

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ  
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ ІМЕНІ ІГОРЯ СІКОРСЬКОГО»  
КАФЕДРА ІНФОРМАТИКИ ТА ПРОГРАМНОЇ ІНЖЕНЕРІЇ

**КУРСОВА РОБОТА**

з дисципліни «Аналіз даних в інформаційних системах»

на тему: «Аналіз продажу ігор у магазинах для ліцензійної дистрибуції»

Студента 2 курсу групи ІП-01

Спеціальності: 121

«Інженерія програмного забезпечення»

Галько Міли Вячеславівни

«ПРИЙНЯВ» з оцінкою

---

доц. Ліхоузова Т.А. / доц. Олійник Ю.О.

---

Підпис

Дата

Київ - 2022 рік

Національний технічний університет України “КПІ ім. Ігоря Сікорського”

Кафедра інформатики та програмної інженерії

Дисципліна Аналіз даних в інформаційно-управляючих системах

Спеціальність 121 "Інженерія програмного забезпечення"

Курс 2 Група ІП-01

Семестр 4

## ЗАВДАННЯ

### на курсову роботу студента

Галько Міли Вячеславівни

---

1.Тема роботи Аналіз продажу ігор у магазинах для ліцензійної дистрибуції

---

---

2.Строк здачі студентом закінченої роботи 17.08.2022

---

3. Вхідні дані до роботи методичні вказівки до курсової роботи, обрані дані з сайту

---

<https://www.kaggle.com/datasets/andrewmvd/steam-reviews>

---

4.Зміст розрахунково-пояснювальної записки (перелік питань, які підлягають розробці)

1.Постановка задачі

---

2.Аналіз предметної області

---

3.Інтелектуальний аналіз даних

---

5.Перелік графічного матеріалу ( з точним зазначенням обов’язкових креслень )

---

---

6.Дата видачі завдання 16.04.2022

---

## КАЛЕНДАРНИЙ ПЛАН

№ п/п	Назва етапів курсової роботи	Термін виконання етапів роботи	Підписи керівника, студента
1.	Отримання теми курсової роботи	16.04.2022	
2.	Визначення зовнішніх джерел даних	20.05.2022	
3.	Пошук та вивчення літератури з питань курсової роботи	15.06.2022	
4.	Обґрунтування методів інтелектуального аналізу даних	5.07.2022	
6.	Застосування та порівняння ефективності методів інтелектуального аналізу даних	25.07.2022	
7.	Підготовка пояснювальної записки	10.08.2022	
8.	Здача курсової роботи на перевірку	17.08.2022	
9.	Захист курсової роботи	17.08.2022	

Студент

\_\_\_\_\_  
(підпис)

Галько Міли Вячеславівни

\_\_\_\_\_  
(прізвище, ім'я, по батькові)

Керівник

\_\_\_\_\_  
(підпис)

доц. Ліхоузова Т.А

\_\_\_\_\_  
(прізвище, ім'я, по батькові)

Керівник

\_\_\_\_\_  
(підпис)

доц. Олійник Ю.О.

\_\_\_\_\_  
(прізвище, ім'я, по батькові)

"17" серпня 2022 р.

## АНОТАЦІЯ

Пояснювальна записка до курсової роботи: 43 сторіноки, 16 рисунків, 6 посилань.

Об'єкт дослідження: інтелектуальний аналіз даних.

Предмет дослідження: створення програмного забезпечення, що проводить аналіз користувацьких відгуків ігор та на основі цього формує основні гідності активних продуктів. Додатково для створення більш повної картини, виконується додатковий аналіз даних про саму активність аудиторії – формування градації ігор за кількістю певних гравців у виді графічного відображення результатів.

Мета роботи: реалізація програмного забезпечення на мові Python, що підготовлює дані для їх подальшого аналізу через різні підходи обробки природньої мови.

Дана курсова робота включає в себе: підготовка набору даних, його зменшення; опис аналізу предметної області для подальшого створення програмного забезпечення інтелектуального аналізу даних; аналіз тональності тексту; TF аналіз; графічне відображення результатів аналізу.

ІНТЕЛЕКТУАЛЬНИЙ АНАЛІЗ ДАНИХ, TERM FREQUENCY, NLP, STATISTICAL METHODS, PART-OF-SPEECH TAGGING, TOKENIZATION, LEMMATIZATION, STEMMING, ПОРІВНЯННЯ АЛГОРИТМІВ, SENTIMENT ANALYSIS

## ЗМІСТ

ВСТУП.....	6
1. ПОСТАНОВКА ЗАДАЧІ .....	8
2. АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ.....	10
3. ВІДБІР ТА ПІДГОТОВКА ДАНИХ.....	12
3.1 Загальний аналіз на коректність даних.....	12
3.2 Видалення полів із пустими значеннями .....	14
3.3 Заміна значень у колонці «Оцінка гри».....	14
3.4 Видалення полів з невеликим впливом .....	15
3.5 Видалення некоректних відгуків.....	16
4. СТРУКТУРОВАНА ОБРОБКА ПРИРОДНЬОЇ МОВИ (NLP) .....	17
5. АНАЛІЗ ТОНАЛЬНОСТІ ВІДГУКІВ .....	21
6. ІНТЕЛЕКТУАЛЬНИЙ АНАЛІЗ ДАНИХ.....	23
6.1 Різноманітні графіки.....	23
6.2 Аналіз частоти слів .....	25
ВИСНОВОК.....	30
ПЕРЕЛІК ПОСИЛАНЬ .....	32
ДОДАТОК А ТЕКСТИ ПРОГРАМНОГО КОДУ .....	33

## ВСТУП

Нині все більша частина людей або є активними користувачами відеоігор або вперше пробує один з продуктів. І це не секрет, що індустрія відеоігор набирає обертів через стрімкий зріст аудиторії та їх активності. Особливо, великий внесок зробив тривалий карантин під час коронавірусу та популяризація окремих ігрових продуктів через серіали, та світові кібертурніри. Дану статистику зможе помітити кожен, проглянувши дані з магазину Steam.

Steam – сервіс, що надає послуги з цифрової дистрибуції. Вже на момент 2019 року, кількість користувачів досягла 1 мільярду. Клієнти мають змогу як купляти відеоігри, так і спілкуватися, писати коментарі-відгуки на продукти сервісу, демонструвати свої досягнення в іграх. Враховуючи увесь спектр сервісу, можемо на базі цього отримати вищеперераховані дані для їх подальшого аналізу.

Загалом, ці дані можна використати для прогнозу майбутніх продуктів. В залежності від задоволеності гравців та його оцінки продукту, можна зробити висновки про успішність теперішнього гри та його причини. Отже, отримавши результати цього аналізу, розробники відеоігор зможуть проглянути найпопулярніші продукти для коректування створення вже своїх нових продуктів.

Основним предметом аналізу у даній курсовій роботі стануть коментарі користувачів. По перше, наша мета – зрозуміти ставлення гравців до продукту. По друге, важливою особливістю даного програмного забезпечення є розкриття реальних причин успіху продуктів зі слів у відгуках клієнтів.

Звичайно, основна мета це важливо, але додатково буде проаналізовані й інші аспекти даних. Важливо зрозуміти не тільки ставлення користувачів до продуктів, але й спиратись на обсяг аудиторії. Реальність така, що є ігри, що мають позитивний відгук, але невелику кількість користувачів. І навпаки, продукт може бути дуже популярним і водночас мати багато ненависників. Цей аспект призводить до інформаційного парадокса. Ми маємо на меті створити

якісний продукт, що зможе отримати позитивні коментарі, однак це не стверджує, що у кінцевому результаті гра буде дуже популярною.

Підсумовуючи усе вищесказане, важливо проаналізувати успіх продуктів та дізнатись їх причини, усвідомлюючи дійсне значення слова «успіх». В залежності від того, чи це значить «популярність», чи «позитивний відгук», чи інше; кінцевий результат аналізу буде мати відповідний вигляд.

## 1. ПОСТАНОВКА ЗАДАЧІ

Під час виконання курсової роботи необхідно виконати наступні завдання:

Спочатку необхідно експортувати вхідні дані з файлу формату .csv у DataFrame зміну програмного забезпечення.

Далі, потрібно відкоригувати вміст. Цей етап фільтрації повинен складатися з переведення можливих значень до булевого представлення, очищення тексту та його нормалізація.

Оскільки початковий файл уміщує понад 6 мільйонів полів, то на кожному етапі підготовки текстів, необхідно зменшувати об'єм даних шляхом видалення не цінних полів.

Наступний етап – обробка відгуків. Він є важким зі сторони оперативності для великої кількості даних, тому надалі буде використовуватися лише частина відгуків приблизно у розмірі 50000 полів. Сам процес обробки природної мови припускає приведення слів до токенів для реалізації лематизації.

Підготовлені кінцеві дані формуємо у файл для збереження. І тепер можемо переходити до аналізу. Оскільки маємо справу з коментарями гравців, то проводимо sentiment analysis для визначення настрою відгуків (позитивний, негативний та нейтральний види).

Для визначення ключових слів формуємо словники найуживаніших слів та словників для різних частин мови.

На кінцевому етапі за кількістю відгуків відбираємо найпопулярніші застосунки. Чим більше гравців, тим більше відгуків, тим більш успішною можна вважати проєкт. Отже, для отриманого списку можна провести аналіз через зіставлення властивостей dataset-у із визначеними переліками слів різних груп та відтворити результати у графічному вигляді.

Додатково в процесі обробки даних необхідно проаналізувати закономірності для більшого обсягу інформації. Цю задачу можна реалізувати через побудову графіків без прив'язки до конкретних ігор. Також важливо на етапі роботи із dataset-ом успішних продуктів повторити ту саму роботу для того, щоб порівняти дані.



Програмне забезпечення повинно бути реалізовано мовою програмування Python 3.

Саму курсову роботу необхідно здати до дедлайну (17 серпня).

## 2. АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

Сьогодні деякі люди вже не можуть уявити своє життя без комп'ютерних ігор. Також є й ті, що ніколи не грали або спробували свої сили лише декілька разів. Заохочування потреб усіх цих категорій клієнтів є однією з найважливіших проблем спеціалістів ігрової індустрії.

Кожна компанія цієї направленості зобов'язана аналізувати реакцію гравців на випущений продукт. Оскільки саме від потоку клієнтів буде залежати прибуток компанії й тим самим її стабільність, можливість мати найманих співробітників.

Загалом, можна стверджувати те, що для створення нової успішної гри важливо проаналізувати гідності вже активно-користувацьких застосунків. А далі приступати до реалізації нової гри відтворюючи обрані переваги. Отже, для повної оцінки успішності активних комп'ютерних ігор доречно проаналізувати відгуки самих клієнтів. Онлайн магазин Steam забезпечує нас наступними даними:

- Користувацькі коментарі ігор;
- Оцінка гри користувачем;
- Оцінка коментарів іншими гравцями.

Цих даних буде достатньо для аналізу самого ставлення клієнтів. Детальна увага буде приділена саме аналізування текстів відгуків для визначення як ставлення до застосунку, так і зрозуміння, що саме мотивує користувача стабільно грати. Оцінка гри, своєю чергою, характеризує лише ставлення до продукту. А ось оцінку коментарів іншими гравцями можна більш віднести по характеристиках до відгуків. Оскільки задовільність коментаря іншим гравцем само собою означає, що він підтверджує слова іншої людини, як її ставлення, так і згоду щодо гідності гри чи її недоліків.

Загалом, вже на цьому етапі можна стверджувати, що основний аналіз припадає все ж на відгуки користувачів, а оцінки можна відповідно вважати за коефіцієнти, що будуть збільшувати вплив даних на кінцевий результат аналізу.

Також оцінки можна використати для поверхневого аналізу даних і побудови допоміжних діаграм.

Підсумовуючи усе вищесказане, необхідно реалізувати програмне забезпечення дотримуючись наступних дій:

- Робота із набором даних (підготовка, зменшення);
- Обробка природньої мови;
- Аналіз тональності відгуків;
- Інтелектуальний аналіз даних;
- Відображення у графічному вигляді результатів аналізів.

### 3. ВІДБІР ТА ПІДГОТОВКА ДАНИХ

#### 3.1 ЗАГАЛЬНИЙ АНАЛІЗ НА КОРЕКТНІСТЬ ДАНИХ

В якості набору даних був обраний dataset наступного вигляду:

app_id	app_name	review_text	review_score	review_votes
10	Counter-Strike	Ruined my life.	1	0
10	Counter-Strike	This will be more of a ''my experience with this game'' type of review, because saying things like '...'	1	1
10	Counter-Strike	This game saved my virginity.	1	0
10	Counter-Strike	• Do you like original games? • Do you like games that don't lag? • Do you like games you can run on...	1	0
10	Counter-Strike	Easy to learn, hard to master.	1	1
10	Counter-Strike	No r8 revolver, 10/10 will play again.	1	1
10	Counter-Strike	Still better than Call of Duty: Ghosts...	1	1

Рисунок 3.1 – Структура вхідних даних

Він налічує у собі такі поля:

- ідентифікатор гри;
- назва гри;
- відгук гравця;
- оцінка гри представлена як 1 – задовольняє, -1 – не задовольняє;
- чи був відгук гравця підтриманий іншими у вигляді 0 – якщо ні, 1 – так.

Отже, необхідно завантажити дані файлу формату .csv у dataframe та перевести поля оцінок та ідентифікатора у тип int. Далі результатом виконання функції info() є перелік загальної кількості полів - 6417106 (рис. 3.2); 5 колонок, як і передбачалося; споживання пам'яті – 171.4+ MB.

```

steam = steam.astype({'app_id':'int', 'review_score':'int', 'review_votes':'int'})
#steam['app_id'].describe()
print(steam.info())
✓ 0.3s

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 6417106 entries, 0 to 6417105
Data columns (total 5 columns):
#   Column      Dtype
---  -
0   app_id      int32
1   app_name    object
2   review_text  object
3   review_score int32
4   review_votes int32
dtypes: int32(3), object(2)
memory usage: 171.4+ MB

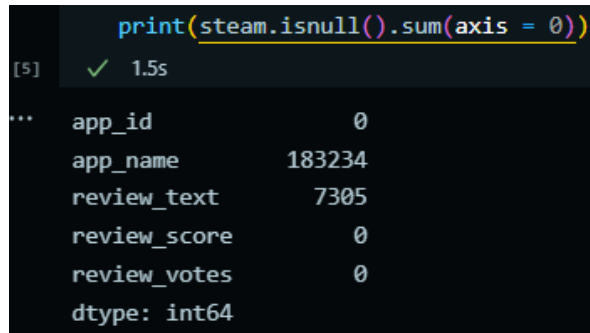
```

Рисунок 3.2 – Результат виконання функції info()

Щодо «чистоти» значень полів, то маємо наступне: що деякі коментарі не мають сенсу (рис. 3.3). А результат оцінки кількості незаповнених полів демонструє, що деяка частина назв ігор та коментарів відсутня (рис. 3.4). І додатково було б краще зобразити оцінку задовільності гри у булевому значенні; тобто як 0 та 1, а не -1 та 1.

app_id	app_name	review_text	review_score	review_votes
		add achievements for CS 1.6 because game is legendary!		
10	Counter-Strike	Wordless.... very very good.	1	1
10	Counter-Strike	Best fps of all time	1	1
10	Counter-Strike	:)	1	1
10	Counter-Strike	I still suck at CS:GO	1	1
10	Counter-Strike		1	1
10	Counter-Strike	Best game thank you valve	1	1
10	Counter-Strike	The best game ever :D !	1	1

Рисунок 3.3 – Приклад некоректних даних



```
print(steam.isnull().sum(axis = 0))
```

```
[5]: ✓ 1.5s
```

```
... app_id          0
     app_name      183234
     review_text    7305
     review_score    0
     review_votes    0
     dtype: int64
```

Рисунок 3.4 – Оцінка загальної кількості незаповнених даних у колонках

Отже, підготовка даних може буде виконана через виконання наступних дій:

- а) видалення полів, що не мають назви гри чи відгуків;
- б) заміна значення -1 на 0 у полі «Оцінка гри»;
- в) пропуск полів, що не мають сильного впливу (розмір відгуку менше ніж 25 символів та не підтриманий іншими гравцями);
- г) видалення полів, що мають некоректні/безглузді відгуки.

### 3.2 Видалення полів із пустими значеннями

З рисунка 3.4 можна зрозуміти, що загалом при очищенні пустот загальна кількість полів може знизитись максимум на 190539 значень. За допомогою бібліотеки pandas маємо змогу видалити стрічки таблиці через використання функції `steam.dropna(inplace = True)`. Таким чином ми змінюємо оригінальну таблицю зменшуючи загальний обсяг стрічок до 6226728.

При повторному виконанні функції `print(steam.isnull().sum(axis = 0))` (рис. 3.4) отримуємо для усіх колонок кількість пустот, що дорівнюють нулю. Отже, очищення було виконано успішно.

### 3.3 Заміна значень у колонці «Оцінка гри»

За допомогою наступного коду вирішується проблема заміни значення -1 на 0 у колонці «Оцінка гри»:

```
counter = 0
```

```
for i in steam.index:
```

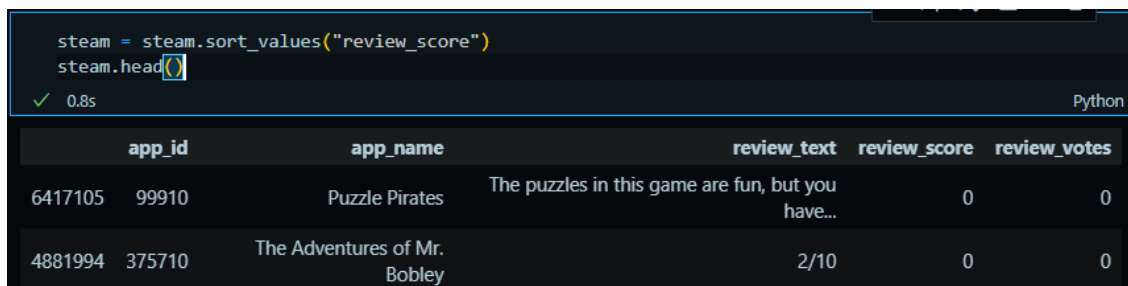
```

if steam.loc[i, "review_score"] == -1:
    steam.loc[i, "review_score"] = 0
    counter += 1

print(counter).

```

Зміна counter дозволяє відміряти кількість замінів. На кінець ітерації значення досягло 1156686. За допомогою методу сортування по колонці «Оцінка гри» та її виведенню, стає видно, що мінімальне значення дійсно стало нулем (рис.3.5).



```

steam = steam.sort_values("review_score")
steam.head()

```

✓ 0.8s Python

app_id	app_name	review_text	review_score	review_votes
6417105 99910	Puzzle Pirates	The puzzles in this game are fun, but you have...	0	0
4881994 375710	The Adventures of Mr. Bobley	2/10	0	0

Рисунок 3.5 – Виведення таблиці відсортованою за «review\_score»

### 3.4 Видалення полів з невеликим впливом

Оскільки задана таблиця досягає великого об'єму у 6226728, то для більш оперативного аналізу слід залишити лише впливові поля. Загалом, якщо гравець написав коментар, що є меншим за 10 символів, то він не розкриває повної думки автора. Якщо ж розмір коментаря до 25, і він не оцінений іншими користувачами, можна вважати його не важливим. Отже, нехай, виконаємо наступний код:

```
steam['letters_count'] = steam['review_text'].apply(lambda x: len(x)).
```

Додаємо нову колонку, що містить розмір коментаря користувача у символах. І тепер, можемо використати нові дані для швидкого створення нового dataframe. У ньому будуть стрічки лише ті, що мають довжину коментарів більше за 10; чи ті, що менше за 25, але мають гарну оцінку. Виконавши заданий алгоритм, отримуємо новий фрейм розміром у 5041078 записів.

### **3.5 Видалення некоректних відгуків**

Даний пункт є у деякому сенсі непростим, оскільки маємо справу з NLP. Важливо визначити, не тільки що саме нас не задовольняє, але й виконати повну нормалізацію текстів для подальшої роботи. Оскільки ця частина вміщує у собі достатню кількість під етапів, то відокремимо його у самодостатній елемент цієї курсової роботи, де розглянемо докладно усі під пункти.



#### 4. СТРУКТУРОВАНА ОБРОБКА ПРИРОДНЬОЇ МОВИ (NLP)

NLP або Natural Language Processing надасть нам змогу проаналізувати природню мову клієнтів сервісу Steam через їх написані відгуки до ігор. Особливістю таких текстів є те, що ми маємо справу не з готовими конкретними даними, а з потоком неструктурованої інформації. Так, ми маємо речення, слова; але через особливості мов маємо багато непотрібної граматичної інформації чи слова вставки, паразити, що не мають багато сенсу при аналізі ігрового ринку. Отже, в першу чергу, необхідно очистити наші відгуки певним способом. Опишемо наступні важливі пункти для реалізації підготовки даних:

- а) переведення знаків тексту у нижній регістр;
- б) видалення посилань;
- в) очищення від непотрібних символів;
- г) токенизація;
- д) видалення стоп-слів;
- е) унормування.

Отож, перейдемо до практичного способу вирішення вищеописаних пунктів. Перші 3 етапи мають на увазі те, що сам текст повинен бути відформатований. Для приведення тексту до нижнього регістру необхідно виконати наступний код:

```
steam['review_text'] = steam['review_text'].astype(str).str.lower().
```

А для інших – очищення текстів за допомогою регулярних виразів, по типу: "[^a-zA-Z\s]+" (текст з букв латинська абетки та пробілів), "w+://s+" (шаблон для пошуку посилань). Оскільки, після коригування текстів їх зміст міг зменшитися, необхідно повторно видалили поля з невеликим впливом (тексти менш як 10 символів чи неоцінені відгуки менше ніж 25 символів). Після виконаної дії кількість полів зменшилась з 5041078 до 4988035.

Наступним етапом є токенизація. Токенизація – обробка тексту, що полягає в розбитті довгих рядків тексту на дрібніші, у нашому випадку – на слова. У цій курсовій роботі токени будуть паралельно існувати у додатковій колонці «tokens». Розбиття реалізується наступною функцією:

```
def delete_long_and_tokenize(text):
    res = [w for w in text.split() if len(w) < 20]
    if len(res) <= 3:
        res = ""
    return res
```

Кожен відгук розбивається на слова, що поміщаються у список при умові, якщо слово менше за 20 символів. Також одразу ж перевіряється кількість слів. Якщо слів менше 3-х включно, то можна вважати коментар не цінним і видалити його після токенизації. Вже на кінець виконання функції отримуємо зменшення фрейму у 200885 до 4787150 полів.

На черзі крок – видалення стоп-слів. Стоп-слова – це ті, що не містять цінної інформації («it», «a», «so» і под.). Їх перелік міститься в стоп-списку, який використовується для видалення відповідних слів у тексті. У нашому випадку використовується стандартний пакет англійських стоп-слів з бібліотеки `nltk`. Єдиний момент, що потребує попереднього редагування у цих словах, це те, що вони мають й інші знаки окрім літер. Наші відгуки вже попередньо були очищенні від таких символів, отже необхідно виконати подібну фільтрацію для пакета стоп-слів. Використовуючи регулярний вираз "[^a-zA-Z]+", очищуємо слова від символів, що не належать літерам латинської абетки. Далі ж виконуємо фільтрування колонки «tokens» від стоп-слів.

В результаті отримуємо колонку з відгуками, яка містить тексти із задуманою граматичною структурою; і чисті токени, що у майбутньому можуть бути унормовані і використані для більш точного аналізу (рис. 4.1). Додатково, як і раніше, перевіряємо отриману кількість токенів, і видаляємо поля у яких вийшло менше 4-х. Таблиця у результаті зменшилася у 453890 до 4333260 полів.

review_text	tokens
best shooter ever buy it or regret it	best shooter ever buy regret
pros no microtransactions fun original game ...	pros microtransactions fun original game mode ...
such game much love many sweat	game much love many sweat
so i just played the original counter strike c...	played original counter strike cs hour hours c...
oh god almost years of my life wasted on this...	oh god almost years life wasted game buy thumbs

Рисунок 4.1 – Результат утворення токенів з відгуків та видалення у них стоп-слів

Завершальний крок – унормування. Виникає потреба у приведенні слів до їх базової форми, тобто кореня. Є два варіанти у вигляді стемізації та лематизації, обидва з яких реалізують це завдання. Єдина різниця полягає у тому, що стемізація на базі суфіксів, префіксів, закінчень, видаляє їх з поданого слова, попри те, чи є отримане слово насправді. А лематизація додатково використовує базу WordNet з наявних в англійській мові слів. Оскільки нас цікавлять дійсні причини успіхів продажу ігор, то лематизація підійде краще. Ґрунтуючись на тому, що бажано привести усі частини мов до їх базової форми, можна створити словник з аббревіатурами частин мов відповідних до тих які використовує `pos_tag`:

```
tags = {"N": wordnet.NOUN,
        "J": wordnet.ADJ,
        "V": wordnet.VERB,
        "R": wordnet.ADV}
```

Алгоритм наступний: обираємо кожне поле колонки «tokens»; розбиваємо на слова; перевіряємо до якої частини мови відноситься слово; приводимо слово до його базової форми, якщо було знайдена відповідна форма у словнику англійської мови. В результаті отримуємо чисті слова, що укладаємо у колонку «tokens». Однак цей алгоритм є достатньо важкий у оперативному сенсі для даних розміром більше за 10 гб. Через це, далі будемо опрацьовувати лише частину даних: `steam = steam[steam['app_id'] <= 20520]`.

Додатково необхідно утворити декілька нових колонок:

а) колонка для утримання кількості токенів:

```
steam['tokens_count'] = steam['tokens'].apply(lambda x: len(str(x).split()));
```

б) колонка для утримання найменування розміру тексту:

```
tbins = np.linspace(min(steam['tokens_count']),max(steam['tokens_count']),
steam['text_size'] = pd.cut(steam['tokens_count'], tbins,
labels=['Little', 'Medium', 'Big'], include_lowest=True).
```

В результаті отримуємо лемматизовані токени із додатковою інформацією, що описують їх (рис. 4.2)

	tokens	tokens_count	text_size
best shooter ever buy regret		5	Little
best game ever joy give play friend unbelievable		8	Little
still always best online fps game ever		7	Little
think every huge counter strike fan game begin...		15	Little
like game play since played lot fine graphic s...		12	Little

Рисунок 4.2 – Кінцевий вигляд токенів, їх кількості, найменування розміру тексту

## 5. АНАЛІЗ ТОНАЛЬНОСТІ ВІДГУКІВ

Кінцевий набір даних зберігає інформацію про оцінки ігор користувачів та оцінку самих відгуків. Але зазначенні параметри зберігаються у булевому форматі і не показують реальний розмах відношення різних людей від позитивного до негативного. Отже, необхідно реалізувати цю «широту» через визначення дійсного відношення людини-коментатора аналізуючи вміст їх відгуків.

Аналіз тональності тексту чи *Sentiment analysis* визначає цілий клас методів в комп'ютерній лінгвістиці. Він надає змогу дослідити великий обсяг повідомлень для автоматизованого виявлення в текстах емоційно забарвленої лексики. Також дозволяє визначити емоційної оцінки авторів, його ставлення щодо об'єктів, мова про які йде в тексті.

В даній роботі для проведення аналізу використовується бібліотека *Vader*. *VADER* (*Valence Aware Dictionary sEntiment Reasoner*) - це лексикон та інструмент аналізу настроїв на основі правил. *Vader* по суті відіграє роль словника, що був сформований через аналіз коротких текстів, по типу тих, що є у соціальних мережах.

Виконуючи функцію для тексту `analyser.polarity_scores(x)`, отримуємо шкалу-оцінку настрою. Вона вміщає в собі параметри: «pos», «neg», «neu» «compound». Загалом, перші з трьох у сумі дорівнюють одиниці й окремо формують коефіцієнт настрою. Останній же варіюється у конкретному діапазоні, де -1 – мінімальне значення та демонструє повний негатив, а 1 – максимальне значення, позитивне ставлення; 0 характеризує нейтральність.

Отже, для визначення настрою відгуків створюємо колонку для збереження «compound». Однак, вже раніше зазначалося, що оцінки ігор користувачів та оцінка самих відгуків можна використовувати як додаткові коефіцієнти при визначенні тональності. Тому нехай визначимо умови для додаткових коефіцієнтів до змінної `compound`:

- а) додати 0.1, якщо оцінка гри позитивна;
- б) додати 0.4 додатково, якщо оцінка гри і оцінка відгуку позитивні;

в) відняти 0.1, якщо оцінка гри негативна;

г) відняти 0.4 додатково, якщо оцінка гри і оцінка відгуку негативні;

І все ж таки, можуть бути ситуації, що навіть при впливі додаткових коефіцієнтів, відгуки із негативною тональністю та позитивною оцінкою гри (та зворотна ситуація) можуть залишитися. Через таку невизначеність, сформуємо додаткове правило. При умові, що відгук тонально негативний, а оцінка гри позитивна, віддаємо перевагу вибору користувача – compound-у присвоюємо 0.2. У зворотній ситуації змінну ототожнюємо з -0.2.

Після підрахунку compound для усіх відгуків додатково формуємо колонку для утримання найменування тональності. При умові, що compound більше за 0.3, то тональності ототожнюємо із «Pos» (позитивний). Якщо ж менше за -0.3, то – «Neg» (негативний), інакше – «Neu» (нейтральний). Як наслідок отримуємо класифіковані відгуки із коефіцієнтом (рис. 5.1).

	tokens	tokens_count	text_size		polarity	compound	sentiment
	best shooter ever buy regret	5	Little	{'neg': 0.215, 'neu': 0.462, 'pos': 0.323, 'co...		0.4400	Pos
	best game ever joy give play friend unbelievable	8	Little	{'neg': 0.0, 'neu': 0.352, 'pos': 0.648, 'comp...		1.0000	Pos
	still always best online fps game ever	7	Little	{'neg': 0.0, 'neu': 0.741, 'pos': 0.259, 'comp...		0.7369	Pos
	think every huge counter strike fan game begin...	15	Little	{'neg': 0.049, 'neu': 0.724, 'pos': 0.227, 'co...		0.7597	Pos
	like game play since played lot fine graphic s...	12	Little	{'neg': 0.093, 'neu': 0.549, 'pos': 0.359, 'co...		0.7486	Pos

Рисунок 5.1 – Результат визначення емоційної оцінки відгуків

## 6. ІНТЕЛЕКТУАЛЬНИЙ АНАЛІЗ ДАНИХ

### 6.1 Різноманітні графіки

Почнемо аналіз з побудови візуалізацій на основі сформованих даних для легшого сприйняття інформації. Бібліотека Seaborn надає можливість створювати діаграми найрізноманітніших виглядів.

Використовуючи її функціонал, побудуємо графік для демонстрації загальної кількості відгуків з позитивною та негативною оцінками ігор пофарбованих зеленим та червоним кольорами відповідно (рис. 6.1.1).

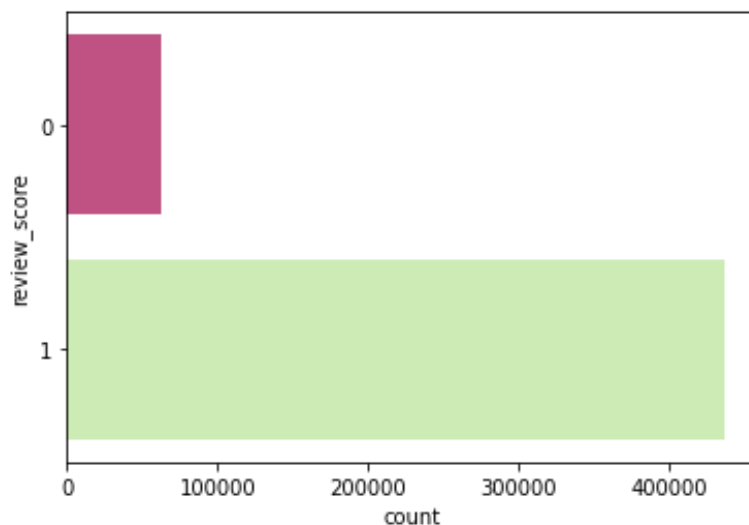


Рисунок 6.1.1 – Графік співвідношення різновидів оцінок ігор до їх кількості

З графіку видно, що наші дані налічують у собі більше позитивних оцінок. Кількість негативних не досягає і одного мільйону.

Подібний графік сформуємо і для представлення тональності відгуків (рис. 6.1.2). Даними для осі ординат стануть різновиди колонки «sentiment».

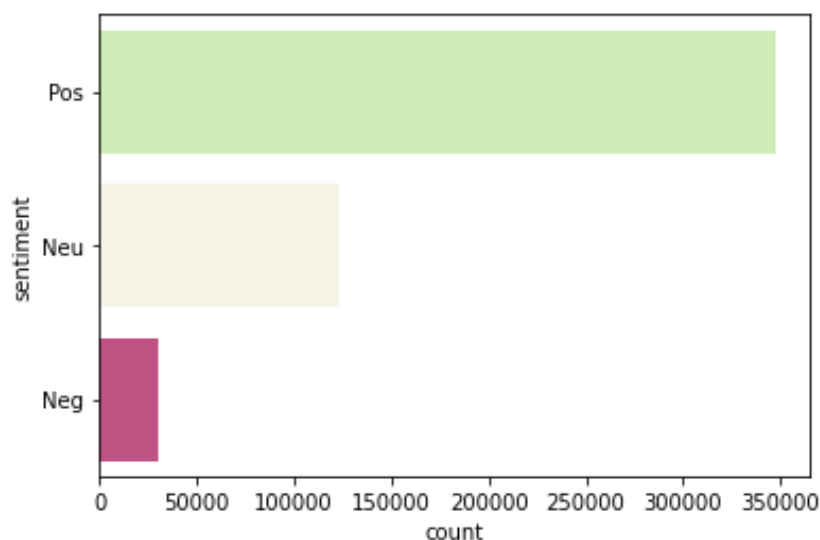


Рисунок 6.1.2 – Графік співвідношення різновидів тональності до їх загальної кількості

Наочно видно, порівнюючи отримані графіки, що відношення зелених ділянок до червоних є достатньо однаковим. Ця закономірність свідчить про те, що аналіз тональності відгуків був реалізований правильно. Звичайно, у другій діаграмі негативна частина є замалою у порівнянні з першим графіком. Але це могло статися через те, що люди залишають незадовільну оцінку гри і недостатньо негативно писали коментар. Саме таким чином поле може потрапити у нейтральну зону. До подібної поведінки відноситься і протилежний випадок (із позитивною емоційною оцінкою). Необхідно також зазначити, що деякі поля також попали у нейтралітет через їх двоякість. Але загалом, оскільки два графіки співвідносяться один до одного, тональність визначена правильно.

Тепер оцінимо відношення кількості слів у відгуках до їх емоційної оцінки (рис. 6.1.3). Для цього використаємо дані колонки «tokens» для підрахунку слів та «sentiment» - для різновидів тональності.



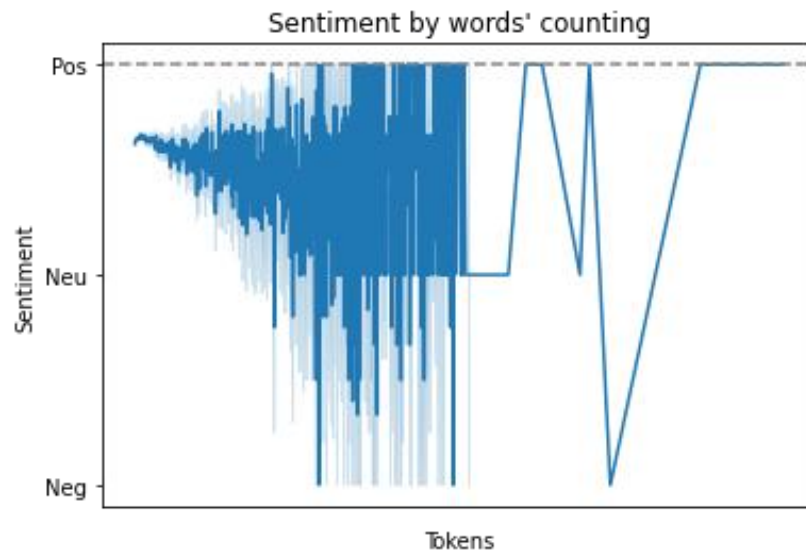


Рисунок 6.1.3 – Графік співвідношення кількості tokenів до емоційності тексту в якому вони були використані

Продивившись результат, можна зазначити декілька висновків. По-перше, на початку графіка (з малою кількістю tokenів), лінія поступово розріджується. Отже, чим менше відгук, тим більша ймовірність, що він позитивний. По-друге, більша частина відгуків середнього чи малого розміру. З середини по лінії абсцис лінія тонка, а отже коментарів у цій області небагато. По-третє, кінець лінії (зона найбільших відгуків) стабільно позитивний. Це свідчить про те, що найбільші коментарі є виключно позитивними. По-четверте, оцінюючи зону негативної тональності, можна зазначити наступне: автори коментарів, судячи з кількості tokenів, віддають перевагу аргументації свого невдоволення.

## 6.2 Аналіз частоти слів

Задачею курсової роботи був аналіз того, що стає успіхом проєктів. Відповідь на це питання може дати уживаність слів. Для отримання даних про повторюваність терміну використаємо FreqDist, у який передаємо усі токени. З отриманого словника відберемо тільки 1000 найуживаніших. Тепер сформуємо такі словники з відібраних tokenів по частинах мови. Для цього виконаємо наступний код:

```
for w in top1000:
```

```

pos = nltk.pos_tag(nltk.word_tokenize(w[0]))
tag = pos[0][1]
if tag == 'NN':
    top_nouns[w[0]] = w[1]
if tag == 'JJ':
    top_adjs[w[0]] = w[1]
if tag == 'VB':
    top_verbs[w[0]] = w[1]

```

В результаті отримали заповнені `top_nouns` (найуживаніші іменники), `top_adjs` (найуживаніші прикметники) та `top_verbs` (найуживаніші дієслова).

Отже, перейдемо до перегляду графіку 20 найуживаніших слів (рис. 6.2.1).

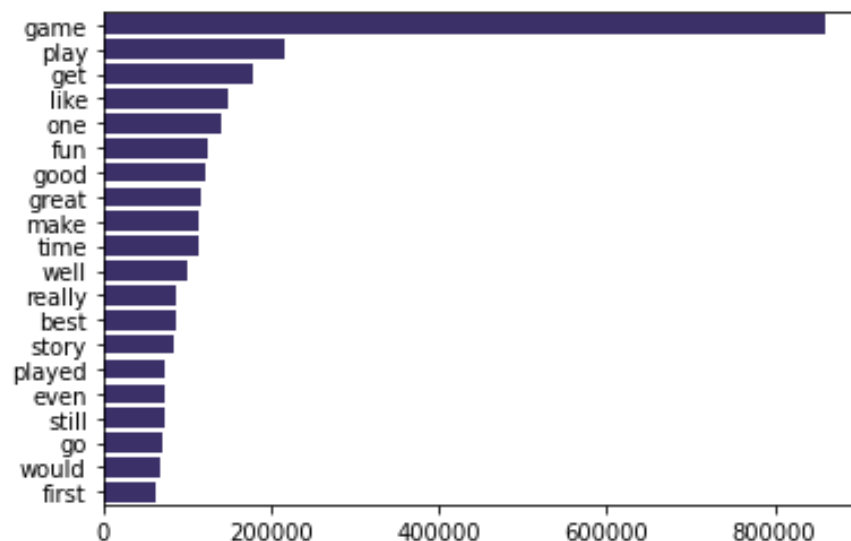


Рисунок 6.2.1 – Графік найуживаніших слів

З нього видно, що топове слово «game», у якомусь сенсі передбачуване. Усі гравці використовують це слово у контексті опису своїх думок, тому це стоп-слово. Подібної характеристики набувають й такі слова: «play», «played», «one», «get», «would».

Тепер розглянемо діаграму для 20 найуживаніших іменників (рис. 6.2.2):

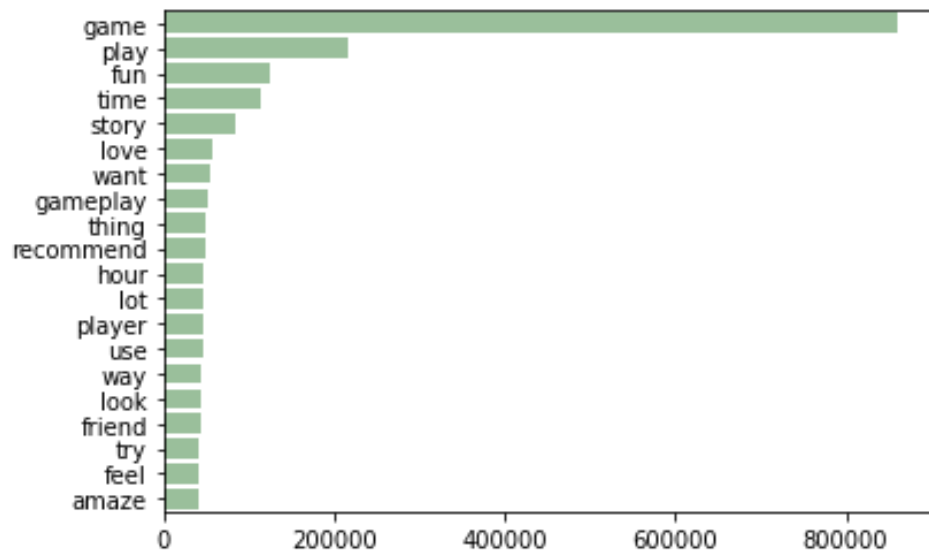


Рисунок 6.2.2 – Графік найуживаніших іменників

Більшість зі слів не мають цінності для розуміння причин успіху, такі як: «game», «play», «player», «gameplay», «lot», «try», «want», «use», «thing», «hour» та т.д.. А ось «fun», «story», «love», «recommend», «friend», «feel», «amaze» важливі у цьому сенсі. З попереднього аналізу нам відомо, що більшість коментарів позитивні та до них відносяться вищеописані важливі слова. Отже, можна стверджувати, що людям-послідовникам люблять історії, забави, здивування, друзі. Якщо комбінувати ці терміни між собою можна утворити конкретні вподобання: «Грати з друзями», «Потішний сюжет», «Непередбачуваний сюжет», «Рекомендує гру» (з найбільшою кількістю відгуків), «Приємне проведення часу», «Чудовий інтерфейс», «Добра взаємодія». А з усіх слів додатково утворюються такі комбінації: «Грати багато годин», «Пробувати грати заново». Через наявність слова «recommend» можна стверджувати, що воно належить до ігор з найбільшою кількістю коментарів. Тому сформуємо перелік 10-ти найпопулярніших ігор (рис. 6.2.3).

app_name	count
Dota 2	40687
Left 4 Dead 2	36679
Portal 2	29272
BioShock Infinite	25579
Garry's Mod	20094
Killing Floor	16550
Portal	12458
Grand Theft Auto IV: The Complete Edition	11460
Just Cause 2	11217
Counter-Strike: Source	10824

Рисунок 6.2.3 – 10 найпопулярніших ігор та кількість їх відгуків

Перелічені ігри належать до жанру стратегічних онлайн чи пригодницьких шутерів. І саме їх рекомендують гравці у своїх відгуках за свою історію, взаємодію і т.д. Однак для конкретизації, додатково продивимося топ для прикметників (рис. 6.2.4) та дієслів (рис. 6.2.5)

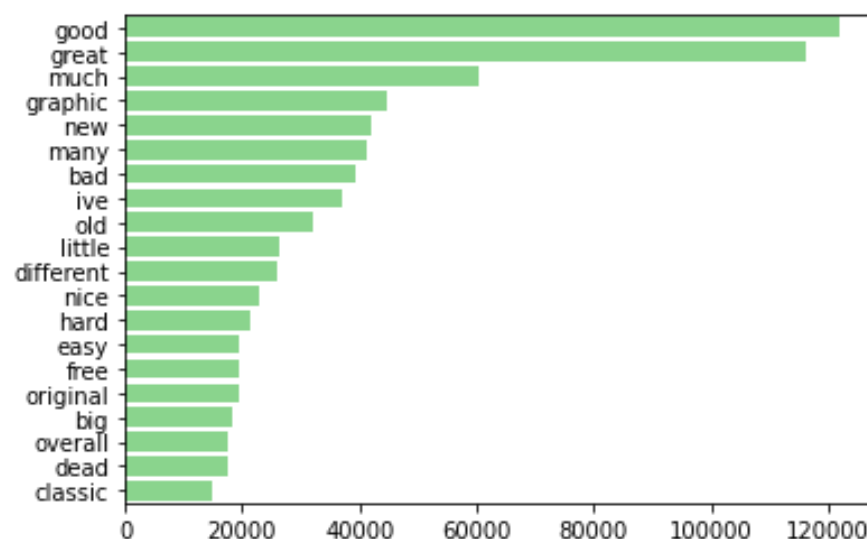


Рисунок 6.2.4 – Найуживаніші прикметники у відгуках

З цікавого, у прикметниках можна віднайти такі конкретизації: «Гарна графіка», «Оригінальна історія», «Безплатна гра». І також зустрічаються антоніми: новий та старий, складний та легкий, великий та малий, добрий та

поганий. Антоніми можуть використовуватись для протиставлення у дискусіях. Тому, враховуючи кількість антонімів по відношенню один до одного, можна стверджувати наступне:

- а) вподобань більше ніж недоліків;
- б) гравці люблять нововведення чи нові ігри;
- в) користувачам цікавіше коли гра складніша ніж простіша.

Щодо дієслів, можна лише визначити, що зробив гравець чи хотів би зробити: дістати, створити, купити, взяти, зберігати, додати, втратити, знайти, побачити щось; піти, повернутись кудись.

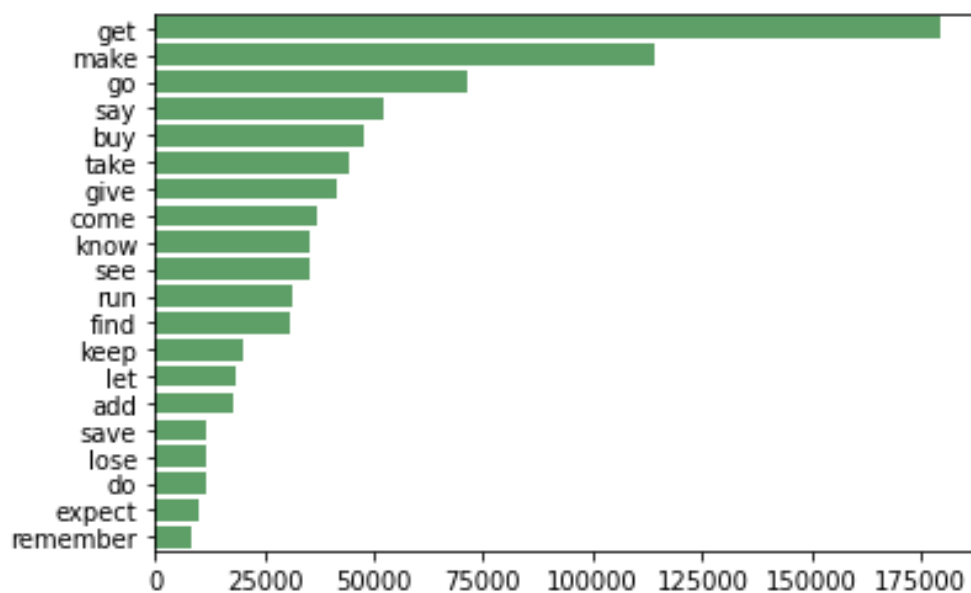


Рисунок 6.2.5 – Найуживаніші дієслова у відгуках

## ВИСНОВОК

В результаті виконання курсової роботи було виконано 6 етапів.

На першому етапі були сформовані цілі цієї роботи, обумовлено стратегію вибору даних, їх обробку та подальші методи аналізу. Також була розкрита проблематика роботи з обраними даними та особливість підходу до розв'язання певних задач.

Далі тема відійшла до суті проєкту. Необхідно визначити проблему, розуміти проблеми галузі, і це все було прописано у розділі. У нашому випадку, освітілась тема формування причин успіху ігрових продуктів, базуючись на словах клієнтів і їх відношенню до продуктів. Отже, для реалізації цього сформувався перелік необхідних даних для майбутньої взаємодії.

Третій етап – відбір даних та їх підготовка. Обраний файл із даними про відгуки клієнтів був описаний на предмет охарактеризування полів та особливостей даних. Сформований список недоліків рядків слугував планом для написання коду. В результаті дані стали більш підготовленою до аналізу.

Четвертий етап є, у свою чергу, продовженням попереднього, у сенсі підготовки даних у стовпцях. Але на цей раз ми концентруємося саме на відгуках. Задачі NLP були сформовані у список і були вирішені за допомогою різноманітних бібліотек Python-а. В результаті отримуємо лематизовані токени через розпізнавання до яких частин мови вони належать.

П'ятий крок є повноцінним переходом до аналізу даних. Предметом роботи став аналіз відгуків користувачів із подальшим розпізнаванням їх емоційного забарвленої лексики. Дана задача була реалізована за допомогою бібліотеки Vader із характеристиками словника. Через аналіз бібліотеки та впливу додаткових коефіцієнтів була визначена тональність відгуків.

Фінальний крок – аналіз даних через приведення інформації до графічного виду та аналізу закономірностей. Через засоби бібліотеки Seaborn були побудовані графіки із різними видами відношень. В більшості випадків, задача була у тому, щоб подивитися на одну конкретну особливість з різних боків. Таким чином формувався висновок щодо успішності продуктів. Сам етап був

реалізований через зіставлення оцінок початкових даних із тими, що довелося прорахувати в процесі аналізу. В першій частині було проаналізована особливість даних: внутрішня класифікація, розподіл відгуків чи оцінок і так далі. В другій частині робота була концентрована на самих токенах та їх кількості. Через методи розподілу слів за частинами мови було визначено, що до вподоби клієнтам чи що вони обговорюють найчастіше.

Підсумовуючи, поставлена задача цієї курсової роботи була вирішена. Ми отримали поради для створення нових ігрових продуктів через аналіз слів користувачів. Саме ж програмне забезпечення демонструє різні підходи аналізу текстів та їх підготовки. Надалі планується реалізувати повноцінне програмне забезпечення із графічним інтерфейсом з лімітом вхідних даних, залишаючи сформовану логіку аналізу.

## ПЕРЕЛІК ПОСИЛАНЬ

1. Wikipedia. Natural language processing//Wikipedia, the free encyclopedia. – 2022. — URL: [https://en.wikipedia.org/wiki/Natural\\_language\\_processing](https://en.wikipedia.org/wiki/Natural_language_processing).
2. Wikipedia. Sentiment analysis // Wikipedia, the free encyclopedia. – 2022. – URL: [https://en.wikipedia.org/wiki/Sentiment\\_analysis](https://en.wikipedia.org/wiki/Sentiment_analysis).
3. Визуализация статистических распределений – 2022 – URL: [https://pyprog.pro/sns/sns\\_6\\_visualization\\_of\\_dist.html](https://pyprog.pro/sns/sns_6_visualization_of_dist.html).
4. Guru99. POS Tagging with NLTK and Chunking in NLP [EXAMPLES] – 2022. – URL: <https://www.guru99.com/pos-tagging-chunking-nltk.html>.
5. Medium. NLP: How does NLTK.Vader Calculate Sentiment? // Medium. – 2020. – URL: <https://medium.com/@mystery0116/nlp-how-does-nltk-vader-calculate-sentiment-6c32d0f5046b>.
6. w3schools. Matplotlib Tutorial – 2022. – URL: [https://www.w3schools.com/python/matplotlib\\_intro.asp](https://www.w3schools.com/python/matplotlib_intro.asp).



**ДОДАТОК А ТЕКСТИ ПРОГРАМНОГО КОДУ**

*Тексти програмного коду забезпечення з створення  
рекомендацій та регресійному аналізу даних про відеоігри*

---

*(Найменування програми (документа))*

*SSD*

---

*(Вид носія даних)*

*10 арк, 133 КБ*

---

*(Обсяг програми (документа), арк., Мб)*

*студента групи ІП-01 ІІ курсу*

*Галько Міли Вячеславівни*

```

# PY libraries
import pandas as pd
import re
import nltk

# NULLS TYPES LETTERS_COUNT COMMENT_FILTER
steam = pd.read_csv(r'SteamReviews.csv')

# int type
# print(steam.isnull().sum(axis = 0))
steam = steam.astype({'app_id':'int', 'review_score':'int', 'review_votes':'int'})

# NULL
steam.dropna(inplace = True)

# -1 to 0
for i in steam.index:
    if steam.loc[i, "review_score"] == -1:
        steam.loc[i, "review_score"] = 0

# letters counts column
steam['letters_count'] = steam['review_text'].apply(lambda x: len(x))

# comments
steam = steam[steam['letters_count'] > 10]
steam = steam[(steam['letters_count'] >= 25) | (steam['review_votes'] == 1)]
steam = steam.sort_values('letters_count')

### LOAD FILE_1
# steam.to_csv('steam1.csv')

```

```

steam = pd.read_csv(r'steam1.csv')
steam.drop('Unnamed: 0', inplace=True, axis=1)
steam = steam.sort_values("app_id")
steam.head()

```

```

# LOWER REGEX COMMENT_FILTER

```

```

def cleanSymbols(text):
    res = re.sub(r"^[a-zA-Z\s]+", "", text)
    return res

```

```

def cleanLinks(text):
    res = re.sub(r"http\s+", "", text)
    res = re.sub("w+://s+", "", res)
    return res

```

```

steam['review_text'] = steam['review_text'].astype(str).str.lower()
steam['review_text'] = steam['review_text'].apply(cleanLinks)
steam['review_text'] = steam['review_text'].apply(cleanSymbols)

```

```

# UPDATE LETTERS_COUNT & DELETE COMMENTS

```

```

steam['letters_count'] = steam['review_text'].apply(lambda x: len(x))
steam = steam.sort_values('letters_count')
steam = steam[steam['letters_count'] > 10]
steam = steam[(steam['letters_count'] >= 25) | (steam['review_votes'] == 1)]
steam = steam.sort_values('letters_count')

```

```

### LOAD FILE_2 4988035

```

```

steam.to_csv('steam2.csv')
steam = pd.read_csv(r'steam2.csv')
steam.drop('Unnamed: 0', inplace=True, axis=1)
steam = steam.sort_values("app_id")

```

```

steam.head()

# TOKENS
def delete_long_and_tokenize(text):
    res = [w for w in text.split() if len(w) < 20]
    if len(res) <= 3:
        res = ""
    return res

steam["tokens"] = steam["review_text"].apply(lambda x:
delete_long_and_tokenize(x))

steam = steam[steam['tokens'].str.len() > 0]

### LOAD FILE_3 TOKENIZED 4787150
steam.to_csv('steam3.csv')
steam = pd.read_csv(r'steam3.csv')
steam.drop('Unnamed: 0', inplace=True, axis=1)
steam = steam.sort_values("app_id")
steam.head()

# STOPWORDS
# nltk.download('stopwords')
SW = list(nltk.corpus.stopwords.words("english"))
for i in range(len(SW)):
    SW[i] = re.sub(r"^[a-zA-Z]+", "", SW[i])
stop_words = set(SW)

def stopwords_cleaner(text):
    text = str(text)[2:-2]
    text = text.split("'", "'")
    res = [w for w in text if w not in stop_words]

```

```

    if len(res) <= 4:
        return "
    return ' '.join(res)
steam['tokens'] = steam['tokens'].apply(stopwords_cleaner)
steam = steam[steam['tokens'].str.len() > 0]
steam.head()

### LOAD FILE_4 STOPWORDS 4333260
steam.to_csv('steam4.csv')
steam = pd.read_csv(r'steam4.csv')
steam.drop('Unnamed: 0', inplace=True, axis=1)
steam = steam.sort_values("app_id")
steam.head()

### LOAD FILE_SMALL 20520
steam = steam[steam['app_id'] <= 20520]
steam.to_csv('small_steam.csv')
steam = pd.read_csv(r'small_steam.csv')
steam.drop('Unnamed: 0', inplace=True, axis=1)
steam = steam.sort_values("app_id")
steam.head()

# LEMMAS
#nltk.download('wordnet')
#nltk.download('omw-1.4')
from nltk.stem.wordnet import WordNetLemmatizer
from nltk.tokenize import word_tokenize
from nltk.corpus import wordnet
from nltk import pos_tag

```

```

def get_pos(word):
    tag = nltk.pos_tag([word])[0][1][0].upper() # 1st word -> tag -> 1st letter
    tags = {"N": wordnet.NOUN,
            "J": wordnet.ADJ,
            "V": wordnet.VERB,
            "R": wordnet.ADV}
    return tags.get(tag, wordnet.NOUN)

lem = WordNetLemmatizer()
def lem_text(text):
    newL = []
    for w in nltk.word_tokenize(text):
        ne = lem.lemmatize(w, get_pos(w))
        newL.append(ne)
    return newL

steam['tokens'] = steam["tokens"].apply(lem_text)
#steam['lemmas'] = steam["tokens"].apply(lem_text)
steam.head()

# TOKENS_COUNT
def list_to_text(text):
    text = str(text)[2:-2].split("'", "'")
    return ' '.join(text)

steam['tokens'] = steam['tokens'].apply(list_to_text)
steam['tokens_count'] = steam['tokens'].apply(lambda x: len(str(x).split()))
steam.head()

### LOAD FILE_SMALL_LEMMED
steam.to_csv('small_steam_lemmed.csv')
steam = pd.read_csv(r'small_steam_lemmed.csv')

```

```

steam.drop('Unnamed: 0', inplace=True, axis=1)
steam = steam.sort_values("app_id")
steam.head()

# NEW COLUMN - TEXT_SIZE
import numpy as np
tbins = np.linspace(min(steam['tokens_count']), max(steam['tokens_count']), 4)
steam['text_size'] = pd.cut(steam['tokens_count'], tbins, labels=['Little',
'Medium', 'Big'], include_lowest=True)

# POLARITY
nltk.download('vader_lexicon')
from nltk.sentiment import SentimentIntensityAnalyzer

analyser = SentimentIntensityAnalyzer()
steam['polarity'] = steam['review_text'].apply(lambda x:
analyser.polarity_scores(x))
steam['compound'] = steam['polarity'].apply(lambda pol: pol['compound'])

# COMPOUND UPDATE
def update_compound(com, score, votes):
    if score == 1:
        com += 0.1
        if votes == 1:
            com += 0.4
    else:
        com -= 0.1
        if votes == 1:
            com -= 0.4

```

```

    com = -1 if com < -1 else com
    com = 1 if com > 1 else com
    com = 0.2 if (score == 1 and com < 0) else com
    com = -0.2 if (score == 0 and com > 0) else com
    return com

    sub['compound'] = sub.apply(lambda row: update_compound(row['compound'],
row['review_score'], row['review_votes']), axis=1)

    sub['sentiment'] = pd.cut(sub['compound'], [-1,-0.3, 0.3, 1], labels=['Neg', 'Neu',
'Pos'], include_lowest=True)

    steam['compound'] = steam.apply(lambda row:
update_compound(row['compound'], row['review_score'], row['review_votes']),
axis=1)

    steam['sentiment'] = pd.cut(steam['compound'], [-1,-0.3, 0.3, 1], labels=['Neg',
'Neu', 'Pos'], include_lowest=True)

    sub.head()

### LOAD FILE_SENTIMENT
# steam.to_csv('sentimented.csv')
steam = pd.read_csv(r'sentimented.csv')
steam.drop('Unnamed: 0', inplace=True, axis=1)
steam = steam.sort_values("app_id")
steam.head()

# GRAPHICS
import seaborn as sns

sns.countplot(y='review_score', data=steam, palette=['#D14081', '#CCF5AC'])

sns.countplot(y='sentiment', data=steam, palette=['#CCF5AC', '#F9F5E3',
"#D14081"]);

```



```

g = sns.lineplot(x='tokens_count', y='sentiment', data=steam)
g.set(xticklabels=[])
g.set(title="Sentiment by words' counting")
g.set(xlabel="Tokens")
g.set(ylabel="Sentiment")
g.tick_params(bottom=False)
g.axhline(0, ls='--', c = 'grey');

```

```

# ALL WORDS FREQUENCY_DIST
from nltk.tokenize import word_tokenize
from nltk.probability import FreqDist

```

```

all_tokens = ' '.join([token for token in steam['tokens']])
all_tokens = nltk.word_tokenize(all_tokens)

```

```

fd = FreqDist(all_tokens)
top1000 = fd.most_common(1000)

```

```

# POS FREQUENCY_DIST
is_noun = lambda pos: pos[:2] == 'NN'
top_nouns = FreqDist()
top_adjs = FreqDist()
top_verbs = FreqDist()

```

```

for w in top1000:
    pos = nltk.pos_tag(nltk.word_tokenize(w[0]))
    tag = pos[0][1]
    if tag == 'NN':
        top_nouns[w[0]] = w[1]
    if tag == 'JJ':

```

```

    top_adjs[w[0]] = w[1]
    if tag == 'VB':
        top_verbs[w[0]] = w[1]

# TOP 20 WORDS
top = fd.most_common(20)
topS = pd.Series(dict(top))
sns.barplot(y=topS.index, x=topS.values, color='#372772')

# TOP 20 NOUNS
top_20n = top_nouns.most_common(20)
topS = pd.Series(dict(top_20n))
sns.barplot(y=topS.index, x=topS.values, color='#94C595')

# TOP 20 ADJS
top_20a = top_adjs.most_common(20)
topS = pd.Series(dict(top_20a))
sns.barplot(y=topS.index, x=topS.values, color='#7EE081')

# TOP 20 VERBS
top_20v = top_verbs.most_common(20)
topS = pd.Series(dict(top_20v))
sns.barplot(y=topS.index, x=topS.values, color='#52AA5E')

# TOP GAMES
games = steam[['app_name']]
games = games.groupby('app_name').value_counts().to_frame()
games.rename(columns={games.columns[0]: 'count'}, inplace=True)
games = games.sort_values('count', ascending=False)

```

```
games = games.head(10)  
games.head(10)
```

Решта коду знаходиться за посиланням: