох цільових колонок, коли звичайні методи класифікації можуть працювати лише з однією. Ідея методу полягає в тому, щоб перетворити багатоміткову класифікацію на кілька задач бінарної класифікації та об’єднати їхні результати. Це відбувається шляхом поділу набору даних на кілька наборів, які містять лише одну цільову колонку. Оптимізовані реалізації даного методу жодним чином не дублюють дані, а лише використовують відповідну колонку як цільову для відповідної моделі, ігноруючи інші.

Суттєвим недоліком такого підходу є те, що моделі не мають ніякої інформації про передбачення інших, а відповідно не можуть використати це для корекції свого передбачення. Іншими словами, ігнорується кореляція між мітками. Для прикладу, мітка «horror» рідко використовується з міткою «comedy». Хоч мітки і не є взаємовиключними, врахування їхніх кореляцій здатне збільшити точність.

До переваг підходу Binary Relevance належать:

— простота інтерпретування;

— простота використання;

— простота навчання через відсутність гіперпараметрів.

Інші дослідники на тому ж наборі даних надали перевагу Classifier Chains, який поєднує в собі обчислювальну ефективність методу Binary Relevance і може враховувати залежності між мітками [5]. Метод будує для кожної мітки бінарний класифікатор, використовуючи вихідну інформацію попереднього класифікатора як вхідну для поточного. Такий алгоритм, як правило, дає більш високі результати порівняно з Binary Relevance.

Також у дослідженнях розробники порівнюють Classifier Chains з іншим підходом розв’язання проблеми багатоміткової класифікації. Підхід Label Powerset — це простий метод, який розглядає кожну унікальну множину міток у вихідній навчальній вибірці як один новий клас у перетворених даних (рисунок 2.2). До нової задачі можуть застосовуватися будь-які алгоритми багатокласової класифікації [6].

Для прикладу, можна застосувати стратегію One-Vs-Rest (також відома як One-Vs-All) [7], відповідно до якої буде виконуватися задача бінарної класифікації послідовно для кожного класу, де кожна ітерація буде сприймати один клас як цільовий, а інші як його відсутність. При застосуванні такого підходу буде навчено стільки моделей, скільки існує класів. Важливо, що кожна модель має повертати не бінарний результат «1» — клас присутній, «0» — клас відсутній, а ймовірність присутності класу. Це зумовлено тим, що при першому підході можливо отримати кілька позитивних відповідей без можливості обрати

серед них ту, яка найкраще підходить за умови, що клас може бути лише один.

Варто зазначити, що хоч стратегія One-Vs-Rest і досить схожа на Binary Relevance, це різні алгоритми. One-Vs-Rest будує бінарний класифікатор для кожного значення класу в цільовій змінній, а Binary Relevance будує бінарний класифікатор для кожної цільової змінної. Тому можливе сумісне використання стратегій, якщо одна з цільових змінних має більше ніж 2 класи.

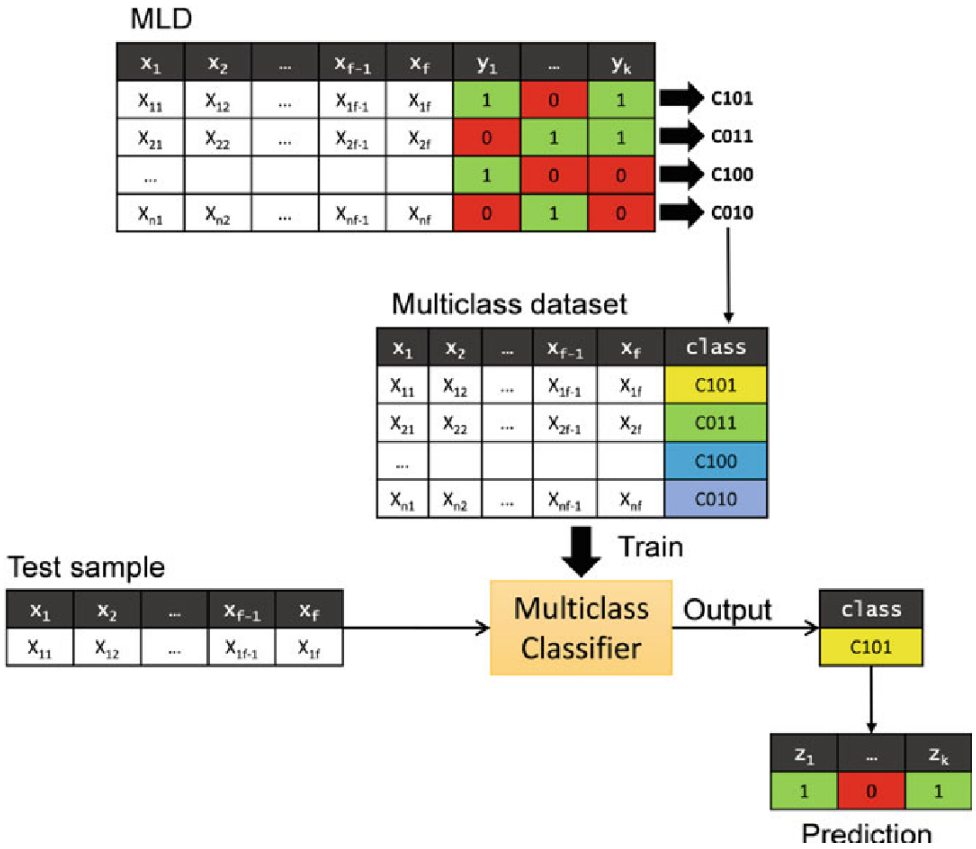

Рисунок 2.2 — Схема роботи методу Label Powerset [6]

Передбачений алгоритмом клас у новій задачі однозначно відповідає певній множині міток у вихідній задачі. Одна з проблем методу Label Powerset полягає в тому, що після перетворення даних велика частина нових класів містить дуже мало об’єктів, і розподіл об’єктів в нових класах є вкрай незбалансованим.

Підхід RAkelD — це метод, що генерує випадкові підмножини міток, тренуючи класичний класифікатор на кожній підмножині. Перевагами цього

методу є простота обчислень, більше можливостей прогнозування, здатність враховувати залежності міток. Недоліком даного підходу є нездатність використовувати невизначені раніше мітки для класифікації [8].

2.2 Критерії порівняння методів

Для оцінювання якості роботи моделі необхідно обрати метрику, з ростом якої підвищувалася б якість моделі. Також важливою є інформативність метрики. Наприклад, якщо метрика набуває значення від 0 до 1, то 0 означає, що модель зовсім не пояснює дані, а при 1 — працює ідеально. Підбираючи метрики, можна змінити поведінку моделі.

Розглянемо ситуації, які виникають при виконанні задачі класифікації. Припустимо, що відомі правильні категорії для деякої кількості документів. Треба згрупувати відповіді гіпотетичного аналізатора так:

— істинно-позитивні (true positives, TP) — ті категорії, які очікували побачити і отримали на виході;

— хибно-позитивні (false positives, FP) — категорії, яких бути на виході не повинно, і аналізатор їх помилково видав на виході;

— помилково-негативні (false negatives, FN) — категорії, які очікували побачити, але аналізатор їх не визначив;

— істинно-негативні (true negatives, TN) — категорії, яких бути на виході не повинно, і на виході аналізатор їх не видав.

В ідеалі кількість неправильних відповідей дорівнює нулю, і на цьому можна зупинитися. Проте на практиці таких систем не буває, і аналізатор буде працювати з помилками на тестовій вибірці. Враховуючи різні типи ситуацій при передбаченні, можна створити різні метрики, які по-різному оцінюють модель.

Наприклад, обираючи метрику, спрямовану на точність (англ. precision), буде створено модель, яка намагатиметься передбачувати якомога менше жанрів, не властивих тексту. Нижче подано формулу (2.1) розрахунку точності:

$$precision = \frac{TruePositive}{TruePositive + FalsePositive} \quad (2.1)$$

де TruePositive — істинно-позитивні результати класифікатора; FalsePositive — хибно-позитивні результати класифікатора.

На противагу цьому виступає повнота (англ. recall), оптимізуючи яку буде отримано модель, яка намагатиметься передбачити всі жанри, властиві тексту, навіть якщо доведеться видати зайве [9]. Розрахунок цієї метрики здійснюється за формулою (2.2):

$$recall = \frac{TruePositive}{TruePositive + FalseNegative} \quad (2.2)$$

де FalseNegative — помилково-негативні результати класифікатора.

При цьому необхідно враховувати баланс між точністю і повнотою, щоб не допускати помилкових спрацювань і пропусків справжніх відповідей. Для розв’язання цього завдання існує F_β-міра. Вона задається формулою (2.3):

$$F_\beta = (\beta^2 + 1)\frac{precision * recall}{(\beta^2 * precision) + recall} \quad (2.3)$$

де, якщо перевага надається точності, то $0 < \beta < 1$, якщо перевага надається повноті, то $\beta > 1$. Якщо $\beta = 1$, то міра називається збалансованою і позначається F_1.

Виділяють два основних підходи при обчисленні F_β-міри для багатоміткових класифікаторів [10]. У підході F_β-micro всі окремі значення TP,

FP і FN підсумовуються, після чого проводиться розрахунок. При F_β-micro усереднення проводиться вже після розрахунку значень F_β-міри для кожного класу. Таким чином, оцінка ефективності F_β-micro рівнозначно інтерпретує всі документи, тобто відображає ефективність для найбільш популярних класів. Оцінка F_β-macro, навпаки, рівною мірою враховує кожну категорію незалежно від того, якою кількістю документів вона представлена у вибірці.

У багатомітковій класифікації перевага надається підходові F1-micro, який використовується при дисбалансі класів. У даному випадку, після приведення міток до бінарного вигляду, кількість 0 стала значно переважати над кількістю 1, тому було вирішено використовувати метрику F1-micro. Оцінка F1 може бути інтерпретована як середньозважене значення точності і повноти, коли оцінка F1 досягає найкращого значення — 1, а найгіршого — 0. Відносний внесок точності й повноти в бал F1 — рівний. Також було додано кілька власних метрик для того, щоб краще оцінити правильність моделі й зрозуміти результат.

Також було прийнято рішення обрати проміжну метрику у вигляді accuracy (точності), оскільки зі збільшенням accuracy буде збільшуватися якість моделі. Для найкращої моделі буде рахуватися більше метрик, що дасть змогу краще оцінити модель і підібрати поріг, який контролює, на скільки відсотків має бути впевнена модель, щоб стверджувати, що такий жанр належить фільму.

На рисунку 2.3 пунктиром зображено поріг, згідно з яким усе, що розміщується лівіше (тобто вірогідність, передбачена моделлю менша за 0,5) буде визнано як відсутність певного жанру щодо фільму, а правіше — присутність. Як видно, що якщо обрати занадто низький поріг, то модель буде передбачувати багато жанрів тексту, навіть якщо вони зайві. Якщо ж обрати занадто високий поріг, то модель буде передбачувати досить мало жанрів і тільки, якщо вона впевнена у їхній наявності. Низьке значення порогу відповідає високому значенню повноти й низькому значенню точності, і навпаки. Це може призвести до того, що певні жанри, до яких належить фільм, не будуть передбачені. Варто зазначити, що, чим більший

перетин площ під червоною і зеленою лініями, тим гірше модель розрізняє наявність жанру. У разі їхнього повного перетину модель буде передбачати випадково, а якщо площа під червоною лінією буде розміщуватися лівіше від площі під зеленою, це означає, що модель плутає відсутність і присутність класу і здебільшого передбачає його відсутність, коли він присутній.

Для оцінки моделей нейронних мереж було також обчислено метрику ROC-AUC (англ. area under receiver operating characteristic curve) [11]. Ця метрика чисельно є площею під кривою залежності True positive від False negative. Тобто, відображає залежність того, як кількість помилково віднесених до «позитивних» класів впливає на кількість правильно віднесених до «позитивних».

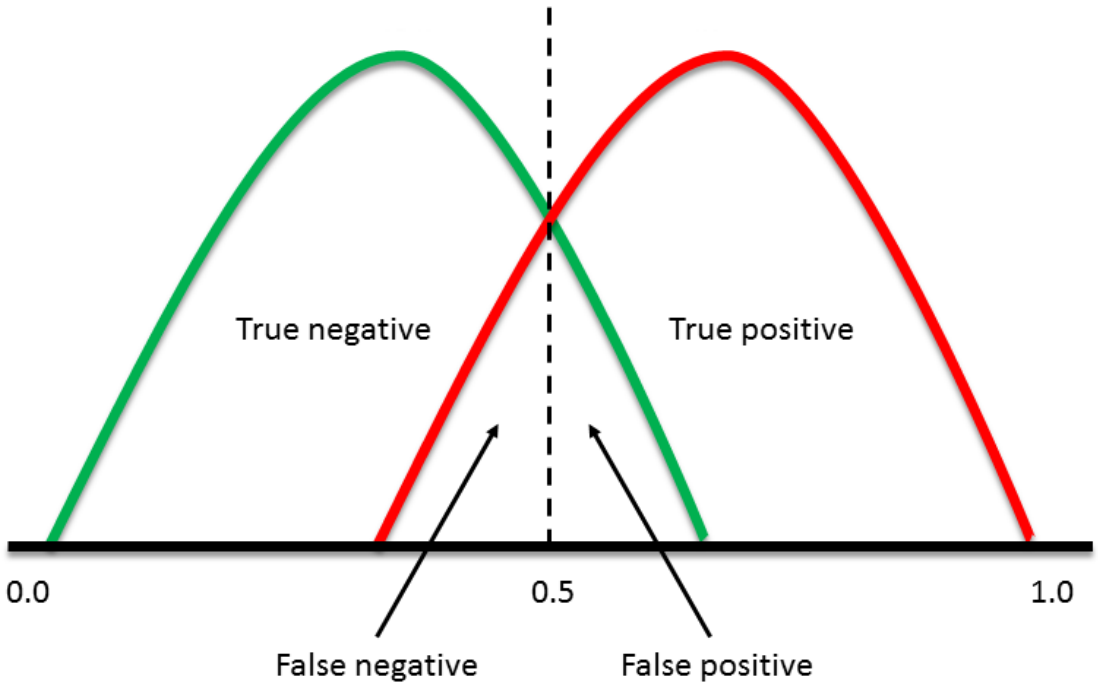

Рисунок 2.3 — Візуалізація проблеми вибору порогу [12]

Максимальним значенням площі під кривою є 1.

Якщо значення метрики дорівнює 0,5, то модель вгадує результат випадково (тобто, у випадку бінарної класифікації, на одну правильну відповідь припадає одна неправильна).

Якщо ж метрика ROC-AUC нижча від 0,5, то модель гірша від випадкового вгадування.

Якщо ж ROC-AUC = 0, то модель ідеально вгадує відсутність класу, але плутає його з присутністю.

Тобто, ROC-AUC вказує на те, на скільки перетинаються площі під червоною і зеленою лініями.

Варто зазначити, що метрика ROC-AUC працює саме з ймовірністю віднесення до певної категорії. Підбираючи поріг, визначається чисельне значення метрики. Важливим є також момент, що ця метрика не може працювати з задачею багатоміткової класифікації і потребує модифікації. Існує кілька підходів, серед яких було обрано micro-averaging [11], що рахує метрику для всіх класів одночасно. Це допомагає впоратися з незбалансованістю класів.

Також було використано метрику Хемінга [12], яка надає додаткову інформацію. Вона базується на відстані Хемінга і обчислюється як частка неправильно спрогнозованих міток, тобто частка неправильних міток від загальної кількості міток.

3. ЗАСОБИ РОЗРОБКИ

Для написання коду системи було використано універсальну високорівневу сучасну мову програмування Python [14], перевагами якої є висока продуктивність програмних рішень і структурований код, який легко читається. Мова Python максимально полегшена, що дає можливість вивчити її за порівняно короткий час. Ядро має дуже зручну структуру, а широкий перелік вбудованих бібліотек дає можливість застосовувати значний набір корисних функцій і можливостей. Ця мова програмування може використовуватися для написання прикладних програм, а також розробки WEB-сервісів. Мова програмування Python може підтримувати широкий перелік стилів розробки додатків, зокрема, дуже зручний для роботи з ООП і функціональним програмуванням. І найголовнішою причиною вибору стала «екосистема» Data Science бібліотек, призначених для Python.

Також було використано TensorFlow — відкриту програмну бібліотеку для машинного навчання, розроблену компанією Google для розв’язання завдань побудови і тренування нейронної мережі з метою автоматичного знаходження та класифікації образів, досягаючи якості людського сприйняття [15]. Вона застосовується як для досліджень, так і для розробки власних продуктів Google. Основний API для роботи з бібліотекою реалізовано для Python, також існують реалізації для C Sharp, C ++, Haskell, Java, Go і Swift. Підтримує роботу з GPU, CPU, TPU (Tensor Processing Unit — це інтегральна схема специфічного застосування (ASIC), призначена для прискорення розрахунків штучного інтелекту, що значно пришвидшують процес навчання). Основою бібліотеки є створення і виконання графу операцій. Це дає змогу використовувати модель, написану однією мовою програмування, відновити з іншої мови програмування, адже граф операцій буде однаковим скрізь.

Бібліотека TensorFlow є досить низькорівневою, тому для спрощення процесу написання нейронних мереж було використано Keras — відкриту нейромережеву бібліотеку, написану мовою програмування Python. Вона є надбудовою над фреймворками Deeplearning4j, TensorFlow і Theano. Націлена на оперативну роботу з мережами глибокого навчання, при цьому спроектована так, щоб бути компактною і модульною. За допомогою окремих класів і методів бібліотеки Keras було проведено обробку даних і реалізовано кінцеву нейронну мережу.

Навчання нейронних мереж вимагає великих обчислювальних потужностей, тому для прискорення процесу навчання й налагодження коду програми було застосовано хмарний сервіс Google Colaboratory, спрямований на спрощення досліджень в галузі машинного і глибокого навчання. За допомогою Colaboratory можна отримати віддалений доступ до машини з підключеною відеокартою, причому абсолютно безкоштовно, що значно пришвидшує процес, особливо, коли доводиться робити глибоке навчання нейронних мереж.

Серед інструментів обробки даних було використано бібліотеку NLTK — пакет бібліотек і програм для символьної і статистичної обробки тексту, написаних мовою програмування Python. Нею користуються розробники, які працюють з комп’ютерною або емпіричною лінгвістикою, когнітивістикою, штучним інтелектом, інформаційним пошуком і машинним навчанням [16].

Для роботи з даними було використано бібліотеку pandas, яка дає можливість зберігати дані у зручному форматі, виконувати обробку, збирати просту статистику.

Під час розв’язання проблеми видалення загальновживаних слів, що не несуть важливої інформації моделі, було використано бібліотеку stop_words, що містить словник зі 180 слів.

Алгоритм передбачення тональності тексту було взято з бібліотеки StanfordCoreNLP, яка пропонує набір інструментів і технологій для роботи з мовою. Він може надавати основні форми слів, їхні частини мови, незалежно, чи це назви компаній, чи людей. Також допомагає нормалізувати дані, час і

числові величини, відмітити структуру речень з точки зору словосполучень і синтаксичних залежностей, вказати, які іменникові фрази стосуються тих самих утворень, визначати настрої, витягувати відношення між згадками суб'єкта, отримувати цитати, про які говорять люди, тощо.

Для передбачення на простих методах машинного навчання було обрано бібліотеку Scikit-learn [17]. Це бібліотека мовою програмування Python з відкритим кодом. З її допомогою можна реалізувати різні алгоритми класифікації, регресії і кластеризації, в тому числі алгоритми SVM, випадкового лісу, k-найближчих сусідів і DBSCAN, побудованих на взаємодії бібліотек NumPy і SciPy з Python.

Перевагами даної бібліотеки є:

— прості та ефективні інструменти для data mining і data analysis;

— зручний доступ до необхідних компонентів;

— те, що вона побудована на основі бібліотек NumPy, SciPy і Matplotlib;

— відкритий програмний код, ліцензія BSD.

Для реалізації клієнтської частини системи було обрано фреймворк Flask — Python для створення веб-додатків, який використовує інструменти Werkzeug, а також шаблонізатор Jinja2. Відноситься до категорії мікрофреймворків — мінімалістичних шаблонів веб-додатків, які надають лише базові функції.

4. ОПИС ПРОГРАМНОЇ РЕАЛІЗАЦІЇ

Класифікація текстів (документів) — завдання комп’ютерної лінгвістики, яке полягає у віднесенні документа до однієї з кількох категорій на підставі його змісту.

4.1. Опис підсистем, вхідні й вихідні дані

У даній системі модель машинного навчання посідає основне місце, оскільки виконує головну бізнес-логіку. Проблема в тому, що різні моделі вимагають різної підготовки даних, а також зручного інтерфейсу, бо сама по собі вона не здатна надавати інформацію у зрозумілій для користувача формі. Основною відмінністю моделей машинного навчання від запрограмованих алгоритмів є те, що, здебільшого, моделі вже є готовими з точки зору програмної реалізації. Тим не менш, вони не можуть бути використані в готовому сервісі як це можна зробити з багатьма модулями та розширеннями для звичайних програмованих алгоритмів. Моделі вимагають процесу налаштування і навчання на даних, тому у системі присутня лише вже навчена модель. Варто зазначити, що процес навчання вимагає не тільки застосування певних методів моделі для визначення нею певних особливостей даних і фіксації їх у внутрішніх параметрах, а й підбору певних параметрів, які впливають на сам процес навчання. Такі параметри називаються гіперпараметрами. Складність машинного навчання полягає в тому, що, крім вибору оптимальної моделі, необхідно також правильно підібрати гіперпараметри, оскільки від них залежить ефективність кожної з них. Не існує чітко заданих способів підбору тих чи інших гіперпараметрів. Здебільшого

інженери користуються евристикою та підбором, що у багатьох випадках зумовлює високі вимоги до потужності обчислювальної інфраструктури.

Визначення тональності тексту є побічною задачею, оскільки у даній системі вона несе додаткову інформацію про сюжет.

Інтерфейс представляє собою веб-сайт, що є посередником між користувачем і алгоритмом машинного навчання. Інформація від користувача оброблюється системою попередньої підготовки даних, що була сформована й випробувана під час процесу навчання моделі, після чого передається завчасно навченій моделі, результати якої також підлягають обробці для формування зрозумілої відповіді.

Системи машинного навчання вимагають чіткої структури і формату вхідних даних. Такі системи здатні опрацьовувати тільки чисельні формати, до того ж вони є вкрай чутливими до шумів і трансформації даних. Це змушує детально аналізувати інформацію й використовувати спеціальні методи для очищення, обробки та перетворення тексту в чисельні вектори. Слід також зазначити, що від способу обробки даних можуть залежати як гіперпараметри моделі, так і вибір самої моделі чи класу моделей. Так, наприклад, дерева рішень і методи, засновані на них, є чутливими до аномальних значень і різного масштабу змінних. Лінійні ж моделі є досить чутливими до різного масштабу, а також аномальних значень, особливо, якщо функцією помилки є MSE (англ. minimum squared error)[18]. Причиною цьому є те, що функція помилки MSE враховує помилку в квадраті, а отже, якщо певне значення сильно відрізняється від інших, то це може досить сильно вплинути на роботу моделі. Те ж саме стосується і ситуації, коли одна зі змінних має значно більший масштаб, оскільки модель буде здебільшого орієнтуватися на неї, як на найбільш важливу.

На вхід програмне забезпечення отримує текстову інформацію, розміри якої не обмежені, оскільки внутрішні алгоритми перетворюють дані у векторну форму, виділяють найбільш значущі слова і обрізають текст до конкретно заданої кількості слів. Після трансформації вектор потрапляє до моделі

машинного навчання, що вже за попередньо навченою схемою виділяє залежність певних слів з мітками — жанрами. Як результат, система повертає 10 значень ймовірностей належності до однієї з міток. Повертається саме така кількість відповідей, бо система пропонує свої варіанти серед 10 жанрів. Таким чином, вихідними даними є ймовірності, з яких необхідно обрати найбільш значущі та вивести користувачеві їхні назви (жанр).

Для візуалізації вищесказаного слід розглянути діаграму IDEF0, подану на рисунку 4.1.

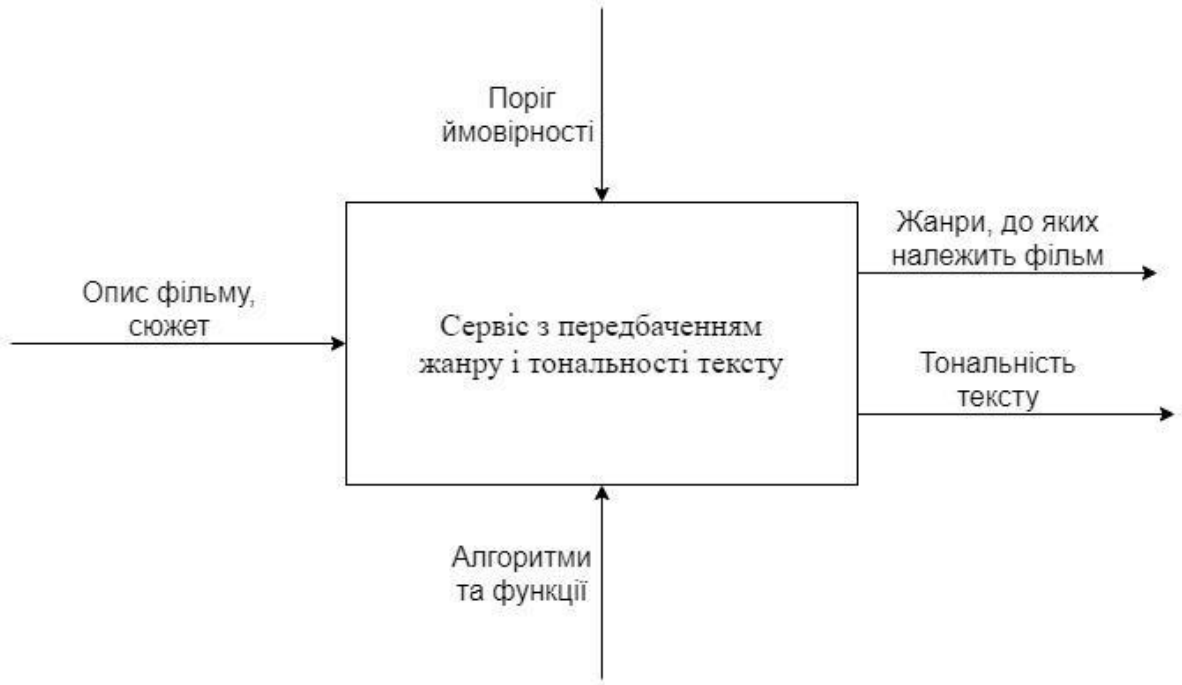

Рисунок 4.1 — Схема інформаційних потоків програмної системи

Будучи під управлінням, у даному випадку, порогу ймовірності жанру відноситись до фільму, система перетворює вхідну текстову інформацію у вихідну (масив жанрів і тональність), використовуючи механізми — алгоритми й функції машинного навчання.

4.2 Аналіз та обробка даних, алгоритм визначення тональності

Першим і одним з найважливіших етапів побудови будь-якої системи класифікації є попередній аналіз та обробка інформації, на основі якої навчатиметься модель.

На основі даних з аналізу стають зрозумілими подальші кроки обробки, а також формуються гіпотези, які необхідно в подальшому перевірити.

Дані було взято з Інтернет-сайту «kaggle.com» [19], призначеного для Data Science-розробників, де проводять змагання з машинного навчання та діляться досвідом розв’язування відповідних проблем. У даних міститься інформація про 34886 кінострічок англійською мовою з детальним описом фільму: сюжет, назва, рік випуску, ім’я режисера та жанри.

Жанрів у фільму може бути кілька або взагалі поле жанру може бути порожнім. Тому першим кроком обробки даних стало видалення об’єктів з порожніми жанрами або сюжетами. Саме на основі цих двох властивостей фільму буде побудована модель машинного навчання, тому зайву інформацію видалено, оскільки, якщо враховувати, наприклад, ім’я режисера, модель може завчити, що цей автор здебільшого випускає фільми певного жанру, що і відбувається реально. Це сприяє перенавчанню моделі на наявних даних, що спричинить проблему у випадках, коли модель натрапить на нового режисера, якого раніше не бачила. Інші ж колонки є просто малоінформативними. Після очистки даних залишилося 28775 рядків.

Наступним кроком обробки даних сюжетів стало видалення зайвих символів і слів, які не несуть смислового навантаження, тому їх користь і роль для класифікації не суттєва і навіть шкідлива.

У деяких випадках модель сама здатна виділити й не враховувати зайві слова, але це вимагає великих обчислювальних потужностей, в інших випадках такі слова будуть просто зменшувати точність. Далі було застосовано зміну

регістру символів до нижнього і проведено стемінг (англ. stemming) — приведення слів до словникової форми з метою скорочення масиву вхідних даних і спрощення моделі [20].

Існує багато підходів до обрізання слова, одним з яких є SnowballStemmer з бібліотеки nltk. Цей метод не ґрунтується на заздалегідь заданому словнику основ слів, а використовує лише заздалегідь задані «знання», правила про конструкції мови, послідовно використовуючи котрі він «обрізає» слово. Для кожної частини мови складається список суфіксів і закінчень (наявність деякого закінчення, наприклад, може накладати обмеження на можливі суфікси). Звісно, в роботі подібного алгоритму трапляються помилки, коли з сайту не видаляються явно зайві частини слова, або ж навпаки, видаляється частина його основи. Однак подібне трапляється не дуже часто. На рисунку 4.2 наведено приклад даних для тренування до стемінгу і після.

№1	film opens two bandits breaking railroad
	film open two bandit break railroad
№2	film family move suburbs hoping quiet life
	film famili move suburb hope quiet life
№3	film features train traveling rockies hold
	film featur train travel rocki hold creat

Рисунок 4.2 — Порівняння слів до і після стемінгу

Популярні слова в даних сюжету подано на рисунку 4.3. Усі кроки, наведені вище, направлені на зменшення розмірності вхідних даних, щоб модель орієнтувалася не на наявність чи відсутність певного слова в певній формі, а розуміла семантичний зміст, оскільки різні закінчення в однокореневих словах роблять їх різними для моделі, і вона ніяк не може провести між ними зв'язок.

Рисунок 4.3 — Візуалізація найуживаніших слів у сюжетах

До жанрів було застосовано методи очистки від зайвих символів, розбиття фраз на слова. Під час аналізу було виявлено, що в даних є багато помилкових і вузькоспеціалізованих жанрів. До помилкових також відносилися назви, написані іншими мовами, крім англійської. Варто також зазначити, що кількість жанрів експоненційно збільшує кількість необхідних обчислень для деяких моделей, тому було прийнято рішення обмежити кількість жанрів з 644 до 10 найпопулярніших (рисунок 4.4).

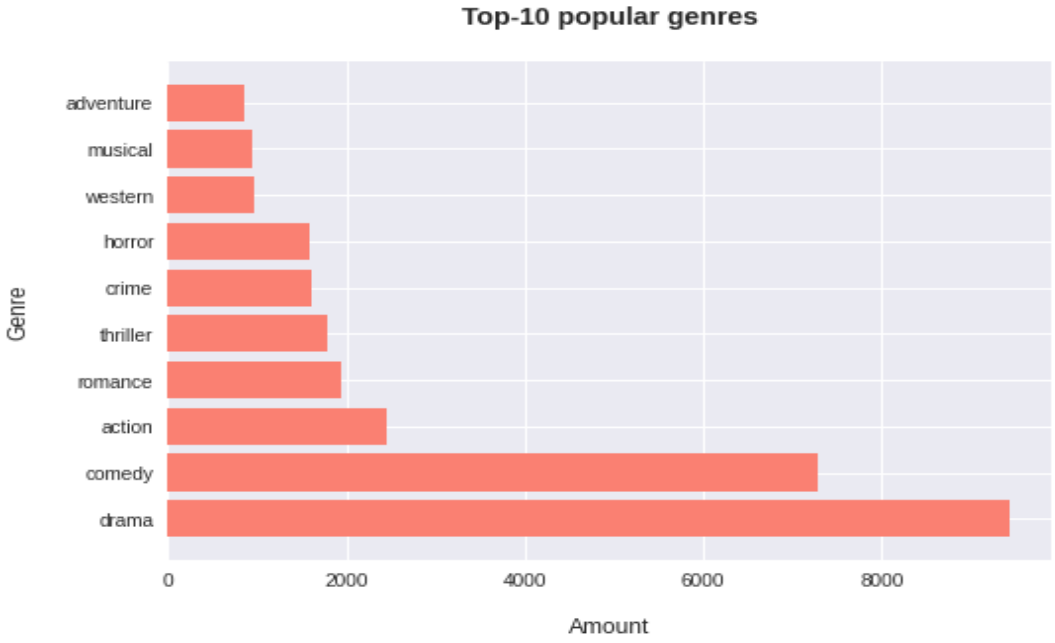

Рисунок 4.4 — Графік уживаності 10 популярних жанрів

Серед цих 10 жанрів у свою чергу найпопулярнішими виявилися «drama» і «comedy», а найменш популярними — «western», «mysical» і «adventure». Розподіл кількості жанрів на фільм подано на рисунку 4.5.

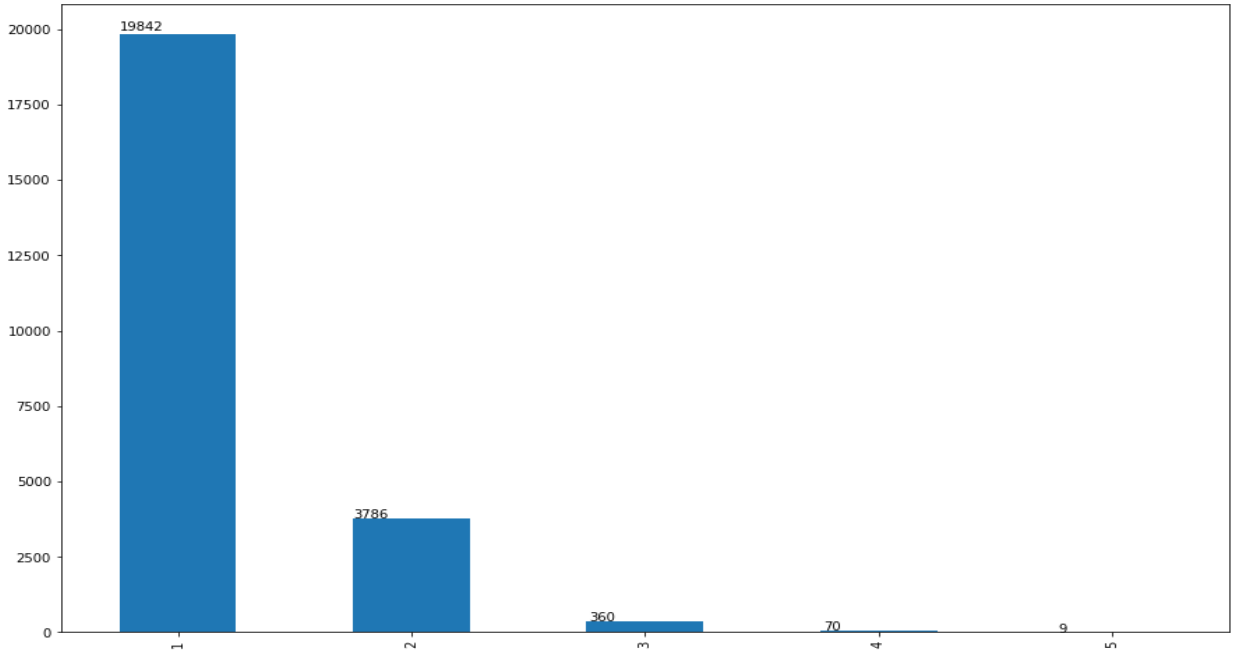

Рисунок 4.5 — Розподіл кількості жанрів на фільм

Останнім кроком обробки стало кодування текстових даних у числові, оскільки нейронна мережа, яка виконує класифікацію, призначена тільки для векторів чисел. Було обрано метод кодування TF — IDF (англ. Term Frequency — Inverse Document Frequency), який нормалізує частоту лексем в документі з урахуванням їхнього вмісту в іншому об’єкті за формулою (4.1) [21].

$$TF - IDF = \frac{n_i}{\sum_k n_k} * \log \frac{|D|}{|(d_i \supset t_i)|} \qquad (4.1)$$

де n_i —кількість входжень певного слова в документ; сума в знаменнику — загальна кількість слів у документі; $|D|$ — кількість документів колекції; $|(d_i \supset t_i)|$ — кількість документів, в яких трапляється слово t_i (коли $n_i \neq 0$).

Цей метод надає перевагу тим термінам, які відповідають конкретному екземпляру, як показано на рисунку 4.6, де лексема «studio» має найвищу релевантність для даного документа, тому що з'являється тільки в ньому. Показник TF — IDF буде вище, якщо певне слово з великою частотою використовується в конкретному тексті, але рідко в інших документах.

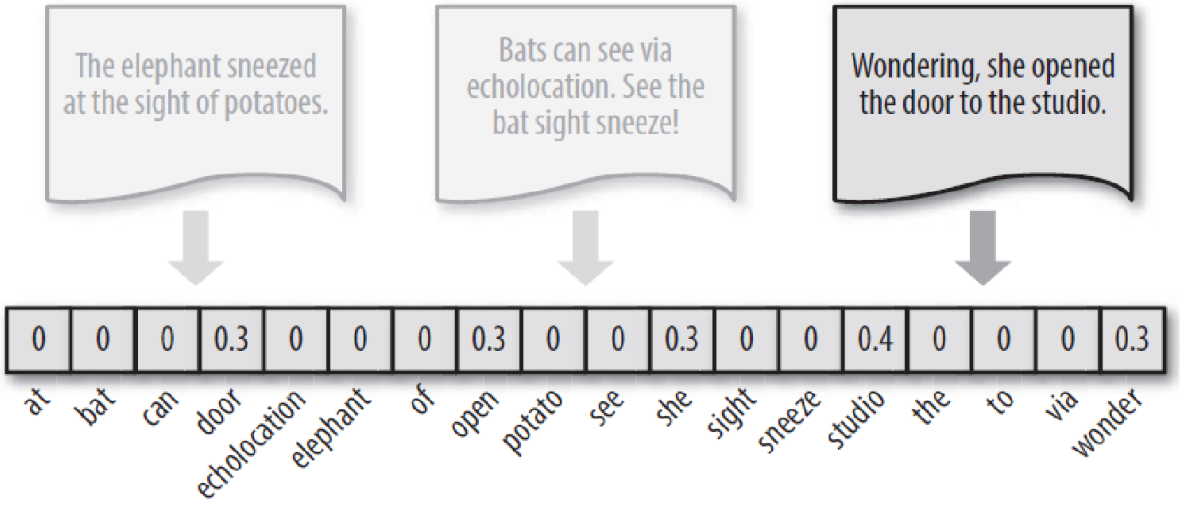

Рисунок 4.6 — Метод кодування TF — IDF

Для трансформації жанрів було використано метод MultiLabelBinarizer з пакету sklearn, який перетворює колонку з мітками в бінарний вигляд, тобто створює для кожної мітки окрему колонку, і якщо мітка присутня в об'єкті, — присвоює 1, інакше — 0.

4.3 Використання і порівняння базових підходів класифікації

Аналізуючи підходи інших розробників, розглянуті в розділі 2, було вирішено порівняти описані там методи на прикладі своїх даних. Слід зазначити, що ці підходи працюють як «обгортка» над основною моделлю, тому було обрано LogisticRegression [22]. Це статистична функція, яка використовується для прогнозування ймовірності виникнення деякої події

шляхом її порівняння з логістичною кривою. Ця регресія видає відповідь у вигляді ймовірності бінарної події (1 або 0).

Порівняння проводилися серед таких підходів класифікації: Binary Relevance, Classifier Chains, Label Powerset та RAkelD. У ході роботи з моделями було використано метод GridSearch з пакету sklearn для знаходження найкращого гіперпараметра «C» моделі логістичної регресії. Цей метод перебирає варіанти параметра, який необхідно оптимізувати, і кожен раз тренує модель із новими значеннями. В результаті обирають той варіант, де точність моделі була найвищою. Змінна «C» — параметр, що контролює вплив регуляризації (метод протидії перенавчанню) на модель. Нижче подано формулу (4.2) логістичної регресії з L_2-регуляризацією.

$$y = \delta(\omega^T * X + b + \frac{1}{2 * C} * \omega^T * \omega) \qquad (4.2)$$

де δ — функція «сигмоїда»; ω — вектор з вагами моделі; X — матриця з даними; b — вільний член; C — параметр регуляризації.

На рисунку 4.7 подано графіки залежності метрики F1-micro від гіперпараметра «C». Можна помітити, що на останньому з графіків при дуже малих значеннях «C» значення метрики F1-micro розміщується в широкому діапазоні і точність моделі нестабільна.

Після підбору оптимізованого гіперпараметра було обчислено значення основних метрик для всіх методів багатоміткової класифікації. Результати подано в таблиці 4.1.

Легко помітити, що метод RAkelD дає найкращі результати на основній метриці F1-micro. Підхід RAkelD генерує випадкові підмножини міток, навчаючи багатомітковий класифікатор на кожній підмножині. Метод Label Powerset, на відміну від попереднього підходу, зіштовхується з меншою кількістю комбінацій, зберігаючи інформацію про кореляцію міток. Метод RAkEL має два суттєві параметри m і k. Перший встановлює кількість

класифікаторів для тренування, а останній — довжину наборів міток, які потрібно генерувати. При k = 1 і m = | L |, RAkEL працює, як метод Binary Relevance, де L — сукупність усіх можливих міток. З іншого боку, при m = 1 і k = | L | отримана модель буде еквівалентна Label Powerset.

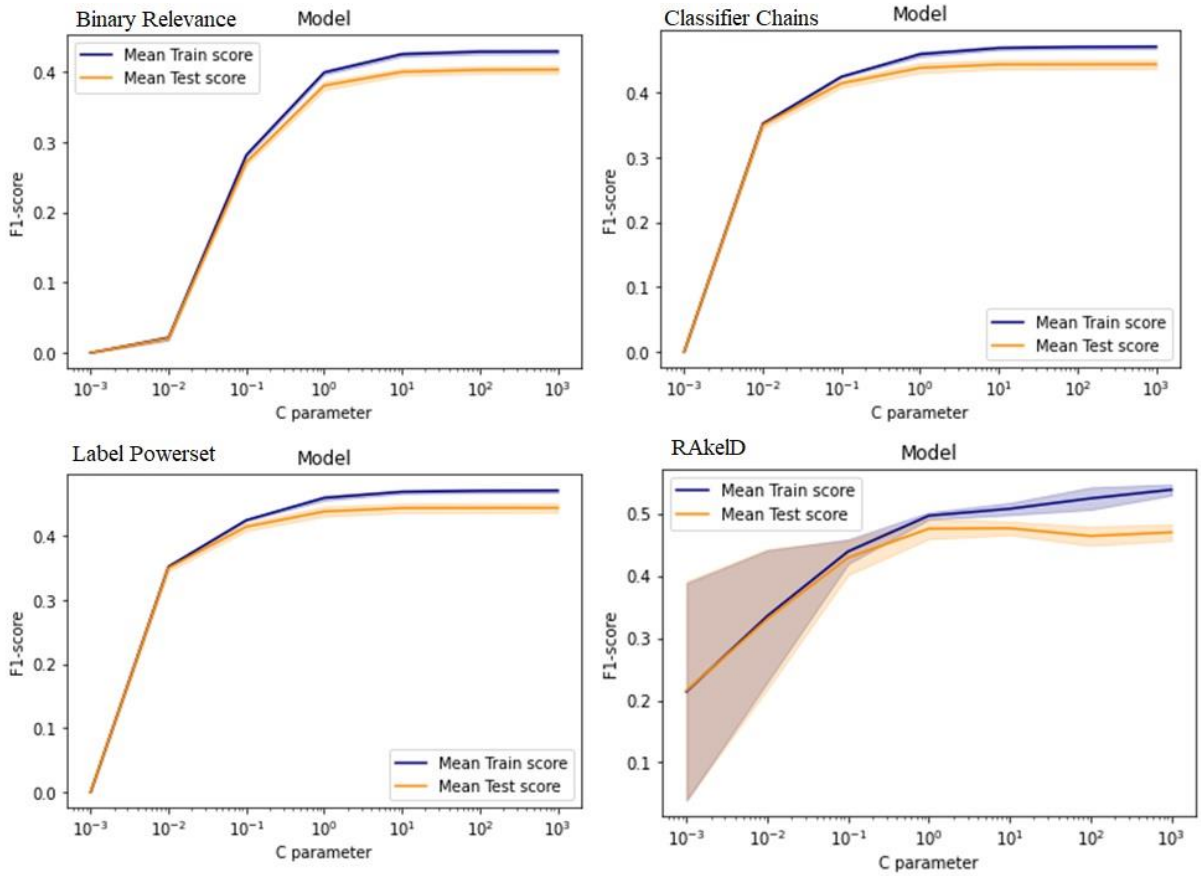

Рисунок 4.7 — Графіки залежності метрики F1 від параметра «C» для методів Binary Relevance, Classifier Chains, Label Powerset і RAkelD

Таблиця 4.1 Оцінка результатів моделей

Метрика	Binary Relevance	Classifier Chains	Label Powerset	RAkelD
F1-micro	0,297	0,264	0,258	0,367
Hamming	0,257	0,210	0,202	0,126
Accuracy	0,059	0,080	0,125	0,226

Використані моделі виявилися занадто простими й нездатними пояснити дані, тому було прийнято рішення використовувати складніші рішення, такі як нейронні мережі.

4.4 Створення і порівняння нейронних мереж

Після аналізу примітивних методів багатоміткової класифікації було вирішено створити нейронну мережу, яка повертатиме 10 значень від 0 до 1 — ймовірностей того, що певний жанр належить тексту. Для покращення результату було також використано техніку «word embedding» — клас прийомів, які дають можливість перетворити слово у вектор, який містить певну семантичну інформацію [23]. Вектори будуються таким чином, що схожі слова будуть розміщуватися поряд у векторному просторі. Існує багато підходів до побудови таких «word embedding». Це методи Word2Vec, GloVe, а також побудова власного словника під час навчання нейронної мережі у першому прихованому шарі. Третій спосіб сповільнює навчання, але у деяких випадках збільшує точність. Було прийнято рішення не тренувати нейронну мережу для побудови «word embedding», а використати ваги вже натренованої методом GloVe мережі, яка тренувалася на 5 мільйонах слів і утворює вектори у стовимірному просторі. Такі вектори зможуть уміщувати більше семантичних зав’язків, які неможливо було б утворити з наявних даних.

Далі було розроблено 7 варіантів нейронних мереж, які відрізняються кількістю вхідних нейронів і шарами. У таблиці 4.2 подано особливості їхньої архітектури.

Таблиця 4.2. Архітектури нейронних мереж

Позначення	Архітектура (назви шарів і кількість вхідних нейронів)
Нейронна мережа №1	InputLayer(217) —> Embedding(217, 100) —> Flatten(217, 100) —> Dense(128) —> Dense(128) —> Dense(10)

Таблиця 4.2. Архітектури нейронних мереж (продовження)

Нейронна мережа №2	InputLayer(217) —> Embedding(217, 100) —> GRU(217, 128) —> LSTM(217, 128) —> LSTM(217, 128) —> Dense(10)
Нейронна мережа №3	InputLayer(217) —> Embedding(217, 100) —> GRU(217, 128) —> LSTM(217, 128) —> Dense(10)
Нейронна мережа №4	InputLayer(217) —> Embedding(217, 100) —> GRU(213, 128) —>Batch(213, 128) —> LSTM(128) —> Dense(10)
Нейронна мережа №5	InputLayer(213) —> Embedding(213, 100) —> LSTM(213, 128) —> LSTM(213, 128) —> LSTM(128) —> Dense(10)
Нейронна мережа №6	InputLayer(213) —> Embedding(213, 100) —> LSTM(500) —> Dense(10)
Нейронна мережа №7	InputLayer(213) —> Embedding(213, 100) —> GRU(264) —> Dense(10)

На рисунку 4.8 подано графіки втрат і точності нейронних мереж.

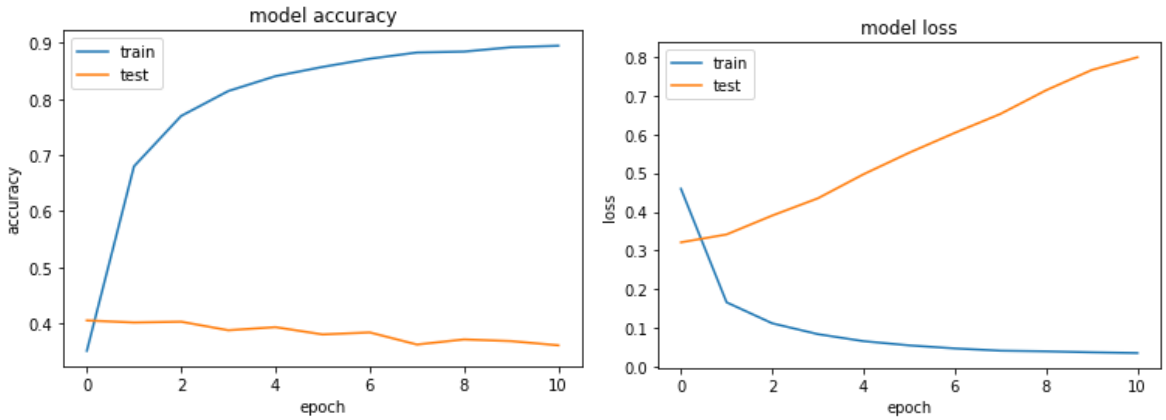

Рисунок 4.8 — Графік функції втрат і точності для нейронної мережі №1

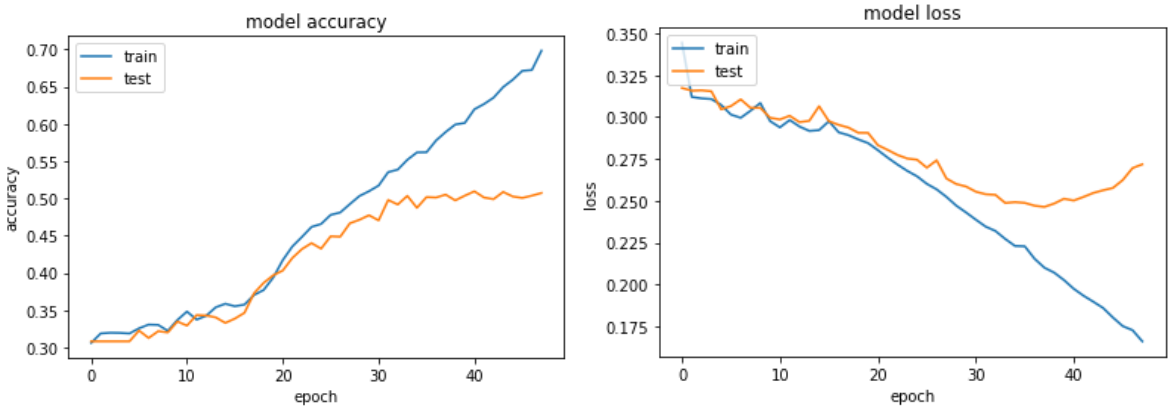

Рисунок 4.9 — Графік функції втрат і точності для нейронної мережі №2

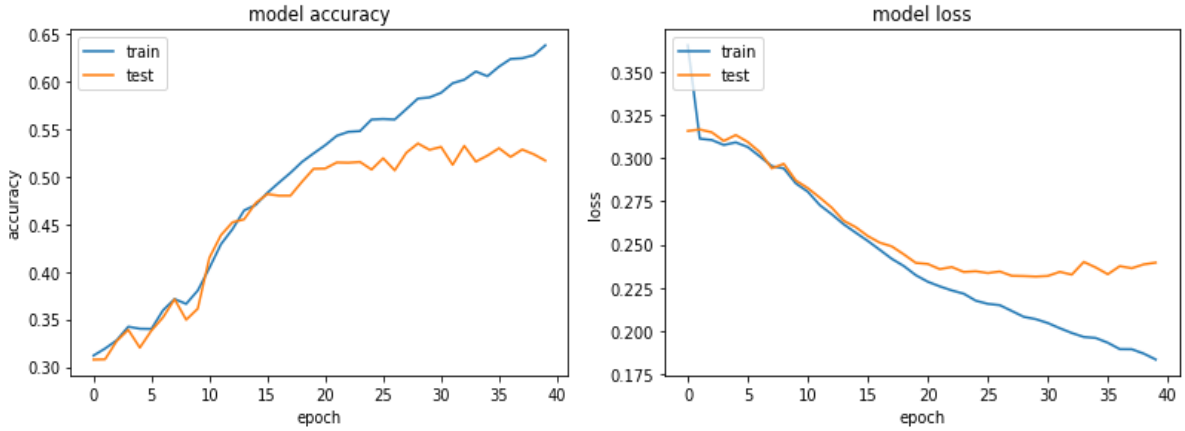

Рисунок 4.10 — Графік функції втрат і точності для нейронної мережі №3

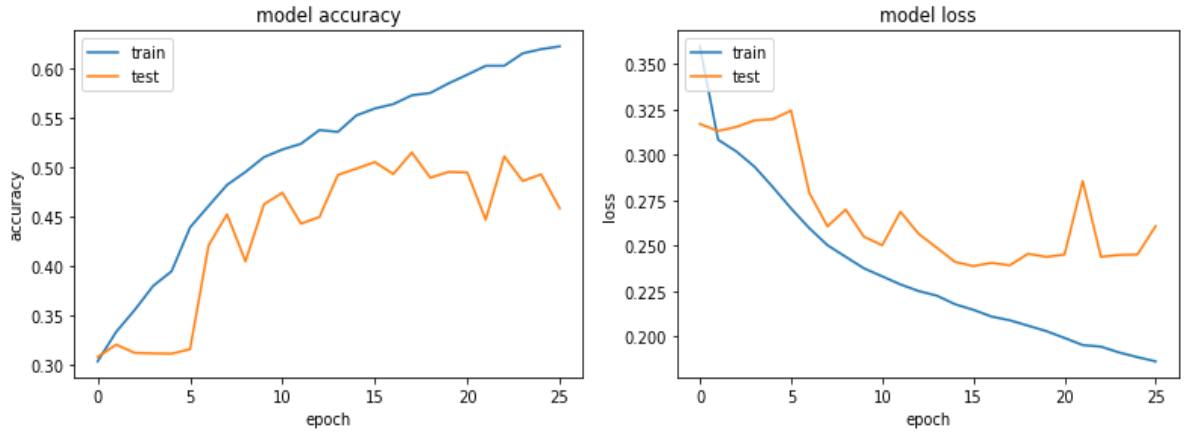

Рисунок 4.11 — Графік функції втрат і точності для нейронної мережі №4

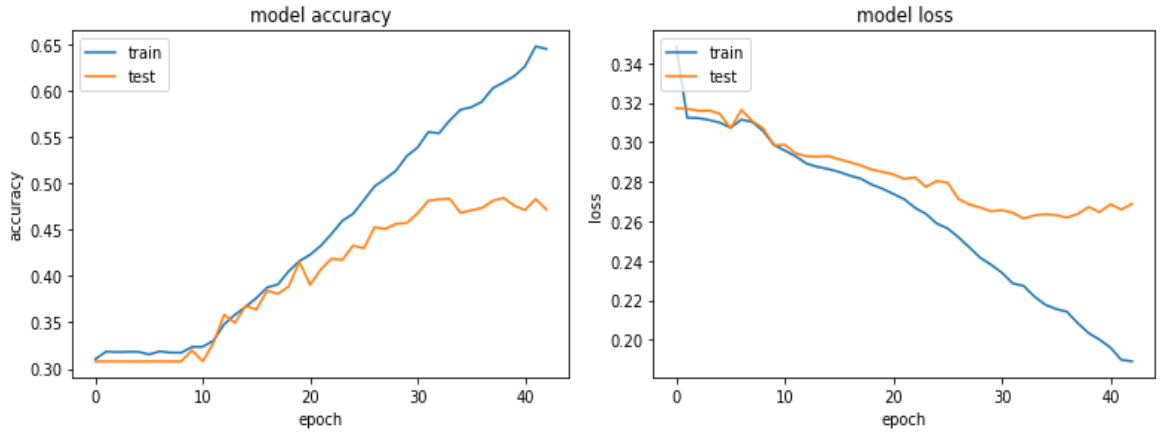

Рисунок 4.12 — Графік функції втрат і точності для нейронної мережі №5

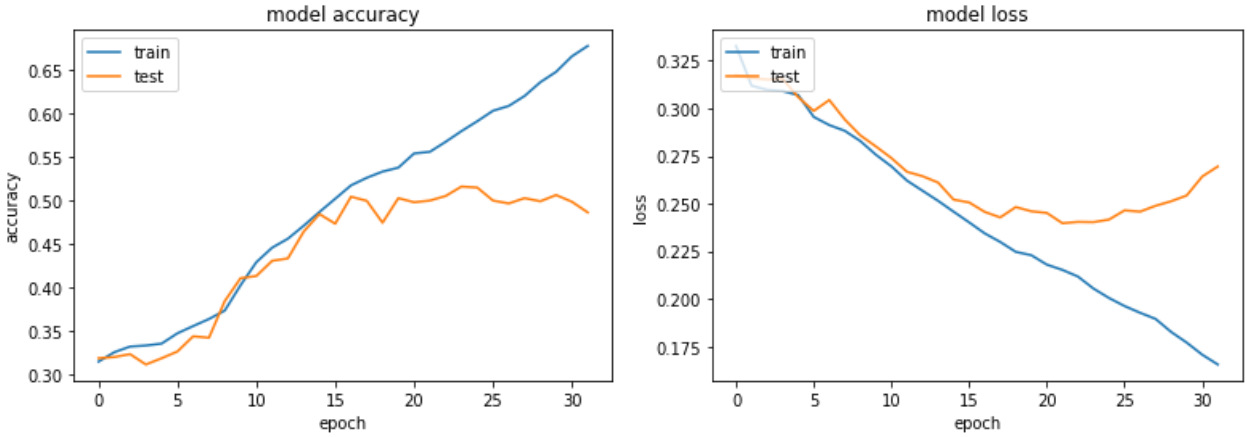

Рисунок 4.13 — Графік функції втрат і точності для нейронної мережі №6

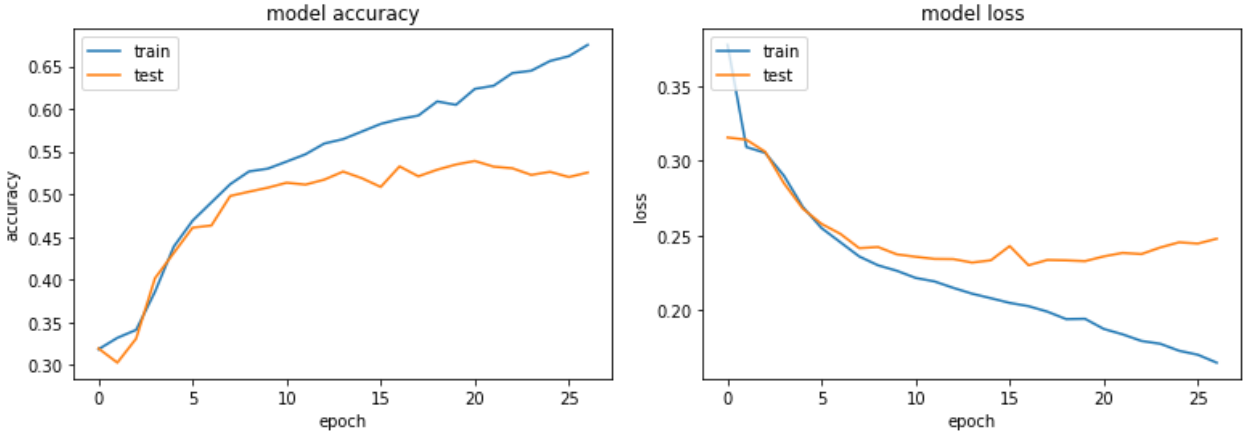

Рисунок 4.14 — Графік функції втрат і точності для нейронної мережі №7

Кожна нейронна мережа складається з шарів. Шар містить сукупність нейронів з єдиними вхідними сигналами [24]. Кількість нейронів у шарі може

бути будь-якою і не залежати від кількості нейронів в інших шарах. У загальному випадку мережа складається з шарів, пронумерованих зліва направо. Зовнішні вхідні сигнали подаються на входи нейронів вхідного шару (його часто нумерують як нульовий), а виходами мережі є вихідні сигнали останнього шару. Крім вхідного і вихідного шарів, в багатошаровій нейронної мережі є один або кілька прихованих шарів. Зв’язки від виходів нейронів деякого шару до входів нейронів наступного шару називаються послідовними.

Слід зазначити, що під час формування архітектури нейронних мереж було використано ідею мінімізації кількості шарів шляхом підвищення кількості нейронів, оскільки складність обчислень при такому підході значно зменшується. Наприклад, якщо нейронна мережа містить два внутрішні шари, кожен з яких складається з 6 нейронів (рисунок 4.15), то тільки між двома прихованими шарами кількість обчислень становила б 36 (6*6=36) кроків. При додаванні ще одного шару з 6 нейронів складність збільшилася б ще в 6 разів, що становило б 216 кроків. У такому випадку треба просто додати більше нейронів в один із шарів, наприклад, в другий. У результаті, кількість обчислень становила б 72 (6*12=72) кроки.

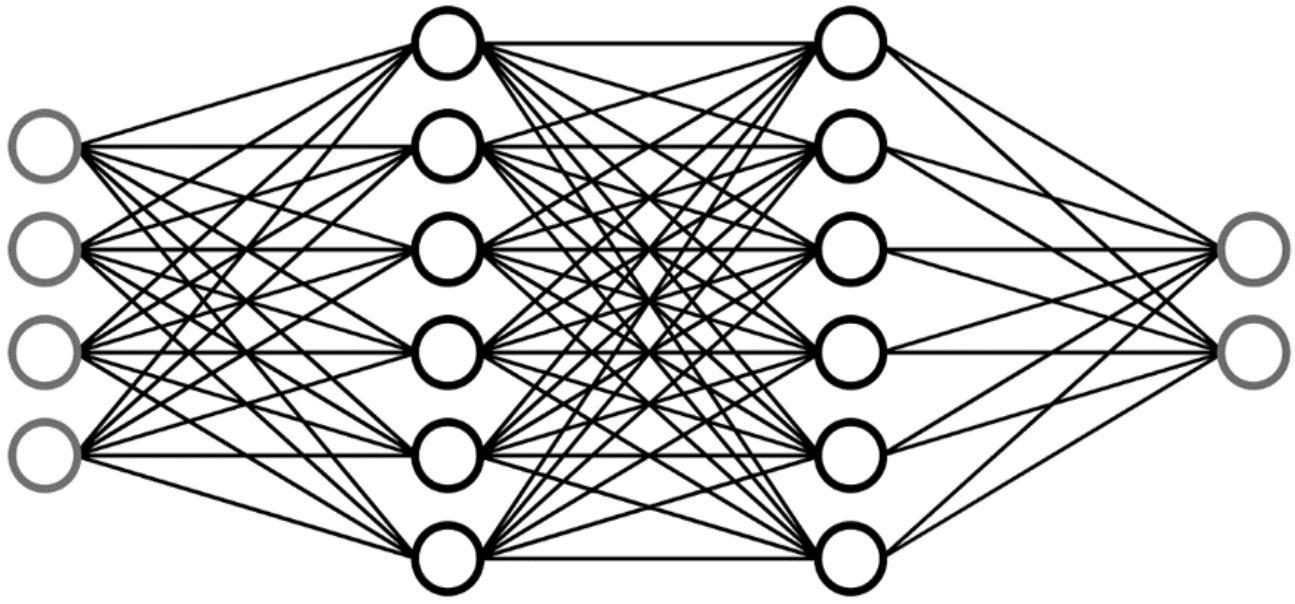

Рисунок 4.15 — Нейронна мережа з двома прихованими шарами [24]

Однією з проблем під час тренування нейронних мереж стало перенавчання — небажане явище, коли ймовірність помилки навченого алгоритму на об’єктах тестової вибірки виявляється істотно вище, ніж середня помилка на навчальній вибірці [25]. Під час тренування нейронних мереж було використано функцію Early Stopping з пакету keras, що дає можливість уникати перенавчання і зупиняє процес тренування тоді, коли втрати на тестових даних починають збільшуватися (рисунок 4.16).

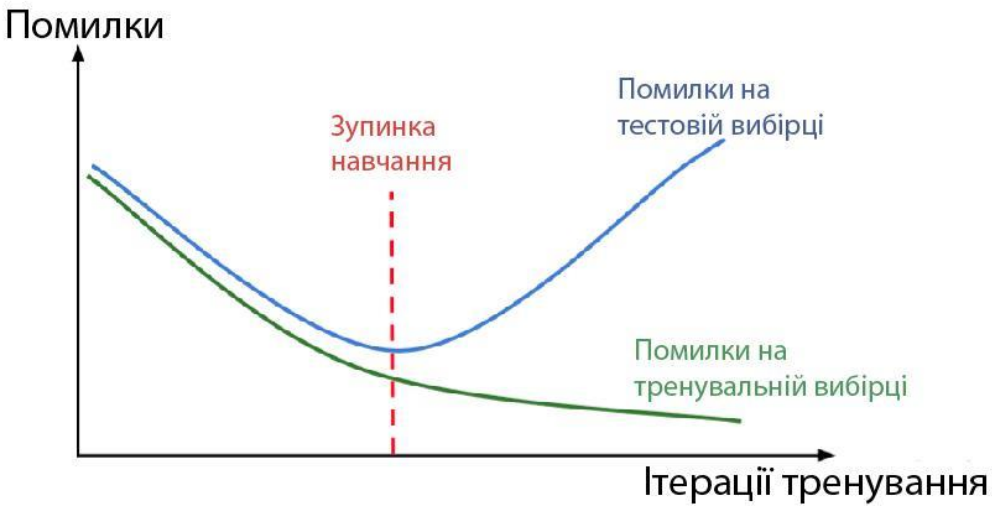

Рисунок 4.16 — Крива помилок

Даний метод має багато параметрів, які дають можливість налаштовувати процес зупинки. Одним з них є «monitor», що слідкуватиме за зміною тієї метрики, яку вказано. Було встановлено значення «val_loss» і Earle Stopping слідкував за зміною значень помилок під час тестування нейронної мережі, оскільки помилки на тренувальній вибірці, як правило, менші через ефект перенавчання, тому можна переоцінити точність отриманої нейронної мережі. Також було використано параметр «patience» = 10, який вказує, що якщо протягом 10 ітерацій якість моделі не покращиться, — треба зупинити навчання.

Результати навчання нейронних мереж подано в таблиці 4.3.

Таблиця 4.3 Результати навчання різних нейронних мереж

	Номер нейронної мережі						
Оцінка	№1	№2	№3	№4	№5	№6	№7
acc	0,8304	0,6291	0,6247	0,5595	0,2033	0,6479	0,6048
roc-micro	0,7779	0,8892	0,9067	0,8846	0,8704	0,9050	0,8879
loss	0,0734	0,1903	0,1877	0,2148	0,6087	0,1770	0,1944
val_acc	0,3914	0,5119	0,5247	0,5053	0,4845	0,5064	0,5349
val_loss	0,4907	0,2504	0,2412	0,2388	0,2672	0,2543	0,2329

Найкращою виявилася нейронна мережа №3 з точністю 0,5247 accuracy та найвищою точністю roc-micro — 0,9067 на тестовій вибірці. Вона складається з кількох шарів (рисунок 4.17).

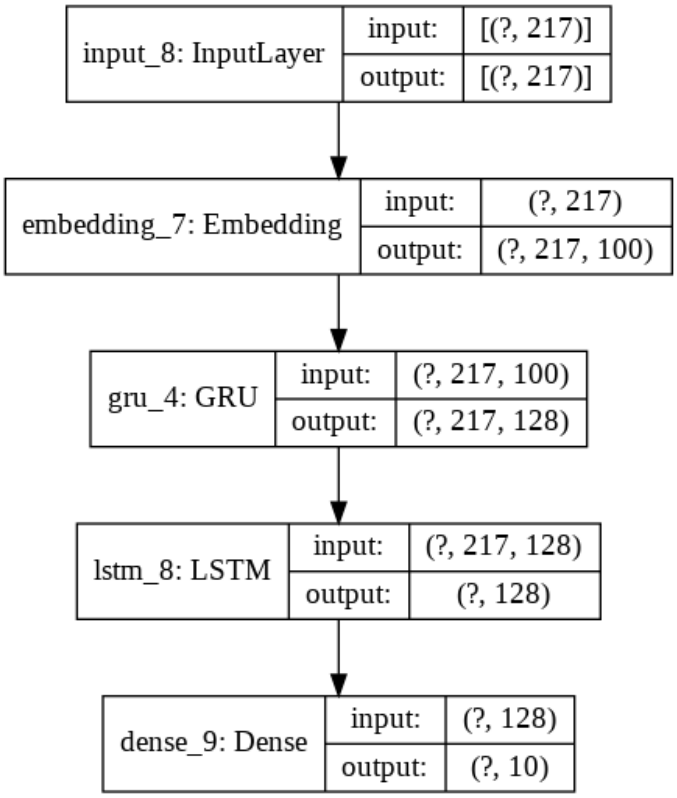

Рисунок 4.17 — Архітектура нейронної мережі №3

Шари InputLayer та Embedding присутні в усіх нейронних мережах. Перший використовується для створення keras-тензору — векторного об’єкта з базовим програмним кодом бібліотек Theano, TensorFlow або CNTK, який доповнено певними атрибутами, що дають можливість будувати модель, знаючи вхідні й вихідні дані моделі. У даному випадку було використано параметр shape = (None, 32,), який вказує, що очікуваний вхід буде партіями 32-вимірних векторів. Елементи цього кортежу можуть бути None (пустими) і їхня форма невідома.

Наступним є Embedding — вбудований шар, який можна використовувати для нейронних мереж на текстових даних. Він вимагає, щоб вхідні дані були закодовані цілими числами, щоб кожне слово було подане унікальним цілим числом. Цей етап підготовки даних було виконано за допомогою API Tokenizer, також наданого Keras. Шар Embedding приймає дані зі словами, ваги яких були вже визначені пакетом GloVe, і перетворює кожне слово на вектор довжиною 100.

Текстові дані — послідовності, у яких кожне наступне слово доповнює попереднє, створюючи для нього контекст, як і кожне речення продовжує те, що було перед ним. Класичні нейронні мережі, хоч і здатні працювати з текстом при його певній обробці, проте не здатні враховувати попередні результати для отримання нових. Для розв’язання цієї проблеми було розроблено цілий клас нейронних мереж для роботи з послідовностями, де кожен наступний шар враховує результат роботи попереднього, а також нові дані. Цей клас має назву RNN (англ. recurrent neural networks), тобто періодичні, або рекурентні нейронні мережі. Схему роботи такої мережі подано на рисунку 4.18.

На рисунку 4.18 у першій лівій секції, відділеній пунктирною лінією, зображено типову нейрону мережу, де прямокутником позначається довільна кількість шарів і вихідний шар. У середній секції зображено типове позначення рекурентної нейронної мережі. Її відмінністю від попередньої є наявність циклу, що означає повторюваність потоку даних. Формально це значить повторюваність самої себе, де кожна наступна копія отримує дані з попередньої.

Детальну схему процесу подано в правій секції рисунка 4.18. Такі мережі обробляють текст поступово, сприймаючи кожне слово як нову точку даних у часі. Вони також можуть обробляти аудіо-інформацію чи будь-яку іншу послідовність. Кінцевим результатом роботи такої мережі є результат останнього кроку обробки. Кожен крок — одношарова нейронна мережа з функцією активації гіперболічного тангенсу. Такі нейронні мережі здебільшого набагато краще обробляють текстову інформацію, проте мають цілий ряд проблем.

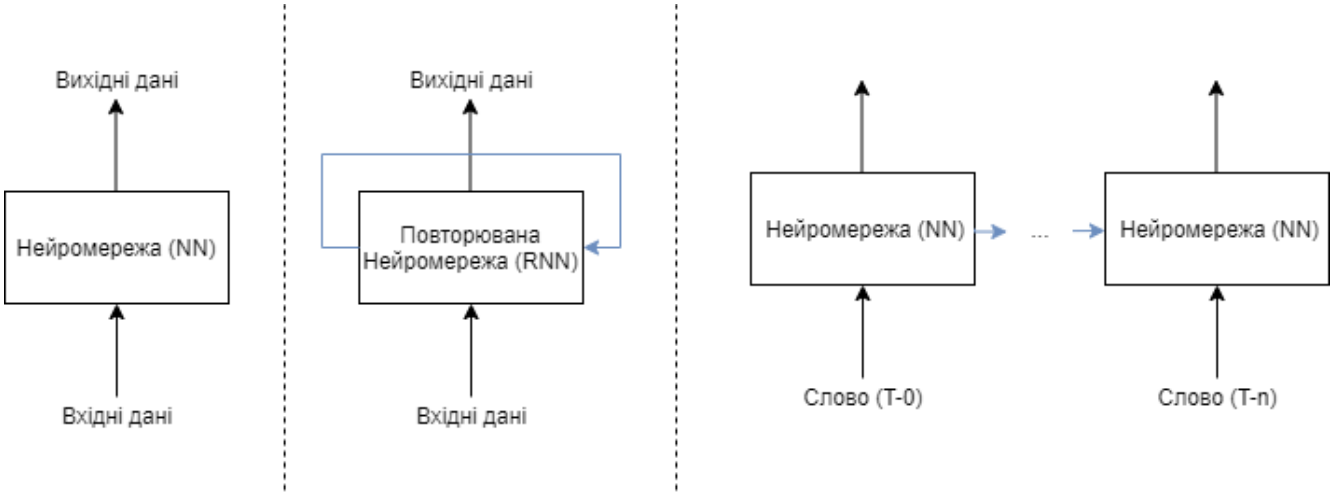

Рисунок 4.18 — Схема порівняння NN і RNN

Інформація, яка передається з попереднього кроку на наступний, реалізує принцип короткої пам’яті (англ. Short term), що є дуже корисним, коли наведено неперервний контекст, тобто послідовно викладена інформація стосується одного об’єкта. Якщо ж наявні контексти, що перетинаються і змінюють один одного, то RNN вже не здатна виокремити всі залежності.

Для розв’язання такої проблеми було розроблено підхід, який реалізує механізм довготривалої-короткотривалої пам’яті (англ. Long-Short Term Memory) і відповідно названу модель, що його реалізує — LSTM.

Вона реалізує іншу архітектуру і має два канали зв’язків між попереднім кроком і наступним:

— канал, який передає результат попереднього кроку;

— канал, який зберігає інформацію з усіх попередніх кроків.

Кожний крок нейронної мережі має вже не один, а чотири шари. Інформація до другого каналу надходить тільки, якщо задовольняються певні умови, а перший передає результат об’єднання вихідних даних мережі та інформації з другого каналу. Такі нейроні мережі демонструють кращі результати порівняно з традиційними RNN.

Шари GRU (англ. Gated recurrent units) і LSTM (англ. Long short-term memory) є принципово важливими, оскільки вони значно впливають на логіку навчання. Шар GRU є вентильним механізмом у рекурентних нейронних мережах [26]. У них на один фільтр менше, ніж у LSTM і вони інакше з’єднані. Фільтри «забування» і входу об’єднуються в один фільтр «оновлення» (англ. Update gate). Цей фільтр визначає, скільки інформації зберегти від останнього стану і скільки інформації отримати від попереднього шару. У більшості випадків GRU працюють так само, як LSTM, найважливіша відмінність у тому, що GRU трохи швидший і простіший в експлуатації, проте має менше можливостей [27].

Кінцевим і спільним для всіх нейронних мереж є шар Dense з 10 нейронами виходу і параметром activation = «sigmoid». Функція активації визначає вихідне значення нейрона залежно від результату зваженої суми входів і порогового значення. Функція сигмоїда є нелінійною за своєю природою, а комбінація таких функцій виробляє теж нелінійну функцію. Вона часто використовується для задач класифікації, через це було обрано саме її.

На рисунку 4.19 подано статистику передбачень у відсотковому форматі для кожного жанру, а на рисунку 4.20 — без підрахунків відсоткового значення.

Бізнес-логіка розроблюваної системи передбачає, що модель буде допомагати реальній людині-оператору обирати жанри, тобто мета — надати користувачу якомога більше жанрів, що справді належать фільму, і при цьому робити якомога менше зайвих передбачень. Тому було обрано поріг 0,35%, при якому здебільшого прогнозується принаймні один жанр (рисунок 4.21).

	action	adventure	comedy	crime	drama	horror	musical	romance	thriller	western
Вгадав правильно	89.78	96.41	71.17	93.67	62.89	93.67	95.94	91.59	92.65	97.29
Передбачив зайве	0.00	0.00	0.10	0.00	0.32	0.05	0.00	0.00	0.00	0.10
Недопередбачив	10.22	3.59	28.74	6.33	36.80	6.28	4.06	8.41	7.35	2.61

Рисунок 4.19 — Статистика класифікації моделі для кожного жанру у відсотках

	action	adventure	comedy	crime	drama	horror	musical	romance	thriller	western
Вгадав правильно	5402	5801	4282	5636	3784	5636	5773	5511	5575	5854
Передбачив зайве	0	0	6	0	19	3	0	0	0	6
Недопередбачив	615	216	1729	381	2214	378	244	506	442	157

Рисунок 4.20 — Статистика класифікації моделі для кожного жанру

	0.00	0.05	0.10	0.15	0.20	0.25	0.30	0.35	0.40	0.45	0.50	0.55	0.60	0.65
hemming	0.119000	0.758000	0.821000	0.851000	0.869000	0.880000	0.888000	0.894000	0.898000	0.900000	0.902000	0.903000	0.904000	0.903000
absolutely_right	0.000000	0.060163	0.145255	0.216221	0.270068	0.316437	0.352834	0.389231	0.410005	0.419811	0.415822	0.396543	0.372445	0.342529
one_error	0.000000	0.265415	0.456041	0.544125	0.605119	0.639688	0.653980	0.653149	0.649493	0.650823	0.662789	0.675918	0.697357	0.722453
f1	0.212471	0.471469	0.524918	0.554837	0.570073	0.578043	0.578039	0.574998	0.566383	0.555052	0.541233	0.517894	0.493986	0.467742

Рисунок 4.21 — Залежність значення метрик від вибраного порогу

На рисунку 4.22 подано розподіл кількості передбачених жанрів на фільм.

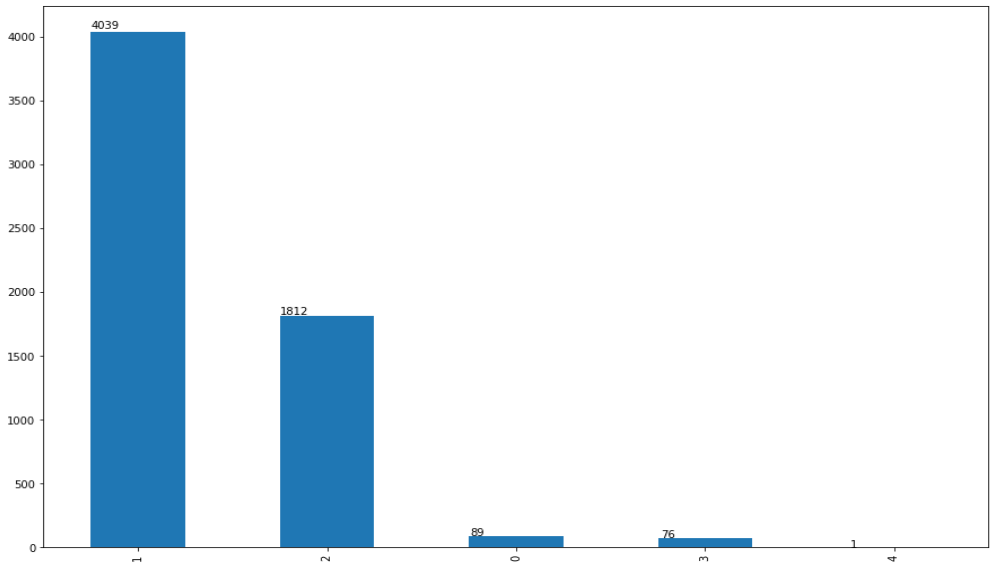

Рисунок 4.22 — Розподіл кількості передбачених жанрів на фільм

При такому порозі на тестовій вибірці було передбачено 0 жанрів (модель не знайшла жодного) для 89 текстів, 1 жанр для 4039, 2 жанри для 1812, 3 жанри для 76 і 4 для 1 тексту (рисунок 4.22).

4.5 Визначення тональності тексту

Під час реалізації системи було вирішено використовувати вже готову модель для передбачення тональності тексту, оскільки, наприклад, для реалізації словникового методу необхідно складати словник тренування для предметної області, де кожному слову призначається значення від -1 до 1, де 1 означає позитивний контекст, а -1 — негативний. Такий підхід, як правило, показує високі результати при визначенні тональності тексту, проте є складними у реалізації. Існують вже сформовані словники, проте основним недоліком їхнього використання є те, що практично завжди значення ваг терміну залежить від обраної предметної області. Наприклад, якщо для вантажного автомобіля термін «важкий» скоріше є позитивно забарвленим, то для гоночного боліда він явно несе негативне забарвлення.

Більшість систем визначення тональності працюють просто, дивлячись на слова ізольовано, даючи позитивні бали за позитивні слова, а негативні — за негативні слова, а потім підсумовують ці значення. Таким чином ігнорується порядок слів і втрачається важлива інформація. Проте модель глибокого навчання, створена дослідниками Стенфордського університету, фактично обчислює тональність на основі того, в якому порядку слова стоять у реченні, та аналізує цілі фрази. Таким чином, модель не так легко обдурити, як попередні моделі. Наприклад, у реченні «Цей фільм насправді був ні таким кумедним, ні дотепним» модель дізналася, що слова «кумедний» і «дотепний» є позитивними, але речення все ж негативне.

Алгоритм працює на основі Recursive Neural Tensor Networks (вид нейронних мереж, які обробляють дані змінної довжини) та Stanford Sentiment Treebank (перший набір даних з повністю розміченими розборами дерев, що дає

можливість провести повний аналіз композиційних ефектів настроїв у мові на основі 215154 фраз у синтаксичному розборі з 11855 речень) [28]. Розробники даного алгоритму стверджують, що точність визначення тональності тексту зросла на 9,7% порівняно зі звичайними словниковими методами.

Процес інсталяції StanfordCoreNLP не такий простий, як у інших бібліотек Python. Бібліотека StanfordCoreNLP написана мовою програмування Java. Тому, щоб інтегрувати цей метод у розроблювану систему, довелося встановити інтерпретатор Java. Після встановлення потрібно було завантажити файли JAR для бібліотеки StanfordCoreNLP. Файл JAR містить моделі, які використовуються для виконання різних завдань у сфері NLP (англ. Natural-language processing — напрям, що вивчає проблеми синтезу природної мови й комп'ютерного аналізу). Наступним кроком став запуск сервера, який обслуговуватиме запити, надіслані обгорткою Python до бібліотеки StanfordCoreNLP. Далі необхідно було викликати метод, який відповідає за визначення тональності тексту, і відправити йому текст. Результатом виконання є слово «Negative» чи «Positive», що свідчить про негативну чи позитивну тональність тексту.

5. РОБОТА КОРИСТУВАЧА З ПРОГРАМНОЮ СИСТЕМОЮ

Для запуску програми треба встановити інтерпретатор python 3. Насамперед, необхідно створити папку, яка інкапсулює всі залежності. Це дасть можливість не «засмічувати» основний python непотрібними бібліотеками. Для цього потрібно виконати такі команди: «python3 -m venv venv», «source venv/bin/activate». Далі необхідно встановити всі залежності командою «pip install -r requirements.txt». Після її виконання можна запускати застосунок, виконавши команду «python3 main.py» і переходити за посиланням http://localhost:5000 — стандартним шляхом для Flask. Якщо порт 5000 буде зайнятий, користувач можете побачити в консолі інтерактивне посилання, яке автоматично відкриє браузер на потрібній сторінці.

Коли користувач переходить на сайт, перед ним з’являється вікно «Передбачення жанрів», куди необхідно вводити текст сюжету фільму. Нижче розміщена кнопка «Аналізувати» (рисунок 5.1.).

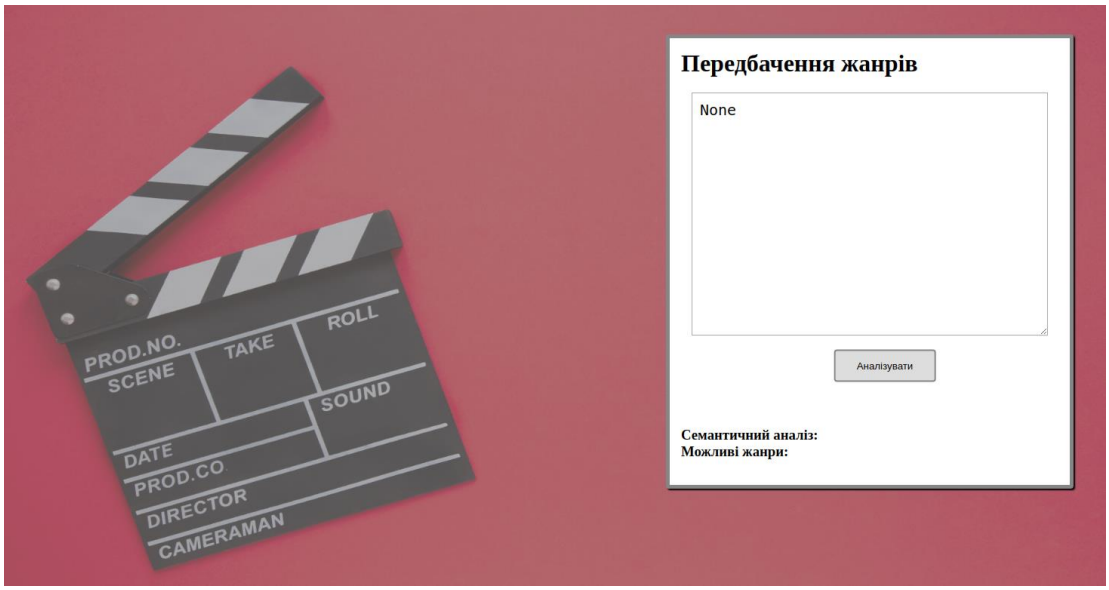

Рисунок 5.1. — Початковий екран користувача

Щоб для введеного користувачем тексту було передбачено жанри і тональність, необхідно натиснути на кнопку «Аналізувати». Як результат, біля поля «Можливі жанри» з’являться теги з жанрами та їхні ймовірності. Поруч з назвою «Семантичний аналіз» відображається веселе чи сумне обличчя залежно від того, позитивний чи негативний настрій має текст (рисунок 5.2).

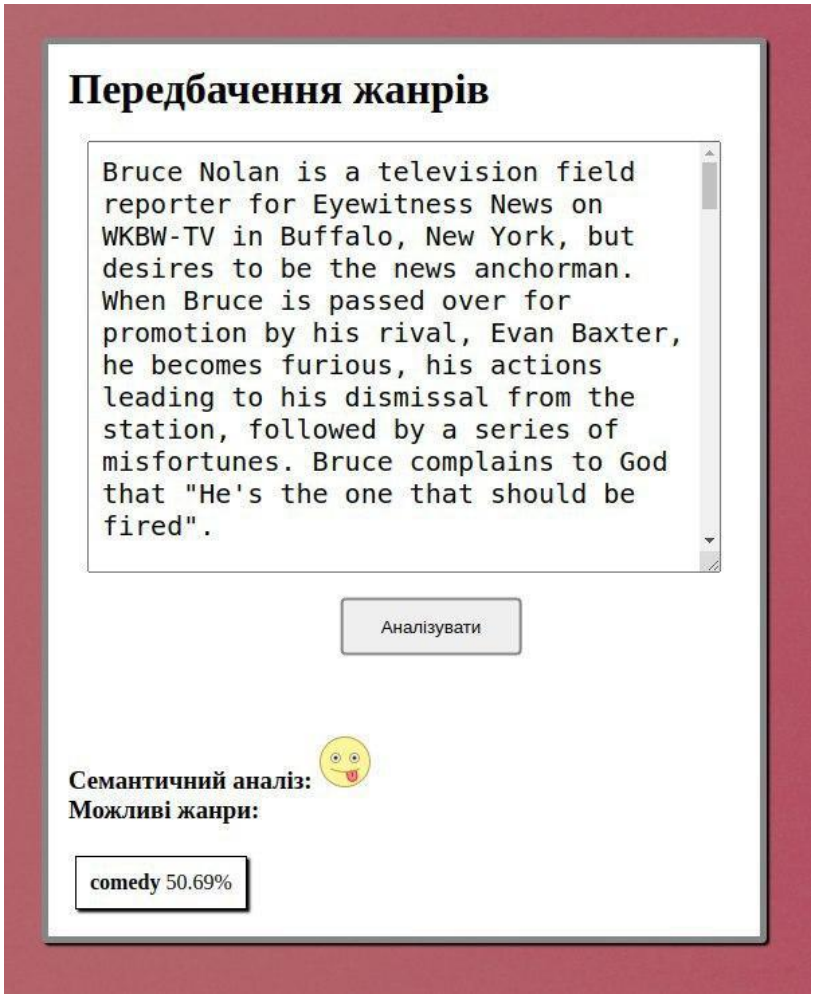

Рисунок 5.2. — Екран з результатами аналізу тексту (приклад 1)

Треба мати на увазі, що семантичний аналіз визначається на основі всього тексту, тому, якщо впродовж всього фільму відбувались хороші події, а кінцівка виявилась сумною, то алгоритм визначить, що фільми позитивний. На рисунку 5.2 наведено приклад передбачення фільму «Брюс всемогутній», що за жанром є комедією. Алгоритм правильно визначив жанр і тональність тексту, а саме,

позитивну тональність і жанр — комедія. Застосунок опрацював текст, який складається з 600 слів, протягом 7 секунд.

На рисунку 5.3 подано приклад передбачення для фільму «Сутінки». Алгоритм визначив сюжет фільму сумним, а жанри — «action» і «thriller».

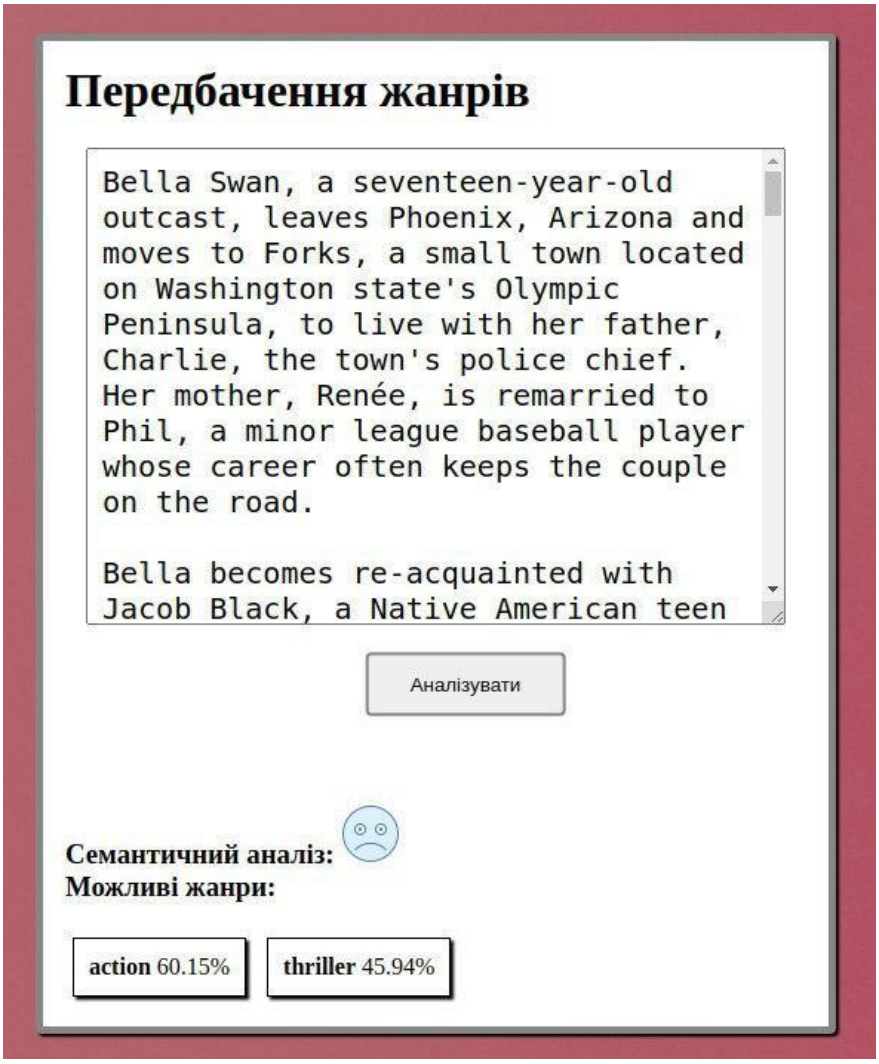

Рисунок 5.3 — Екран з результатами аналізу тексту (приклад 2)

Передбачення жанру і тональності залежать від того, як описати фільм, оскільки можна весело описувати сумну подію і навпаки.

Варто зазначити, що передбачені жанри не є вичерпною відповіддю, а лише рекомендацією оператору. Тобто, спеціаліст повинен перевірити, чи всі з передбачених жанрів відповідають фільмам. Здебільшого оператор буде відкидати зайві, ніж додавати непередбачені.

ВИСНОВКИ

У дипломній роботі розроблено й реалізовано систему класифікації тексту і визначення його тональності.

Було порівняно й проаналізовано різні підходи щодо розв’язання проблеми багатоміткової класифікації, а саме — визначено відмінності між бінарними методами Binary Relevance, Classifier Chains, Label Powerset і RAkelD і визначено метод, який найкраще передбачує жанри на поточних даних. Також було проведено порівняння різних нейронних мереж, пояснення шарів найкращої моделі та обґрунтовано вибір порогу, який визначає, яку мінімальну ймовірність повинен мати жанр для того, щоб можна було стверджувати, що він належить тексту.

У результаті виконання роботи було виконано всі поставлені завдання, а саме:

— проведено аналіз підходів до розв’язання задачі класифікації текстів за жанрами і визначення їхньої тональності;

— досліджено переваги і недоліки існуючих прийомів класифікації, обґрунтовано вибір методу для реалізації;

— реалізовано моделі багатоміткової класифікації;

— застосовано функції з визначення тональності тексту;

— реалізовано інтерфейс і серверну частину.

Результати роботи доповідалися на конференціях «Сучасні проблеми наукового забезпечення енергетики» [21] і «Сучасні аспекти розробки програмного забезпечення».

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. stackoverflow.com [Електронний ресурс] — Режим доступу до ресурсу: https://stackoverflow.com/
2. Taylor J. An Introduction to Error Analysis: The Study of Uncertainties in Physical Measurements / J. Taylor. — University Science Books, 1999. — P. 128-129.
3. Herrera F. Multilabel classification: problem analysis, metrics and techniques / F. Herrera, F. Charte, A. J. Rivera, M. J. del Jesus // Springer International Publishing Switzerland, 2016. — P. 65-78.
4. Godbole S. Discriminative methods formulti—labeled classification / S. Godbole, S. Sarawagi // Adv.Knowl. Discov. Data Mining, 2004. — P. 22-30.
5. Read J. Classifier chains for multi-label classification [Електронний ресурс] / J. Read, B. Pfahringer, G. Holmes, E. Frank // Department of Computer Science, 2011. — P. 5-16. — Режим доступу: https://www.researchgate.net/publication/227319967_Classifier_Chains_for_Multi-label_Classification
6. Boutell M. Learning multi-label scene classification. Pattern Recognition [Електронний ресурс] / M. Boutell, J. Luo, X. Shen, C. Brown// Science Direct, 2004. — P. 1757-1771. — Режим доступу: https://www.rose-hulman.edu/~boutell/publications/boutell04PRmultilabel.pdf
7. Galar M. An overview of ensemble methods for binary classifiers in multi-class problems: Experimental study on one-vs-one and one-vs-all schemes / M. Galar, A. Fernández, E. Barrenechea, H. Bustince, F. Herrera // Pattern Recogn. 44(8), 2011.

8. Tsoumakas G. Random k-Labelsets: an ensemble method for multilabel classification [Електронний ресурс] / G. Tsoumakas, I. Vlahavas — 2007, vol. 4701. — P. 406-417. — Режим доступу: https://link.springer.com/chapter/10.1007/978-3-540-74958-5_38
9. Perruchet P. The exploitation of distributional information in syllable processing / P. Perruchet, R. Peereman // Neurolinguistics, 2004. — P. 97-119.
10. Powers D. Evaluation: From Precision, Recall and F-Score to ROC, Informedness, Markedness & Correlation // Journal of Machine Learning Technologies, 2011. — P. 37-63.
11. Fawcett T. An introduction to ROC analysis / T. Fawcett // Pattern Recognition Letters, 2006. — P. 868-872.
12. Powers D. The Problem of Area Under the Curve // International Conference on Information Science and Technology, 2012. — P. 12-17.
13. Hamming R. Error detecting and error correcting codes / R. Hamming. — Bell System Technical Journal, 1950. — P. 147-160.
14. Deily N. Python Insider / N. Deily. — The Python Core Developers, 2019. — 240 p.
15. Dean J. TensorFlow: Large-scale machine learning on heterogeneous systems./ J. Dean, R. Monga // Google Research, 2015. — P. 19.
16. Bird S. Natural Language Processing with Python. / S. Bird, E. Klein, E. Loper. //O'Reilly Media, 2009. — P. 274.
17. Garreta R. Learning scikit-learn: Machine Learning in Python Paperback / R. Garreta, G. Moncecchi // Packt Publishing, 2013. — P. 183-190.
18. Berger J. 2.4.2 Certain Standard Loss Functions. Statistical Decision Theory and Bayesian Analysis (2nd ed.) / J. Berger. — New York: Springer-Verlag, 1985. —60 p.
19. Kaggle.com [Електронний ресурс] — Режим доступу до ресурсу: https://www.kaggle.com/jrobischon/wikipedia-movie-plots
20. Jongejan B. Automatic training of lemmatization rules that handle morphological changes in pre-, in- and suffixes alike / B. Jongejan, H. Dalianis

// Conference of the 47th Annual Meeting of the Association for Computational Linguistics, 2009. — P. 145-153.

21. Aizawa A. An information-theoretic perspective of tf-idf measures // Information Processing and Management, 2003. — P. 45-65.
22. Tolles J. Logistic Regression Relating Patient Characteristics to Outcomes / J. Tolles,W. Meurer // JAMA, 2016. — P. 533.
23. Qureshi M. EVE: explainable vector based embedding technique using Wikipedia / M. Qureshi. — Journal of Intelligent Information Systems, 2018. — P. 137–165.
24. Schmidhuber J. Deep Learning in Neural Networks: An Overview. — 2015. — P. 28-37.
25. Tetko I. V. Neural network studies. 1. Comparison of Overfitting and Overtraining. / I. Tetko, D. Livingstone, A. Luik, J. Chem. // Inf. Comput. Sci, 1995. — P. 54.
26. Cho K. Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation [Електронний ресурс] / K. Cho, B. van Merrienboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, Y. Bengio. — 2014. — P. 2-3. — Режим доступу: https://arxiv.org/abs/1406.1078
27. Sak H. Long Short-Term Memory recurrent neural network architectures for large scale acoustic modeling / H. Sak, A. Senior, F. — Beaufays, 2018. — P. 14.
28. Socher R. Recursive Deep Models for Semantic Compositionality Over a Sentiment Treebank / R. Socher, A. Perelygin, Y. Jean Wu, J. Chuang, C. Manning, A. Ng, C. Potts // Stanford University, 2013. — P. 1-6.
29. Єрохіна А.О. Web-сервіс з передбаченням жанру фільмів / А.О. Єрохіна, Л.І. Кублій // Сучасні проблеми наукового забезпечення енергетики: Матеріали XVIII Міжнародної науково-практичної конференції молодих вчених і студентів 2020 року. — К.: КПІ ім. Ігоря Сікорського, 2020. — Т. 2. — С. 117.

ДОДАТОК А

Web-сервіс з передбаченням жанру і тональності тексту

Специфікація

УКР.НТУУ“КПІ ім. Ігоря Сікорського”.ТВ6131_20Б

Аркушів 2

2020

Позначення	Найменування	Примітки
Документація		
УКР.НТУУ «КПІ ім. Ігоря Сікорського».ТВ6131_20Б 81-1	Записка	Пояснювальна записка
Компоненти		
УКР.НТУУ «КПІ ім. Ігоря Сікорського».ТВ6131_20Б 12-1	Текст програмного модуля	Текст модуля обробки даних
УКР.НТУУ «КПІ ім. Ігоря Сікорського».ТВ6131_20Б 13-1	Опис програми	

ДОДАТОК Б

Web-сервіс з передбаченням жанру і тональності тексту

Текст програмного модулю

УКР.НТУУ“КПІ ім. Ігоря Сікорського”.ТВ6131_20Б 12-1

Аркушів 8

2020

```
# -*- coding: utf-8 -*-
"""plot.ipynb

Automatically generated by Colaboratory.

Original file is located at

https://colab.research.google.com/drive/1IOcK3DR6oxY9Ct5npaS0OPGyNMV2-eFL
"""

!pip install stop-words
!pip install scikit-multilearn
!pip install wordcloud

import pandas as pd
from stop_words import get_stop_words
from keras.utils import to_categorical


from sklearn.feature_extraction.text import CountVectorizer
from sklearn.feature_extraction.text import TfidfTransformer
from sklearn.linear_model import SGDClassifier
from sklearn.model_selection import GridSearchCV
from sklearn.pipeline import Pipeline, FeatureUnion
from sklearn.preprocessing import MultiLabelBinarizer
from sklearn.model_selection import train_test_split
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.ensemble import RandomForestClassifier
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import train_test_split
from sklearn.base import BaseEstimator, TransformerMixin
from sklearn.preprocessing import LabelBinarizer
from nltk.stem.snowball import SnowballStemmer
```

```
from sklearn.tree import ExtraTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.multiclass import OneVsRestClassifier
from sklearn.metrics import hamming_loss
from sklearn.linear_model import LogisticRegression

import warnings
from sklearn.model_selection import ShuffleSplit


from skmultilearn.problem_transform import BinaryRelevance
from sklearn.naive_bayes import GaussianNB
from skmultilearn.problem_transform import ClassifierChain
from sklearn.linear_model import LogisticRegression
from skmultilearn.problem_transform import LabelPowerset
from skmultilearn.adapt import MLkNN

from sklearn.metrics import f1_score
from sklearn.multiclass import OneVsRestClassifier
from sklearn.metrics import hamming_loss
from sklearn.linear_model import LogisticRegression
from sklearn.tree import ExtraTreeClassifier
from sklearn.ensemble import RandomForestClassifier
import numpy as np

from xgboost import XGBClassifier
import re
import matplotlib.pyplot as plt
plt.style.use('seaborn')
import seaborn as sns

from google.colab import drive
drive.mount('/content/gdrive')
```

```
data = pd.read_csv("/content/gdrive/My
Drive/plot/wiki_movie_plots_deduped.csv")

print("Number of rows: {}, columns: {} in the
dataset".format(*data.shape))
data.head()

"""Let`s delete unimportant columns"""

data = data.drop(["Release Year", "Title", "Origin/Ethnicity",
"Director", "Cast", "Wiki Page"], axis = 1)

data.head(5)

print("Genre column contains {} cells with whitespaces and {} null
values ".format(data.Genre.str.isspace().sum(),

data.Genre.isnull().sum()))
print("Plot column contains {} cells with whitespaces and {} null
values".format(data.Plot.str.isspace().sum(),

data.Plot.isnull().sum()))

"""After analysis the data, you need to delete the "unknown" values
and whitespaces in "Genre"."""

data = data.drop(data[data.Genre.str.isspace()].index)
data = data.drop(data[data.Genre == "unknown"].index)

plot = data["Plot"]
label = data["Genre"]

# data.drop(data.index, inplace=True) #delete data in dataframe

"""First we will need to transform each element of labels before
```

```
passing it to the MultiLabelBinarizer. Label needs to be cleaned
from extra characters and separated by comma.  Let's push data
processing into a separate classes."""

class LabelPreprocessing(TransformerMixin):
    def __init__(self, *args, **kwargs):
        pass

    def fit(self,X ,y=None):
        super(LabelPreprocessing, self).fit(X)
        return self

    def transform(self, y, X=None):
        y = y.str.findall(r'([a-zA-Z]{3,})')
        y = y.replace(regex=r'(film)', value=' ')
        return y.values

class PlotPreprocessing(BaseEstimator, TransformerMixin):
    def __init__(self):
        self.REPLACE_BY_SPACE_RE = re.compile('[/(){}\[\]\|@,;]')
        self.BAD_SYMBOLS_RE = re.compile('[^0-9a-z #+_]')
        self.stemmer = SnowballStemmer("english")
        self.stopwords = get_stop_words('en')

    def fit(self, X, y=None, **fit_params):
        return self

    def transform(self, X, y=None, **fit_params):
        X = X.str.lower()
        X = X.map(lambda x: re.sub(self.REPLACE_BY_SPACE_RE," ",x))
        X = X.map(lambda x: re.sub(self.BAD_SYMBOLS_RE," ",x))
        X = X.map(lambda x: re.sub(r'\s+'," ",x))
        X = X.apply(self.del_stopwords)
        print(X)
        X = X.apply(self.stemming)
```

```
        print(X)

        return X.values

    def fit_transform(self,y,X, **fit_params):
        self.fit(X,y)
        return self.transform(X,y)



    def stemming(self, sentence):
      stemSentence = ""
      for word in sentence.split():
        stem = self.stemmer.stem(word)
        stemSentence += stem
        stemSentence += " "
      stemSentence = stemSentence.strip()
      return stemSentence

    def del_stopwords(self, sentence):

      querywords = sentence.split()
      resultwords  = [word for word in querywords if word not in
self.stopwords]
      result = ' '.join(resultwords)
      return result

import numpy as np
X = PlotPreprocessing().transform(X=plot)
label_preprocessing = LabelPreprocessing().transform(label)

from wordcloud import WordCloud
wordcloud = WordCloud(
background_color="white").generate("".join(X))

# Display the generated image:
```

```
# the matplotlib way:
plt.imshow(wordcloud, interpolation='bilinear')
plt.axis("off")
plt.show()

"""We need to delete some unpopular genres for improving
classification"""

genres = {}
for row in label_preprocessing:
  for item in row:
    genres[item] = 0
for row in label_preprocessing:
  for item in row:
    genres[item] +=1



top_genres = list(map(lambda x: x[0], sorted(genres.items(),
key=lambda x: x[1], reverse=True)[:10]))
amount_ofeach_genre = list(map(lambda x: x[1],
sorted(genres.items(), key=lambda x: x[1], reverse=True)[:10]))
y_pos = np.arange(len(amount_ofeach_genre))

# visualisation
plt.barh(y_pos, amount_ofeach_genre, color='salmon')
plt.yticks(y_pos, top_genres)
plt.xlabel('\nAmount', fontsize=12)
plt.ylabel('Genre\n', fontsize=12)
plt.title('\nTop-10 popular genres\n', fontsize=14,
fontweight='bold');

filtered_genres = list(map(lambda genres: [genre for genre in
genres if genre in top_genres], label_preprocessing))

X = X[[x != [] for x in filtered_genres]]
```

```
filtered_genres = [x for x in filtered_genres if x != []]

mlb = MultiLabelBinarizer()
mlb_fit = mlb.fit(filtered_genres)
y = mlb_fit.transform(filtered_genres)

data_dict = {x: y for x, y in zip(mlb.classes_, y.transpose())}
data_dict.update({"X":X})

df = pd.DataFrame(data_dict)
df.to_csv("/content/gdrive/My Drive/plot/prep_data.csv")
```

ДОДАТОК В

Web-сервіс з передбаченням жанру і тональності тексту

Опис програмного модуля

УКР.НТУУ“КПІ ім. Ігоря Сікорського”.ТВ6131_20Б 13-1

Аркушів 4

2020

АНОТАЦІЯ

Метою роботи була розробка функцій з обробки даних. Даний модуль оброблює вхідну інформацію та повертає вже готові дані для тренування моделі. Наявні функції були використані не тільки під час навчання моделей, а й впроваджені в кінцеву систему, тому коли користувач вводить текст на сайті, конвеєр обробки даних отримує цю інформацію, виконує внутрішні перетворення та повертає інформацію у вигляді, прийнятному для використання нейронною мережею. В перспективі код може бути доповнений для покращення процесу обробки даних, що, можливо, покращить якість роботи моделі класифікації.

ЗМІСТ

В.1 ВІДОМОСТІ ПРО ПРОГРАМНИЙ МОДУЛЬ

В.1.1 Загальні відомості

Даний програмний модуль було розроблено мовою програмування Python3.6. Наразі код являє собою набір класів та функцій, що розміщені у модулі. Під час розробки використовувалися лише бібліотеки sklearn, skmultilearn, matplotlib, keras pandas.

В.1.2 Функціональне призначення

Модуль призначений для обробки та підготовки даних до використання моделлю машинного навчання та зберіганню їх у файл. Це дає змогу отримати набір даних значно меншого розміру, що майже готові до використання, що дозволяє швидше транспортувати необхідну інформацію на сторонні сервіси з розробки програмного забезпечення.

В.1.3 Опис логічної структури

Було розроблено модуль, основними класами якого є «LabelPreprocessing» і «PlotPreprocessing», де знаходиться основна логіка процесу обробки даних. У першому з них розміщено функції з обробки жанрів. Після виклику методу «fit_transform», в який передаються дані з жанрами, у класі «LabelPreprocessing» відбувається видалення усіх символів, крім букв, а також за допомогою регулярних виразів видаляється слово «film», яке не несе важливої інформації, проте часто зустрічається в жанрах.

Наступним кроком обробки є виклик методу «fit_transform» у класу «PlotPreprocessing», який створений для обробки текстової інформації, яку класифікуватимуть. Першим кроком стало видалення зайвих символів та цифр

за допомогою регулярних виразів. Далі викликається метод SnowballStemmer з пакету nltk, що перетворює слова до їх початкової словникової форми. Наступним кроком стало видалення слів, що не несуть важливої інформації про конкретний текст. Останнім кроком стало видалення зайвих пробілів.

В.1.4 Використані технічні засоби

Код може бути запущений на будь-якому сучасному комп’ютері та браузері. Характеристики комп’ютера, що використовувався для написання програмного коду: *Intel Core i5*-9400F, CPU 2.1 GHz.

В.1.5 Виклик і завантаження

Програмний модуль написаний мовою Python, тому вимагає встановлення інтерпретатору python 3.6.

В.1.6 Вхідні та вихідні дані

Вхідними даними є файл, що містить 35 тисяч рядків і 7 колонок з різними характеристиками фільму. Вихідними даними ініціалізованого модулю є текстові дані з 24 тисячами рядків оброблених даних що містять жанри та текст.

ДОДАТОК Г

Web-сервіс з передбаченням жанру і тональності тексту

АПРОБАЦІЯ

Міжнародна науково-практична конференція «Сучасні проблеми наукового забезпечення енергетики: Матеріали XVIII Міжнародної науково-практичної конференції молодих вчених і студентів 2020 року», м. Київ, 2020

УКР.НТУУ“КПІ ім. Ігоря Сікорського”.ТВ6124_20Б

Аркушів 3

2020

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ ІМЕНІ ІГОРЯ СІКОРСЬКОГО»

СУЧАСНІ ПРОБЛЕМИ НАУКОВОГО ЗАБЕЗПЕЧЕННЯ ЕНЕРГЕТИКИ

Матеріали XVIII Міжнародної
науково-практичної конференції
молодих вчених і студентів
2020 року

ТОМ 2

Київ- 2020

УДК 004.855.5:004.912

Студент 4 курсу, гр. ТВ-61 Єрохіна А.О.
Доц., к.т.н. Кублій Л.І.

WEB-СЕРВІС З ПЕРЕДБАЧЕННЯМ ЖАНРУ ФІЛЬМІВ

З розвитком і вдосконаленням інформаційних технологій темпи їхнього застосування невпинно зростають. Зокрема, тема машинного навчання останнім часом спричинила величезний сплеск інформації. Згідно зі статистикою Google Trends 2020 року [1] фразу "machine learning" шукають в 10 разів частіше, ніж у 2010 році.

У даному дослідженні використано такі методи машинного навчання, як логістична регресія, GNB, LSTM, RNN [2], проведено їхнє порівняння і оптимізацію параметрів моделей для одержання найточнішої класифікації. Завданням роботи є передбачення жанру фільму за коротким описом його сюжету, тобто за текстовими даними. Складність полягає у тому, що фільм може мати кілька жанрів, що значно ускладнює процес навчання і перевірки точності моделі. Тому за основу було взято багатотемну класифікацію, в якій об'єкт, який класифікується, може належати до кількох класів одночасно, а самі класи є не взаємовиключними (можливо, навіть вкладеними). Як результат, система оцінює, наскільки точно жанр пов'язаний з фільмом і вибирає тільки ті мітки (жанри), міра відповідності яких вища від заданого порогу. Навчання моделей відбувалося на наборі даних, в якому у відповідність до 30886 унікальних текстових описів фільмів ставляться жанри. У наборі налічується 100 унікальних жанрів, найпопулярнішими серед яких є "drama", "comedy" і "horror". Мірою точності моделі вибрано метрику F1 [3], яка є середнім гармонійним точності й повноти. Це гарантує найбільшу кількість визначених фільмів кожного жанру з найменшою помилкою.

Використано рекурентну нейронну мережу, реалізовану в бібліотеці Keras, що дає можливість оцінювати "позитивний", чи "негативний" опис фільму подано системі.

Обробка текстової інформації про фільм передбачає видалення непотрібних символів, поділ тексту на слова, видалення слів, які не несуть важливої інформації, подання всіх символів на нижньому регістрі і проведення лематизації, тобто зведення різних форм одного слова до словникової форми.

Спочатку було випробувано логістичну регресію як одну з простих базових моделей. Цей статистичний метод повертає ймовірність належності певного фільму до певного жанру. Після тренування моделі значення метрики F1 досягло відмітки 37%. Це значення досить мале, тому було обрано ансамбль (використання кількох алгоритмів машинного навчання з метою отримання кращого результату) Majority Rule Voting (GNB + LSTM). У результаті одержано найбільше значення F1-метрики — 53% на тестовій вибірці, яка становила 20% від загального обсягу даних. Для написання модулів системи вибрано мову програмування Python, бібліотеку Keras і фреймворк Django.

Система може бути корисною для сервісів, які надають можливість переглядати кінофільми онлайн. Надаючи інформацію щодо жанру фільму, система також полегшить роботу адміністратора ресурсу. У подальшому дана система може бути використана іншою системою для надання користувачам рекомендацій щодо фільмів.

Перелік посилань:

1. Machine learning — https://trends.google.com/trends/explore?date=all&q=machine%20%20learning
2. Goodfellow I., Bengio Yo., Courville A. Deep Learning. — https://www.deeplearningbook.org/
3. F1 Score. — https://www.ritchieng.com/machinelearning-f1-score/