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 російськомовних новин. Створення ETL процесу та класифікація на основі нейронних мереж”

Студента 2 курсу ІТ-02 групи

Спеціальності: 121

«Інженерія програмного забезпечення»

Макаров Іллі Сергійовича

«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 ІТ-02 Семестр 4

**ЗАВДАННЯ**

**на курсову роботу студента**

Макарова Іллі Срегійовича

1.Тема роботи: Аналіз російськомовних новин. Ствоерення ETL процесу та класифікація на основі нейронних мереж

2.Строк здачі студентом закінченої роботи: 19.06.2022

3. Вхідні дані до роботи методичні вказівки до курсової роботи, обрані дані з сайту: <https://github.com/dkulagin/kartaslov/tree/master/dataset/kartaslovsent>

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

1.Постановка задачі

2.Аналіз предметної області

3.Розробка ETL процесу

4.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 | 20.04.2022 |  |
| 3. | Пошук та вивчення літератури з питань курсової роботи | 22.04.2022 |  |
| 4. | Розробка ETL процесів | 29.04.2022 |  |
| 5. | Сентиментальний аналіз | 05.05.2022 |  |
| 6. | Класифікація новин | 15.05.2022 |  |
| 7. | Підготовка пояснювальної записки | 04.06.2022 |  |
| 8. | Здача курсової роботи на перевірку | 14.06.2022 |  |
| 9. | Захист курсової роботи | 15.06.2022 |  |

| Студент |  |  | Наконечна Ірина Олександрівна |
| --- | --- | --- | --- |
|  | (підпис) |  | (прізвище, ім’я, по батькові) |

| Керівник |  |  | доц. Ліхоузова Т.А |
| --- | --- | --- | --- |
|  | (підпис) |  | (прізвище, ім’я, по батькові) |
| Керівник |  |  | доц. Олійник Ю.О. |
|  | (підпис) |  | (прізвище, ім’я, по батькові) |

"26" червня 2022 р.

**АНОТАЦІЯ**

Пояснювальна записка до курсової роботи: сторінок 42, 13 рисунки, 8 посилань.

Об’єкт дослідження: інтелектуальний аналіз даних.

Предмет дослідження: створення програмного забезпечення, що проводить інтелектуальний аналіз даних з подальшим прогнозуванням та визначенням тональності даних.

Мета роботи: проектування та реалізація ETL процесів, реалізація програмного забезпечення для завантаження даних з телеграм каналів та їх подальшого аналізу та прогнозування, візуалізація отриманих результатів.

Дана курсова робота включає в себе: створення власного датасету, реалізація ETL процесів (Extract, Transform, Load), опис створення програмного забезпечення для інтелектуального аналізу даних, сентиментальний аналіз, прогнозування даних за допомогою моделей машинного навчання та нейронних мереж, візуалізація та аналіз отриманих результатів.

МОДЕЛЬ ПРОГНОЗУВАННЯ, ІНТЕЛЕКТУАЛЬНИЙ АНАЛІЗ ДАНИХ, ELT ПРОЦЕСИ, МОДЕЛЬ SVM, МОДЕЛЬ K NEAREST NEIGHBORS, МОДЕЛЬ DECISION TREE, МОДЕЛЬ GAUSSIAN NAIVE BAYES, СЕНТИМЕНТАЛЬНИЙ АНАЛІЗ.

**ЗМІСТ**

[ВСТУП](#_gjdgxs) 6

[1.ПОСТАНОВКА ЗАДАЧІ 7](https://docs.google.com/document/d/1hRKw6KZWLFLywNr_SMoR_Iekn7nAYSdlBst8OX7P-yg/edit#heading=h.30j0zll)

[2.АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ 9](https://docs.google.com/document/d/1hRKw6KZWLFLywNr_SMoR_Iekn7nAYSdlBst8OX7P-yg/edit#heading=h.tyjcwt)

[3.РОЗРОБКА СХОВИЩА ДАНИХ](https://docs.google.com/document/d/1hRKw6KZWLFLywNr_SMoR_Iekn7nAYSdlBst8OX7P-yg/edit#heading=h.3dy6vkm) 8

[3.1.РОЗРОБКА ETL ПРОЦЕСІВ](https://docs.google.com/document/d/1hRKw6KZWLFLywNr_SMoR_Iekn7nAYSdlBst8OX7P-yg/edit#heading=h.4d34og8) 8

3.2.ПЕРЕДОБРОБКА ДАНИХ 9

[4.ІНТЕЛЕКТУАЛЬНИЙ АНАЛІЗ ДАНИХ](https://docs.google.com/document/d/1hRKw6KZWLFLywNr_SMoR_Iekn7nAYSdlBst8OX7P-yg/edit#heading=h.2s8eyo1) 10

4.2. КЛАСИФІКАЦІЯ НОВИН 11

4.2.1 КЛАСИФІКАЦІЯ ЗА ДОПОМОГОЮ NN МОДЕЛЕЙ 11

4.2.2 ВИСНОВКИ 15

5.ВІЗУАЛІЗАЦІЯ ОТРИМАНИХ РЕЗУЛЬТАТІВ 16

5.1 РОЗПОДІЛ ТИПІВ НОВИН 16

5.2 РОЗПОДІЛ ТИПІВ НОВИН ЗА ЧАСОМ 17

5.3 РОЗПОДІЛ НОВИН ЗА ЕМОЦІЙНИМ ЗАБАРВЛЕННЯМ 21

5.4 НОВИНИ В КАТЕГОРІЇ “SHELLING” 23

[ВИСНОВКИ](https://docs.google.com/document/d/1hRKw6KZWLFLywNr_SMoR_Iekn7nAYSdlBst8OX7P-yg/edit#heading=h.3rdcrjn) 24

[ПЕРЕЛІК ПОСИЛАНЬ](https://docs.google.com/document/d/1hRKw6KZWLFLywNr_SMoR_Iekn7nAYSdlBst8OX7P-yg/edit#heading=h.26in1rg) 25

ДОДАТОК А ТЕКСТИ ПРОГРАМНОГО КОДУ 26

**ВСТУП**

Основним предметом даного дослідження є аналіз російськомовних новин. Головною метою перед собою ми поставили класифікувати новини, провести сентиментальний аналіз та порівняти між собою російськомовні проукраїнські та російські пропагандистські новинні телеграм канали. У зв’язку з початою Росією війною проти України, ця тема є надзвичайно актуальною, адже демонструє, які існують відмінності між російською пропагандою та правдивими новинами.

Нам також було цікаво прослідкувати наскільки змінились новини до та після початку війни. Зважаючи на це, для нашого дослідження ми вирішили обрати новини, які були опубліковані в період з 10 лютого по 26 травня 2022 року.

В рамках даної курсової роботи був розроблений ETL процес для збору та обробки даних, реалізовані моделі машинного навчання та нейронних мереж для прогнозування категорій новин, здійснений сентиментальний аналіз новин та зроблена візуалізація результатів нашої роботи.

**2. АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ**

Через розпочату Росією війну проти України, ранок чи не кожного українця починається з листання стрічки новин. В Телеграмі все частіше з’являються нові канали з новинами, а на старі з кожним днем підписується все більше людей, щоб бути в курсі всіх останніх подій. Багато з цих телеграм каналів працюють на ворога, дезінформуючи та дезорієнтуючи своїх читачів. Через це виникла гостра необхідність у знаходженні основних рис, що відрізняють російську пропаганду від правдивих новин.

Усі новини ми класифікуємо на такі чотири категорії:

* політичні
* економічні
* гуманітарні
* військові

У програмному забезпеченні буде реалізовано наступну функціональність, що включає в себе:

* створення ETL процесів для завантаження даних
* препроцесинг даних
* використання моделей прогнозування даних (Support Vector Machine, K Nearest neighbors, Decision Tree, Gaussian Naive Bayes)
* сентиментальний аналіз
* візуалізація отриманих результатів та їх аналіз.

**3.** [**РОЗРОБКА СХОВИЩА ДАНИХ**](https://docs.google.com/document/d/1hRKw6KZWLFLywNr_SMoR_Iekn7nAYSdlBst8OX7P-yg/edit#heading=h.3dy6vkm)

*3.1 РОЗРОБКА ETL:*

Для збору даних о новинах, був створений ETL скрипт для збору та preprocessing новин.

Пайплайн може працювати у двох режимах:

1. Зібрати дані, за обраний період у часі та виконати препроцесинг, після чого розсортувати новини, створивши для кожного новостного каналу окремий csv файл з даними.
2. Зібрати дані, за обраний період у часі та виконати препроцесинг, після чого розсортувати новини, створивши для кожного новостного каналу окремий csv файл з даними. Виконати класифікацію новин на 4 категорії, описані в “АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ”, виконати сентиментальний аналіз, розділивши дані на 3 категорії (позитивні, негативні, нейтральні). Та створити csv файл зі всіми словами, що зустрічаються в категорії Shelling.

Те, в якому режимі працювати вказує env variable: “PIPELINE\_WORK\_MODE”, якщо він дорівнює “FULL” то працює у режимі “2”, якщо “FETCH” то в “1”

Список каналів, дані з яких треба зібрати зберігається в “channel\_registry.json” файлі, в директорії config/. Репозиторій це json файл, що має в собі лише одне поле: CHANNEL\_LIST, що зберігає список посилань на телеграм канали. У даний список можна додавати, як посилання на канали, так і унікальну назву каналу, працювати буде, що те, що інше.

Там же, в config/ директорії, також є файл зі стоп словами, що необхідно вилучити з тексту, під час предобробки, і поруч із ним є файл config.py, що зберігає в собі купу інших налаштувань роботи системи.

В ході ELT новини будуть зібрані в директорії data/news-posts, де буде створений окремий csv файл з даними для кожного каналу. Після чого, під час наступного кроку, по побудові predictions ці файли будуть склеєні в один, data/processed-data/classified-news.csv, де будуть зібрані всі новини, з усіх каналів. Поруч, також буде створений data/processed-data/words-bag-shelling.csv де будуть зберігатись слова пов’язані Shelling Category наших новин, ці слова потім нам знадобляться для побудови Words Cloud.

3.2 *ПЕРЕДОБРОБКА ДАНИХ*

Ледь не найважливішим в NLP, є preprocessing даних. Оскільки це є частиною ETL процесу, я опишу як відбувається preprocessing саме тут. Клас для preprocessing реалізує інтерфейс ITextPreprocessor, що має декілька методів:

1. preprocess\_text(text: str) -> str - винонує весь preprocessing тексту
2. preprocess\_and\_lemmatize(text: str) -> str - винонує весь preprocessing тексту та лемматизує його, перш ніж повернути
3. sklearn\_vectorize\_text(self, texts, make\_preprocessing) - виконує необхідні дії, щоб список текстів, що прийде на вхід до метода перетворився на читабельний для ml моделей з sklearn. Робить preprocessing, токенізує текст.
4. keras\_tokenize\_and\_pad\_text(...) - виконує необхідні дії, щоб список текстів, що прийде на вхід до метода перетворився на читабельний для нейронної мережі формат. Робить preprocessing, векторизує та вирівнює текст.

Який же preprocessing реалізує наш клас TextPreprocessor. Ось список методів:

1. remove\_links(text: str), що прибирає з тексту посилання
2. remove\_emoji(text: str), що прибирає з тексту emoji
3. remove\_stop\_words(text: str), вилучає стоп слова, що зберігаються в config/russian\_stop\_words.json
4. remove\_punctuation(text: str), приберає всі можливі знаки пунктуації
5. remove\_extra\_spaces(text: str), видалить зайві пробіли

Ось якось так, під капотом клас юзає кілька ліб для ефективного preprocessing, серед них: pymorphy2, nltk.

Також, при ініціалізації препроцесора, він приймає на вхід sklearn.feature\_extraction.text.TfidfVectorizer та keras.preprocessing.text.Tokenizer, що дозволяє нам підготувати зовсім сирі дані одразу до формату приємного для NN та ML моделей, за допомогою виклику всього одного відповідного методу. Це дуже спрощує нам тестування наших моделей в подальшому, так як для наглядної демонстрації роботи нам достатньо просто провести написаний нами текст через відповідні методи, та він одразу буде готовий до класифікації.

**4.** [**ІНТЕЛЕКТУАЛЬНИЙ АНАЛІЗ ДАНИХ**](https://docs.google.com/document/d/1hRKw6KZWLFLywNr_SMoR_Iekn7nAYSdlBst8OX7P-yg/edit#heading=h.2s8eyo1)

КЛАСИФІКАЦІЯ НОВИН

Найперша проблема з якою ми зіткнулися, це відсутність даних для навчання. Навряд можна було знайти десь датасет з російськомовними новинами, що розділений на 4 вказаних категорії. До того ж дані категорії достатньо специфічні для новин саме воєнного часу. Тож най першим нашим завдання було створити такий датасет. Робили ми це звісно вручну, відібравши близько 1.5к новин, і проставивши їм відповідні категорії. 1.5 тис. записів це надзвичайно мало, особливо для навчання глибоких нейронних мереж, однак результати які ми отримали були достатньо “нормальними” для того, аби ми перестали мучати себе, а я нагадую вам, що нам доводилось читати і маркувати в тому числі російські новини, що є окремим “задоволенням”.

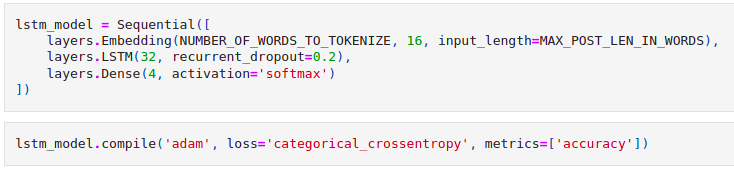
*КЛАСИФІКАЦІЯ ЗА ДОПОМОГОЮ NN МОДЕЛЕЙ*

Зазвичай, для тренування моделей нейронних мереж, треба сотні тисяч записів, особливо коли ми говоримо про глибокі нейронні мережі. Однак, тут, я все ж вирішив подивитись, що в мене вийде з таким обмеженим датасетом. Для тренування я обрав дів архітектури: LSTM та одновимірну CNN.

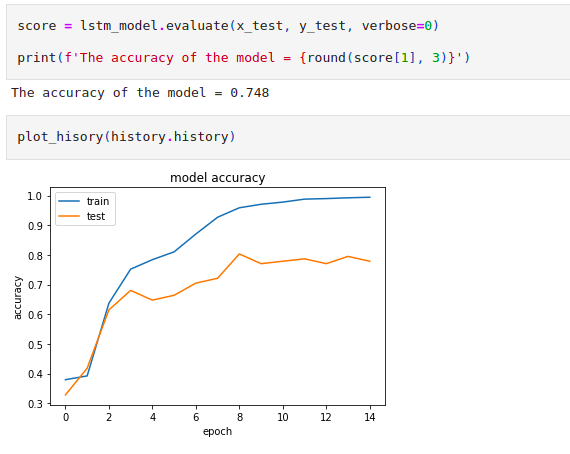
Однак, кількість даних це не єдина велика проблема з якою ми зіткнулись. Нейрона мережа, не сприймає текст, їй потрібен вектор, що буде подаватись на вхід. Для цього, в нейронних мережах створюється спеціальний слой Embedding, це перший шар нашої мережі, що перетворює вхідні дані на щільні вектори. Однак, для навчання такого слою треба просто якась неймовірна кількість даних, тому зазвичай використовують вже навчанні слої (щось накштал word2vec, для англійської мови), однак очевидно, що такі штуки специфічні для кожної мови окремо, і для російської мови ми щось шукали-шукали, та так і не знайшли, тому довелось навчати його самим, що значно знижує якість роботи мережі.

**LSTM**

Дана архітектура є досить старим покращенням звичайно рекурентної нейронної мережі. Проблема будь-якої RNN, що при “розгортанні” в часі нашої моделі (що ми робимо для її навчання) в нас завжди виходить дуже і дуже глибока нейронна мережа, з десятками, якщо не сотнями слоїв (в залежності від довжини текстів, що ми аналізуємо). Така глибина створює купу проблем, в тому числі велику проблему “зникаючого градієнта”. В архітектурі LSTM є кілька суттєвих покращень, що разом дозволяють навчати наші моделі значно краще.



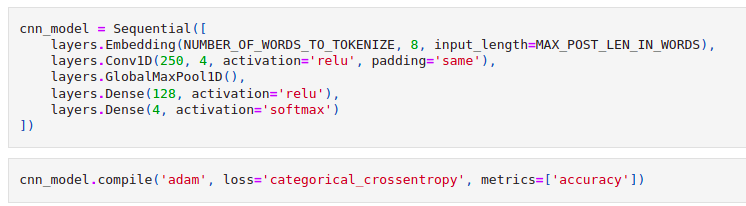
Архітектуру першої моделі тут видно, не зважаючи на те, що наче в нас всього один прихований шар, при розгортанні цієї моделі в часі, вона перетвориться на справжню глибоку сітку.

Давайте подивимось на результати навчання:

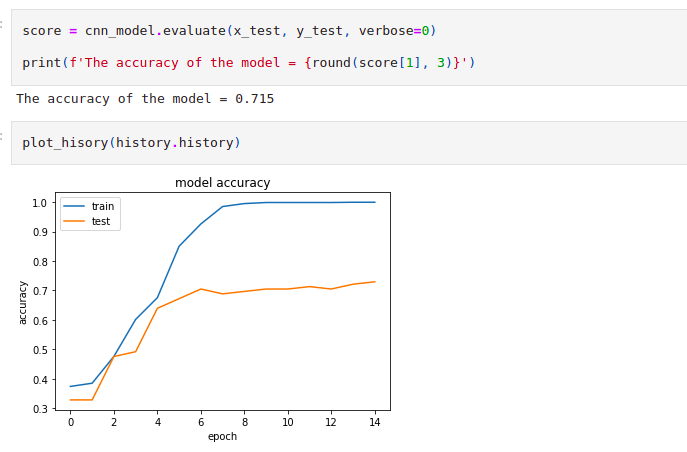
Як видно, точність є не дуже гарною, і на графіку нижче, дуже гарно видно, як починаючи вже з 4-5 епохи тестова точність нашої моделі майже не змінюється, тобто почалось перенавчання.

Чи можна тут позбутись перенавчання? Я думаю ні, бо рекурентна мережа LSTM є достатньо “розумною”, і в ситуації з дуже обмеженим датасетом, мережа дуже швидко просто вивчає його, замість того, щоб шукати закономірності, і починається перенавчання.

**CNN**

Принцип роботи згорткових нейронних мереж - це пошук патернів, потім пошук більш складних патернів серед перших, що дуже добре працює з аналізом зображень, однак, чи буде працювати з текстом? Ось зараз і перевіримо. Взагалі, серед літератури, що я прочитав останнім часом, я часто зустрічав думки, що CNN моделі скоро замінять рекурентні мережі, бо вони значно легше в тренуванні, та значно менші самі по собі. 

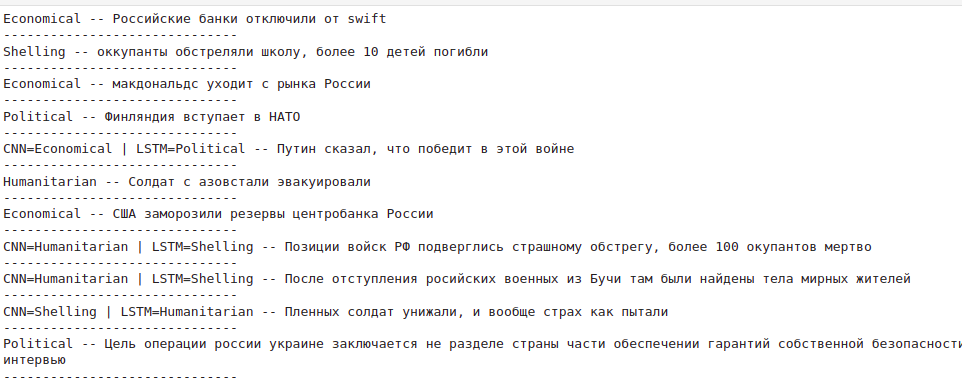
Нічого незвичайного, в нас є лише один згортковий шар, потім йде max пулінг, я читав, в що задачах NLP він значно краще за будь-який інший (avgpool або minpool). І далі ще кілька просто повнозв’зних слоїв.



Як бачимо результати тут трохи гірші, перенавчання починається вже на 4й епосі, далі точність зовсім не йде догори.

**Results**

Тепер давайте поглянемо як наші моделі predict реальні тексти:



Як видно, CNN помилилась аж в 4-х випадках, в той час як LSTM лише в одному, таким чином я думаю, що рано ще казати, що згорткові нейронні мережі скоро замінять нам рекурентні.

*ВИСНОВКИ*

Таким чином, ми розглянули 6 моделей, як видно всі вони, крім Decision Tree видають приблизно однакові результати, близькі до 75% точності. 75% це зовсім не багато, і ми довго думали, що ж робити, однак детальний аналіз та тестування показало, що моделі рідко помиляються однаково, тому, в нашій роботі ми не обмежуємо себе однією моделлю, а використовуємо цілих 5 (всі окрім Decision Tree). Отримавши 5 predictions, ми обираємо найпопулярніший та беремо його в якості відповіді.

Для цього був написаний класс Predictor, що під капотом має 3 модельки ML та дві нейронні мережі та передбачає новини, обираючи найпопулярніше передбачення серед всіх них.

**5. ВІЗУАЛІЗАЦІЯ ТА АНАЛІЗ ОТРИМАНИХ РЕЗУЛЬТАТІВ**

Для наглядності нашого дослідження ми зробили візуалізацію даних в Google Data Studio, в якій ми зображуємо отримані результати та аналізуємо їх. Дивлячись на отримані таблиці та чарти можна порівняти як саме відрізняються російськомовні проукраїнські та проросійськи новинні канали.

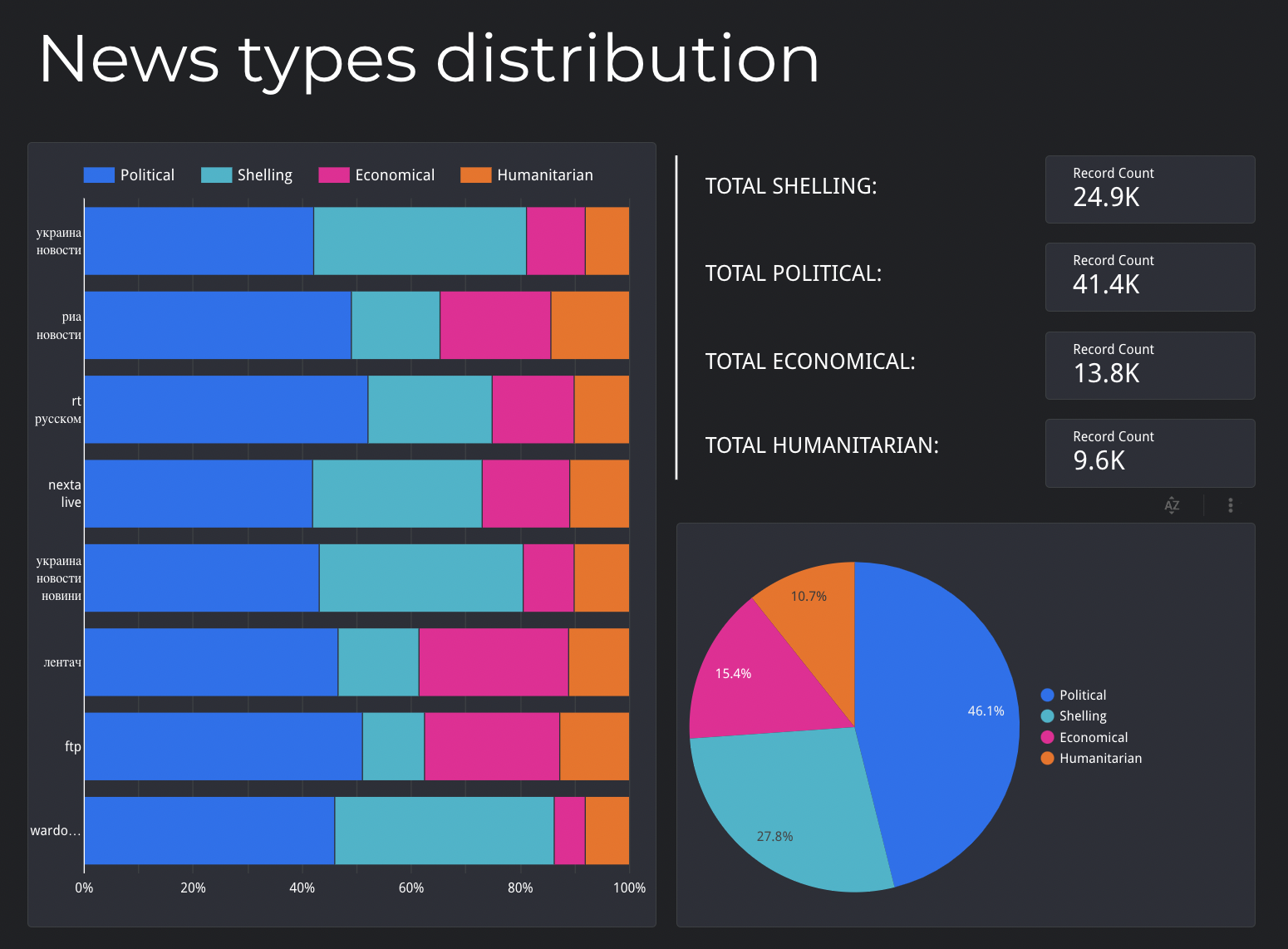


Рисунок 1

На рисунку 1 зображено, на які саме типи ми ділимо всі новини, та як вони співвідносяться між собою.

Отже, всі новини ми поділяємо на чотири групи: економічні, політичні, гуманітарні та воєнні. На даній сторінці звіту можна побачити скільки всього новин кожної групи було відібрано. Як бачимо, кількість політичних новин за період війни найбільша.

Серед вибраних каналів проукраїнськими є “Украина Новости”, “Украина Новости Новини” та “Nexta Live”. Як ми можемо помітити, то в проукраїнських новинних каналах частка новин з воєнною тематикою переважно більше, ніж в проросійських.

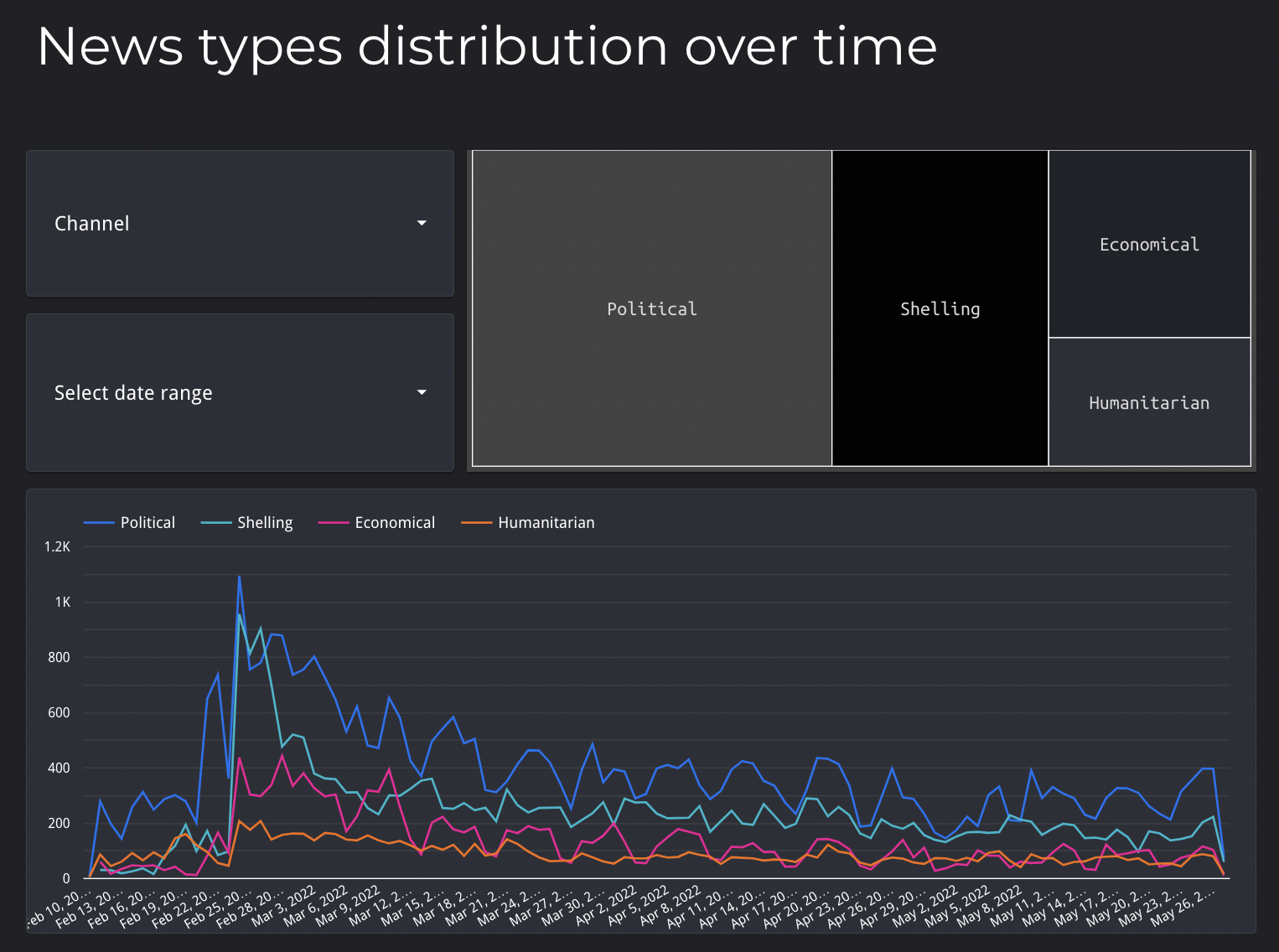


Рисунок 2

Для даного дослідження, ми обрали новини, які були опубліковані в період з 10 лютого 2022 року до 26 травня 2022 року.

На рисунку 2 ми можемо спостерігати те, що найбільша кількість новин про військові дії припадає саме на початок повномасштабної війни, тобто на 24 - 28 лютого, далі ми можемо спостерігати спад. Та ж сама ситуація і з політичними новинами. Щодо економічних та гуманітарних новин, то ми не бачимо занадто різких скачків, однак кількість економічних новин теж дещо зросла на період початку повномасштабної війни.

Дана сторінка візуалізації є інтерактивною, ми можемо обрати окремі канали та певний проміжок часу, а також конкретну групу новин, що хочемо побачити.

Порівняємо публікації з пропагандистського проросійського каналу “Риа Новости” з проукраїнським каналом “Nexta Live” за перший місяць війни.

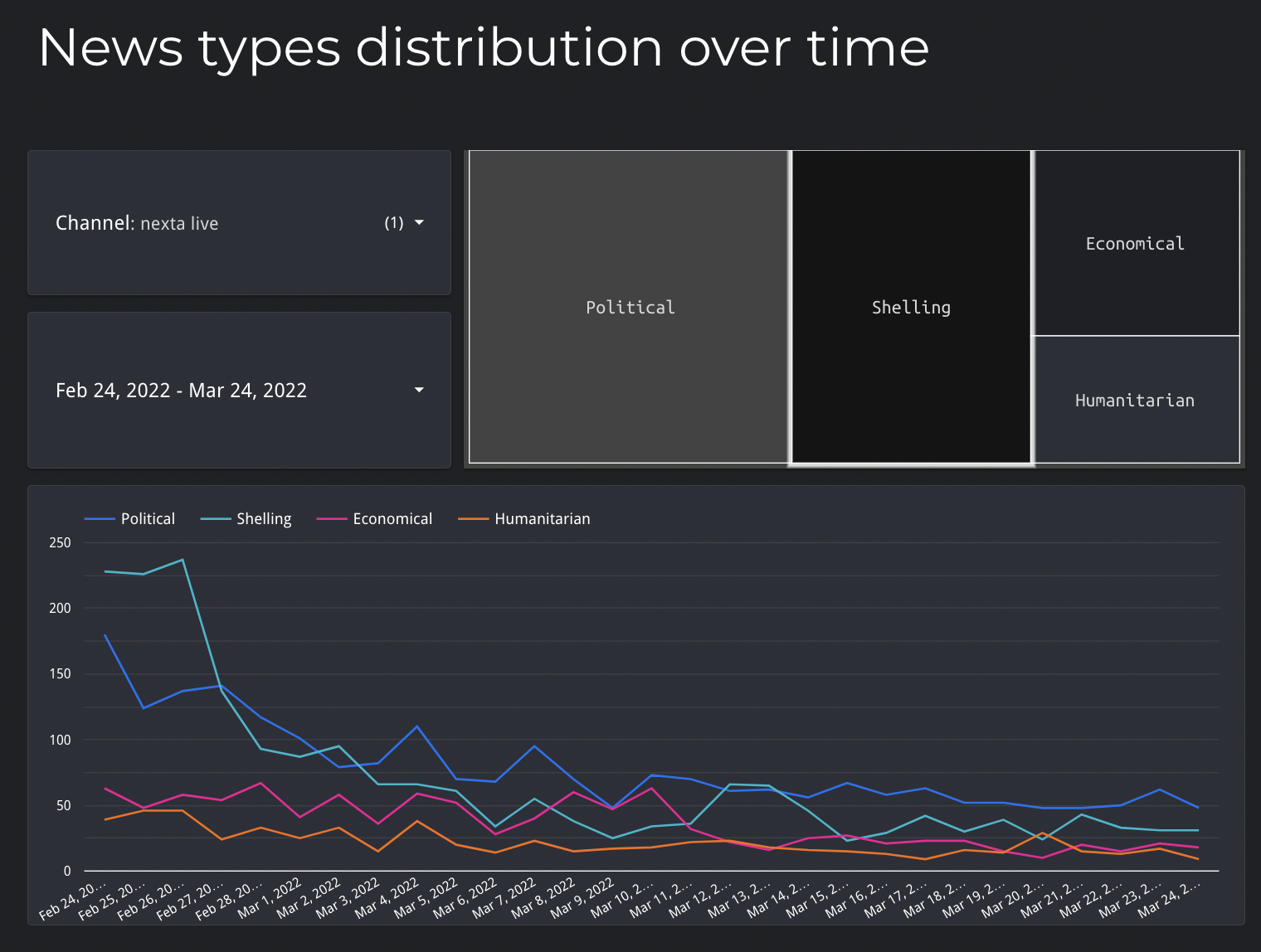


Рисунок 2.1 (Канал Nexta Live)

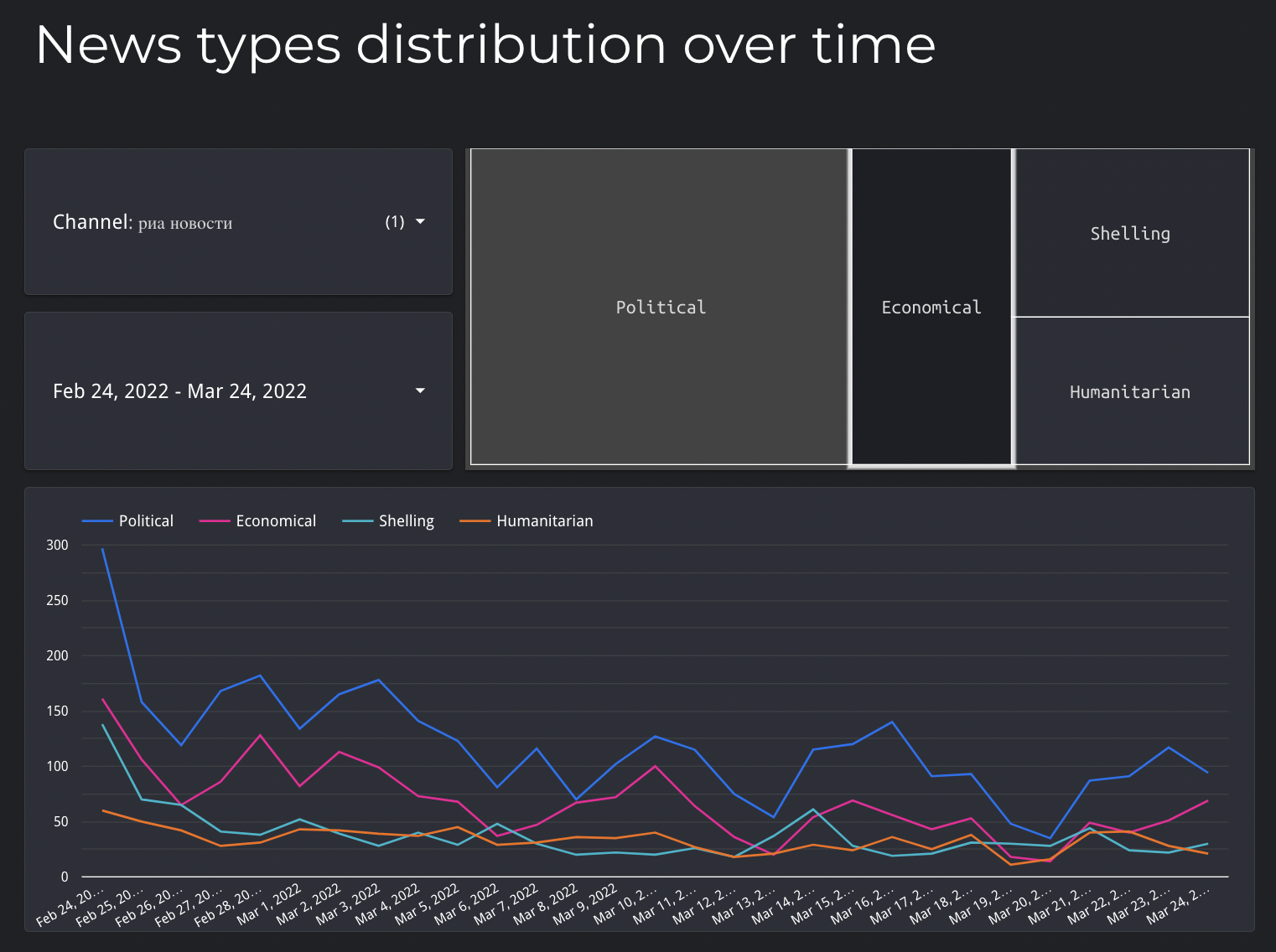


Рисунок 2.2 (Канал Риа Новости)

Як бачимо, в телеграм каналі “Nexta” переважають новини воєнного характеру, в той час, як в “Риа Новости” більшість новин політичні, а частка воєнних новин не є значною.

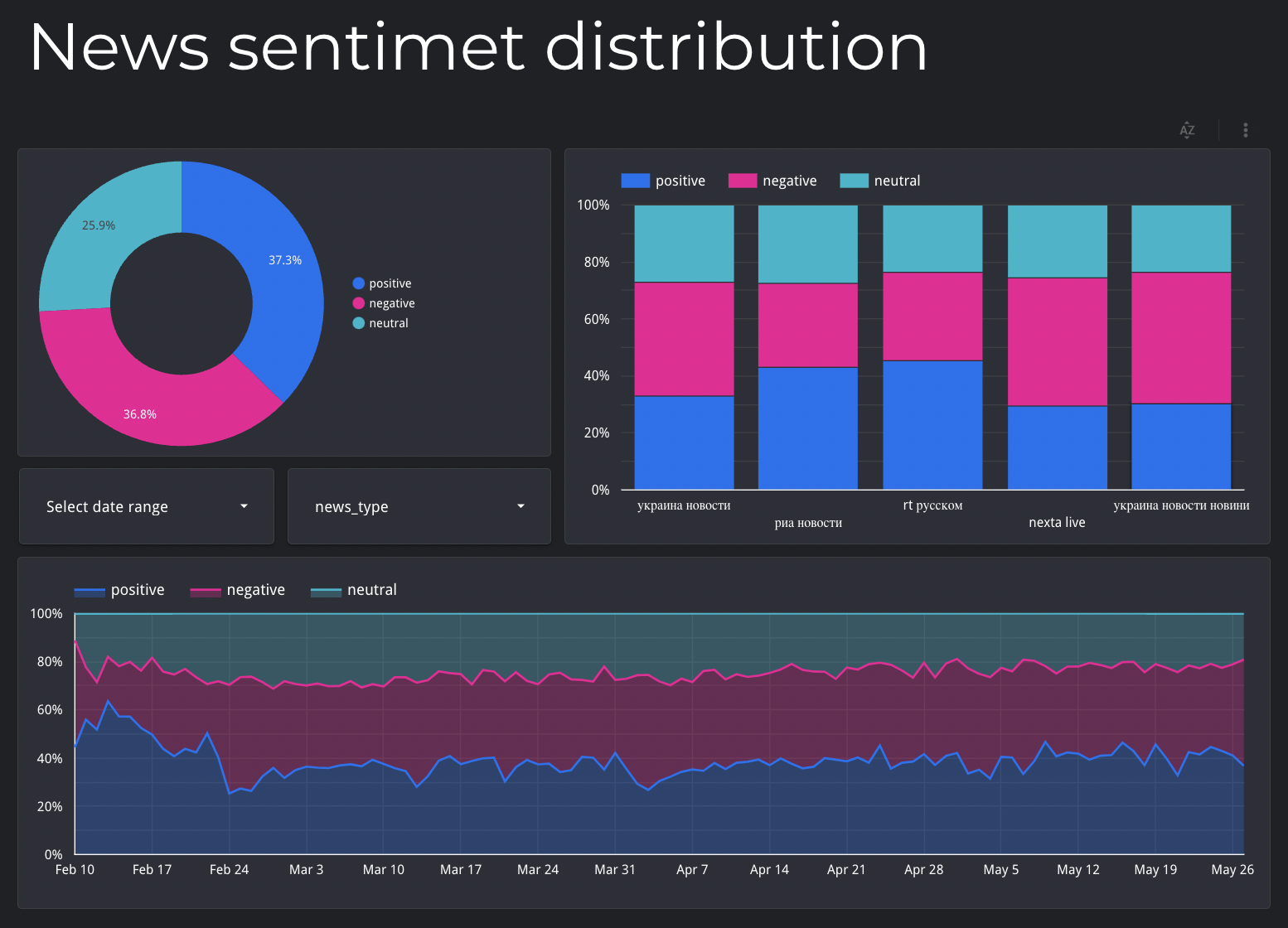


Рисунок 3

На Рисунку 3 показані результати сентиментального аналізу. Можна переглянути відсотки кожного типу новин для різних телеграм каналів, подивитись зміну кількості позитивних, нейтральних та негативних новин в період з 10.02,2022 по 26.05.2022.

Звернувши увагу на гістограму, можна побачити, що в проросійських новинних каналах відсоток негативних публікацій менший, в порівнянні з проукраїнськими каналами. З цього можна зробити висновок, що в російських пропагандистських новинах не використовують яскраво вираженою негативної лексики (наприклад, не “взрыв”, а “хлопок”).

Дана сторінка звіту є інтерактивною, ми можемо виводити дані з необхідними нам групами новин та періодом часу.

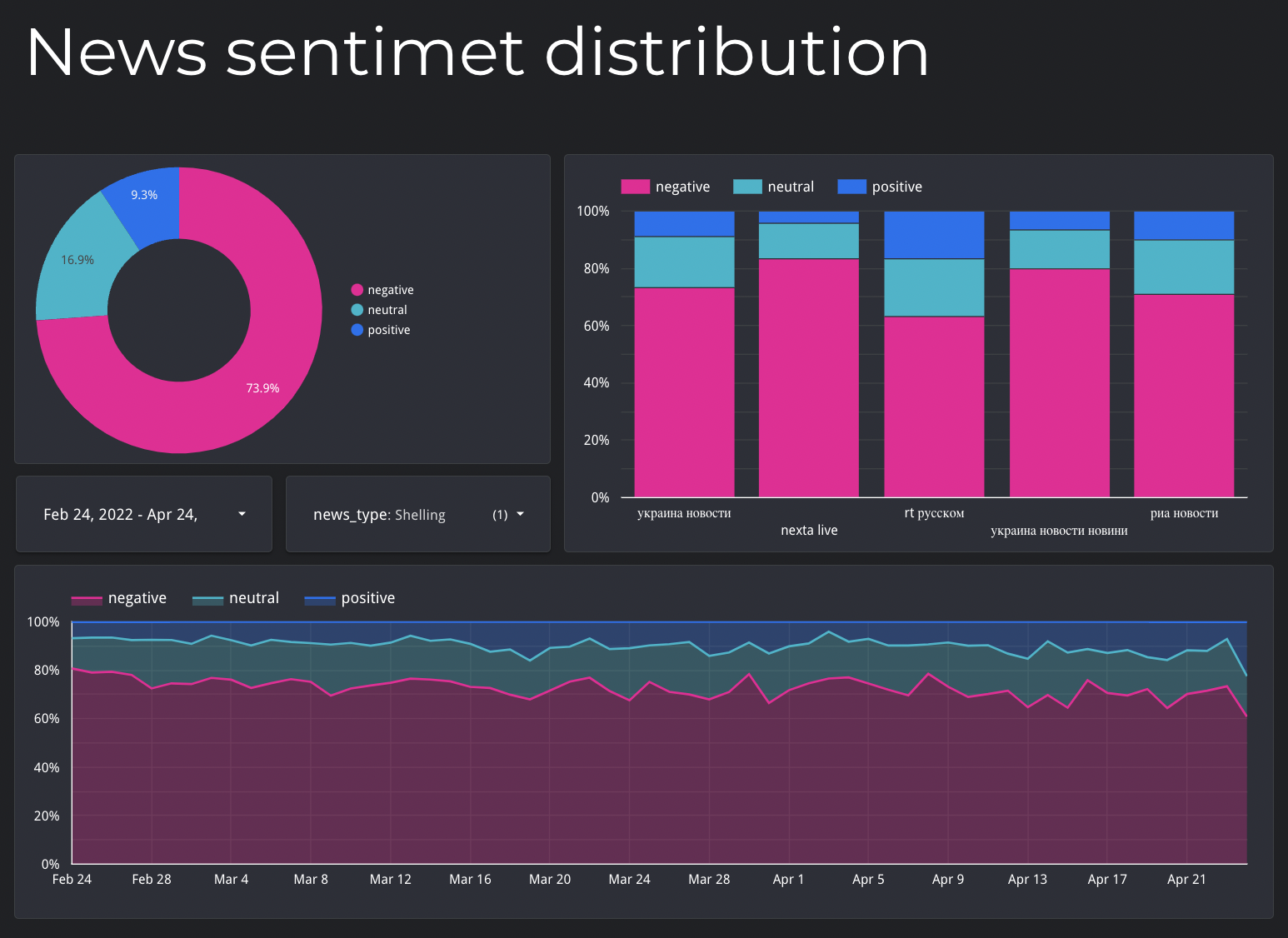


Рисунок 3.1

На рисунку 3.1 зображені дані для типу новин “Shelling” з 24 лютого по 24 квітня 2022 року. Як можемо бачити, в категорії новин, пов’язаний з воєнними діями переважають саме негативні новини.

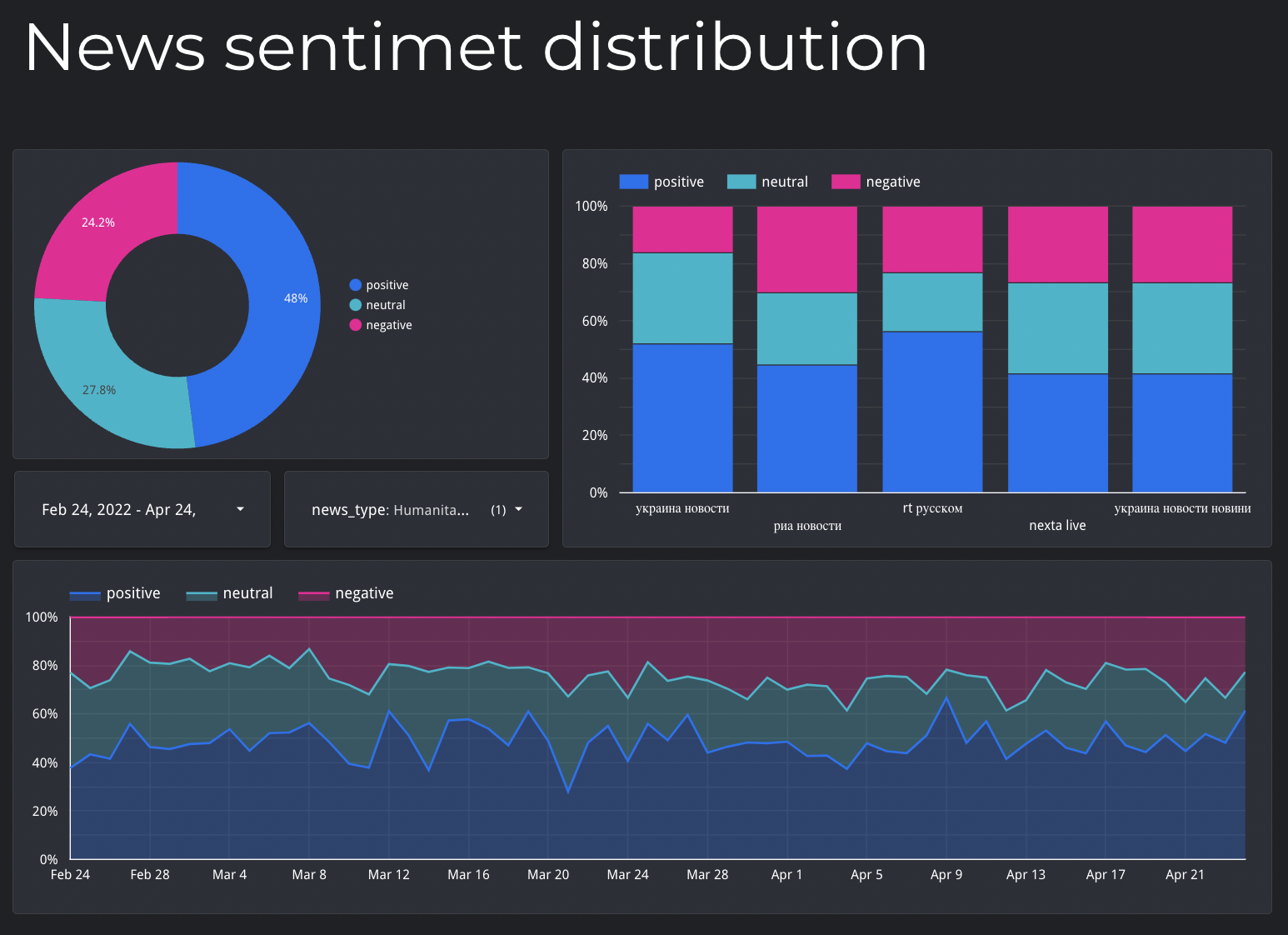


Рисунок 3.2

На рисунку 3.2 зображені дані для новин з категорії “Humanitarian” за той самий період що і в рисунку 3.1. Як бачимо, в цій категорії майже половина новин має позитивний окрас, а кількість новин з негативною тональністю складає найменшу частину.

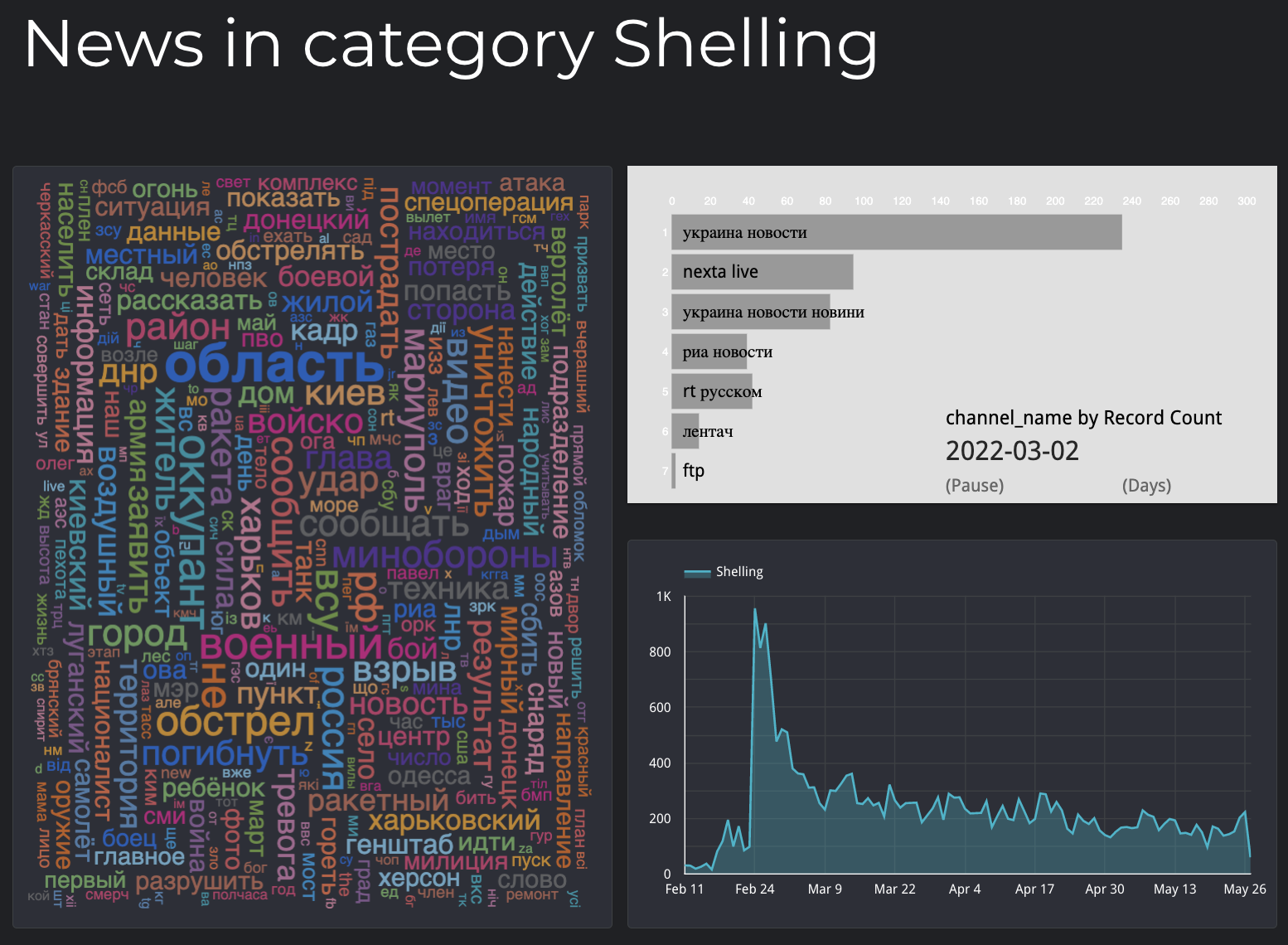


Рисунок 4

На рисунку 4 знаходиться результати дослідження новин в категорії “Shelling”. Ця сторінка звіту також є інтерактивною. Значення гістограми автоматично змінюються, і відображають для різних каналів кількість

1. **ВИСНОВКИ**

В результаті виконання курсової роботи були реалізовані ETL процеси для завантаження даних у файли. Оскільки ми не знайшли готового датасету, в якому б знаходились новини про війну Росії проти України з необхідною нам класифікацією, то ми виконали доволі об’ємну роботу, створивши свій власний датасет. В ньому знаходяться російськомовні новини, де для кожної публікації задано, до якої категорії новини вона відноситься.

Як частину реалізації ETL процесу ми також включаємо передобробку даних. В ході її реалізації ми отримали досвід використання різних бібліотек мови програмування Python, що дуже допомогло у приведенні текстів до необхідного вигляду.

На основі проведеного інтелектуального аналізу даних для прогнозування категорії, до якої належить новина, ми отримали практичні навички аналізу текстів, використовуючи різні моделі машинного навчання та нейронні мережі. Також, ми порівняли результати роботи різних методів прогнозування даних.

Одним із наших спостережень стало те, що алгоритм Decision Tree показує не дуже гарні результати в класифікації текстів. Через те, що цей алгоритм показав значно гірші результати, ніж інші алгоритми, які ми використовували в цій роботі, ми вирішили не враховувати його прогнози в кінцевому ухваленні рішення до якої категорії відносити новину.

Також, в даній роботі ми реалізували сентиментальний аналіз новин.

В ході виконання курсової роботи ми отримали навички з роботою з Business Intelligence засобами, а саме Data Studio, в якій ми виконалами інтерактивний звіт до нашої роботи. В ньому можна наглядно побачити результати дослідження та побачити деякі закономірності та відмінності, наявні між російськомовними проукраїнськими новинами та російською пропагандою.

Отже, ми виконали всі поставлені перед нами задачі та виконали якісне та цікаве дослідження.

ПЕРЕЛІК ПОСИЛАНЬ

1. Посилання на програмну реалізацію курсової роботи - <https://github.com/kinfi4/Telegram-News-Analyzer>
2. Посилання на інтерактивний звіт в Data Studio - <https://datastudio.google.com/u/0/reporting/c76dd534-e89d-4153-a2d9-82d6e5776b6a/page/g2otC>
3. Посилання на словник тональності, що був використаний для сентиментального аналізу - <https://github.com/dkulagin/kartaslov/tree/master/dataset/kartaslovsent>
4. Посилання на матеріали по класифікації текстів та NLP:
   1. <https://monkeylearn.com/text-classification/>
   2. <https://towardsdatascience.com/machine-learning-nlp-text-classification-using-scikit-learn-python-and-nltk-c52b92a7c73a>
   3. <https://developers.google.com/machine-learning/guides/text-classification>
   4. https://www.tensorflow.org/tutorials/keras/text\_classification

ДОДАТОК З КОДОМ А

***ELT pipeline:***

**pipeline.py**

import os

os.environ['TF\_CPP\_MIN\_LOG\_LEVEL'] = '3' *# we need this, in order not to print tensorflow warnings*

from views import fetch\_and\_preprocess\_news, classify\_news, stack\_shelling\_words

if \_\_name\_\_ == '\_\_main\_\_':

print(f'PIPELINE STARTED')

print('-' \* 30)

fetch\_and\_preprocess\_news()

if os.getenv('PIPELINE\_WORK\_MODE', 'FULL') == 'FULL':

print('-' \* 30)

classify\_news()

print('-' \* 30)

stack\_shelling\_words()

print('PIPELINE ENDED')

**views.py**

import os

import csv

from glob import glob

from datetime import datetime

import pandas as pd

from telethon import TelegramClient

from telethon.tl.custom.message import Message

from telethon.tl.types import Channel

import config.config as conf

from config.constants import NewsType

from services.utils import cut\_channel\_link, export\_post\_to\_csv, get\_or\_create\_channel\_file

from services.config\_reader import ConfigReader

from services.domain.text\_preprocessor import TextPreprocessor

from services.domain.predictor import Predictor

client = TelegramClient('session-1', api\_id=conf.API\_ID, api\_hash=conf.API\_HASH)

def fetch\_and\_preprocess\_news():

with client:

client.loop.run\_until\_complete(collect\_posts())

async def collect\_posts():

channels\_to\_export\_raw = conf.CHANNEL\_REGISTRY['CHANNEL\_LIST']

channels\_to\_export\_cut = map(cut\_channel\_link, channels\_to\_export\_raw)

conf\_reader = ConfigReader('./config/.config')

last\_post\_to\_fetch\_date = datetime.strptime(conf\_reader.get(conf.LAST\_POST\_PUBLISH\_DATE), conf.DATE\_FORMAT)

offset\_date\_string = conf\_reader.get(conf.FIRST\_POST\_PUBLISH\_DATE)

offset\_date = datetime.strptime(offset\_date\_string, conf.DATE\_FORMAT) if offset\_date\_string else datetime.now().astimezone(tz=conf.LOCAL\_TIMEZONE)

last\_post\_parsed\_date = offset\_date

processor = TextPreprocessor()

try:

for channel\_name in channels\_to\_export\_cut:

print(f'Starting collecting data from {channel\_name}: ')

entity: Channel = await client.get\_entity(channel\_name)

with get\_or\_create\_channel\_file(channel\_name) as destination\_file\_obj:

csv\_writer = csv.writer(destination\_file\_obj)

message: Message

async for message in client.iter\_messages(entity, limit=conf.MESSAGES\_MAX\_NUMBER\_LIMIT, offset\_date=offset\_date):

if not message.text or len(message.text) < 20:

continue

post\_date = message.date.astimezone(tz=conf.LOCAL\_TIMEZONE)

if post\_date < last\_post\_parsed\_date:

last\_post\_parsed\_date = post\_date

if last\_post\_to\_fetch\_date and post\_date < last\_post\_to\_fetch\_date:

break

export\_post\_to\_csv(csv\_writer, processor, message, post\_date.date())

finally:

conf\_reader.set(conf.FIRST\_POST\_PUBLISH\_DATE, last\_post\_parsed\_date.strftime(conf.DATE\_FORMAT))

def classify\_news():

predictor = Predictor.create\_from\_files(

conf.SENTIMENT\_DICTIONARY\_PATH,

conf.SKLEARN\_VECTORIZER,

conf.KERAS\_TOKENIZER,

conf.KNN\_MODEL\_PATH,

conf.SVC\_MODEL\_PATH,

conf.GAUSSIAN\_MODEL\_PATH,

conf.LSTM\_MODEL\_PATH,

conf.CNN\_MODEL\_PATH

)

destination\_file\_path = os.path.join(conf.CLASSIFIED\_NEWS\_FILE)

with open(destination\_file\_path, 'w') as destination\_file:

csv\_writer = csv.writer(destination\_file)

csv\_writer.writerow(['channel\_name', 'text', 'date', 'news\_type', 'sentiment\_type'])

news\_file\_paths = glob(os.path.join(conf.NEWS\_DATA\_FOLDER\_PATH, '\*.csv'))

for filepath in news\_file\_paths:

filename = filepath.split('/')[-1]

print(f'Start classification for file: {filename}...')

with open(filepath) as source\_csv\_file:

csv\_reader = csv.reader(source\_csv\_file)

*# 0 - channel name, 1 - post text, 2 - post date*

for line in csv\_reader:

channel\_name, post\_text, post\_date = line

news\_type = predictor.get\_news\_type(text=post\_text)

sentiment\_type = predictor.get\_sentiment\_type(text=post\_text, news\_type=news\_type)

csv\_writer.writerow([

channel\_name,

post\_text,

post\_date,

news\_type.value,

sentiment\_type,

])

def stack\_shelling\_words():

print(f'Start stacking words from category: shelling')

df = pd.read\_csv(conf.CLASSIFIED\_NEWS\_FILE)

df = df[df['news\_type'] == NewsType.SHELLING.value]

df = df['text'].str.split()

df = df.apply(pd.Series).stack().reset\_index(drop=True)

df = df[~df.isin(conf.RESTRICTED\_LEMMAS\_FOR\_WORDS\_CLOUD)]

df.to\_csv(os.path.join(conf.PROCESSED\_DATA\_FOLDER\_PATH, 'words-bag-shelling.csv'), index=False)

**utils.py**

import os

import typing

from datetime import date

from emoji import EMOJI\_DATA

from telethon.tl.custom.message import Message

from services.domain.text\_preprocessor import ITextPreprocessor

from config.config import NEWS\_DATA\_FOLDER\_PATH

def cut\_channel\_link(channel\_link: str) -> str:

*"""*

*This function takes channel link in full format (https://t.me/<channel\_name>) and*

*returns it's short form: <channel\_name>*

*"""*

return channel\_link.split('/')[-1] if channel\_link.startswith('https://t.me') else channel\_link

def chat\_name\_standardizer(chat\_name: str) -> str:

*"""*

*This function gets chat\_name in a raw format, and removes all the spaces, invalid characters and so on*

*Example: 24/7 - Новини України🇺🇦 -> 24/7\_Новини\_України*

*"""*

symbols\_to\_remove = ('|', '"', '\'', '-', ',', '.')

new\_chat\_name = chat\_name.translate({ord(c): '' for c in symbols\_to\_remove})

new\_chat\_name = ''.join([c for c in new\_chat\_name if c not in EMOJI\_DATA and ord(c) < 1200])

new\_chat\_name = new\_chat\_name.strip()

new\_chat\_name = new\_chat\_name.replace(' ', '\_')

return new\_chat\_name

def export\_post\_to\_csv(csv\_writer, processor: ITextPreprocessor, message: Message, post\_date: date):

*"""*

*Exports specified message's text into a file using specified csv\_writer.*

*Before that, message.text will be preprocessed using specified processor*

*"""*

channel\_name = message.chat.title

channel\_name = processor.preprocess\_text(channel\_name)

*# post\_text = processor.preprocess\_text(text=message.text)*

post\_text = processor.preprocess\_and\_lemmatize(text=message.text)

csv\_writer.writerow([

channel\_name,

post\_text,

post\_date,

])

def get\_or\_create\_channel\_file(channel\_cut\_name: str) -> typing.TextIO:

file\_path = os.path.join(NEWS\_DATA\_FOLDER\_PATH, channel\_cut\_name + '.csv')

if os.path.exists(file\_path):

return open(file\_path, 'a')

return open(file\_path, 'w')

**config\_reader.py**

class ConfigReader:

def \_\_init\_\_(self, config\_file\_path: str):

self.\_file\_path = config\_file\_path

with open(self.\_file\_path) as conf\_file:

self.\_file\_lines: list = conf\_file.readlines()

def get(self, key: str):

for line in self.\_file\_lines:

tokens = line.split('=')

if len(tokens) == 1 and tokens[0] == key:

return

if len(tokens) > 2:

raise ValueError(f'Line: {line} has a "=" sign in a value, it is forbidden')

line\_key, line\_value = tokens

if line\_key == key:

return line\_value.strip()

raise AttributeError(f'Variable with key: {key} was not found in a config file')

def set(self, key: str, value: str):

for index, line in enumerate(self.\_file\_lines):

line\_key, line\_value = line.split('=')

if line\_key == key:

self.\_file\_lines[index] = f'{key}={value}'

break

else:

self.\_file\_lines.append(f'{key}={value}')

with open(self.\_file\_path, 'w') as conf\_file:

conf\_file.writelines(self.\_file\_lines)

**predictor.py**

import pickle

from abc import ABC, abstractmethod

from collections import Counter

import joblib

import pandas as pd

from keras.models import load\_model

from config.config import MAX\_POST\_LEN\_IN\_WORDS

from config.constants import NewsType

from services.domain.text\_preprocessor import ITextPreprocessor, TextPreprocessor

from services.domain.sentiment import SentimentAnalyzer, ISentimentAnalyzer

class IPredictor(ABC):

@abstractmethod

def get\_sentiment\_type(self, text: str, news\_type: NewsType) -> str:

pass

@abstractmethod

def get\_news\_type(self, text: str) -> NewsType:

pass

class Predictor(IPredictor):

def \_\_init\_\_(

self,

text\_preprocessor: ITextPreprocessor,

sentiment\_analyzer: ISentimentAnalyzer,

\*,

knn\_model,

svc\_model,

gaussian\_model,

lstm\_model,

cnn\_model,

):

self.\_text\_preprocessor = text\_preprocessor

self.\_sentiment\_analyzer = sentiment\_analyzer

self.\_knn\_model = knn\_model

self.\_svc\_model = svc\_model

self.\_gaussian\_model = gaussian\_model

self.\_lstm\_model = lstm\_model

self.\_cnn\_model = cnn\_model

self.\_ml\_models = [self.\_knn\_model, self.\_svc\_model, self.\_gaussian\_model]

self.\_nn\_models = [self.\_lstm\_model, self.\_cnn\_model]

def get\_sentiment\_type(self, text: str, news\_type: NewsType, make\_preprocessing: bool = False) -> str:

if make\_preprocessing:

text = self.\_text\_preprocessor.preprocess\_and\_lemmatize(text)

return self.\_sentiment\_analyzer.define\_sentiment\_type(text, news\_type).value

def get\_news\_type(self, text: str) -> NewsType:

ml\_prediction\_results = self.\_get\_ml\_models\_predictions(text)

nn\_prediction\_results = self.\_get\_nn\_models\_predictions(text)

counter = Counter([\*ml\_prediction\_results, \*nn\_prediction\_results])

most\_common\_prediction = counter.most\_common(1)[0][0]

return most\_common\_prediction

def \_get\_nn\_models\_predictions(self, text: str):

text\_sequences = self.\_text\_preprocessor.keras\_tokenize\_and\_pad\_text(

pd.Series([text]),

make\_preprocessing=True,

max\_words\_number=MAX\_POST\_LEN\_IN\_WORDS,

padding='pre',

truncating='post',

)

*# model.predict is going to return a list with a single value in it*

prediction\_results = []

for nn in self.\_nn\_models:

result = nn.predict(text\_sequences, verbose=0)[0]

prediction\_results.append(result.argmax())

return tuple(map(self.\_get\_predicted\_news\_type\_label, prediction\_results))

def \_get\_ml\_models\_predictions(self, text: str):

test\_vectors\_for\_ml\_models = self.\_text\_preprocessor.sklearn\_vectorize\_text(

[text],

make\_preprocessing=True

)

*# model.predict is going to return a list with a single value in it*

prediction\_results = [model.predict(test\_vectors\_for\_ml\_models.toarray())[0] for model in self.\_ml\_models]

return tuple(map(self.\_get\_predicted\_news\_type\_label, prediction\_results))

@staticmethod

def \_get\_predicted\_news\_type\_label(label\_idx: int) -> NewsType:

labels\_indexes = {0: NewsType.ECONOMICAL, 1: NewsType.POLITICAL, 2: NewsType.SHELLING, 3: NewsType.HUMANITARIAN}

if label\_idx not in labels\_indexes:

raise AttributeError(f'The value of label\_idx must be between 0 and 3, got label\_idx = {label\_idx}')

return labels\_indexes[label\_idx]

@classmethod

def create\_from\_files(

cls,

sentiment\_dictionary\_path: str,

sklearn\_vectorizer\_path: str,

keras\_tokenizer\_path: str,

knn\_model\_path: str,

svc\_model\_path: str,

gaussian\_model\_path: str,

lstm\_model\_path: str,

cnn\_model\_path: str,

):

sklearn\_vectorizer = pickle.load(open(sklearn\_vectorizer\_path, 'rb'))

keras\_tokenizer = pickle.load(open(keras\_tokenizer\_path, 'rb'))

knn\_model = joblib.load(open(knn\_model\_path, 'rb'))

svc\_model = joblib.load(open(svc\_model\_path, 'rb'))

gaussian\_model = joblib.load(open(gaussian\_model\_path, 'rb'))

lstm\_model = load\_model(lstm\_model\_path)

cnn\_model = load\_model(cnn\_model\_path)

text\_preprocessor = TextPreprocessor(sklearn\_vectorizer=sklearn\_vectorizer, keras\_tokenizer=keras\_tokenizer)

sentiment\_analyzer = SentimentAnalyzer(sentiment\_dictionary\_path)

return cls(

text\_preprocessor=text\_preprocessor,

sentiment\_analyzer=sentiment\_analyzer,

knn\_model=knn\_model,

svc\_model=svc\_model,

lstm\_model=lstm\_model,

cnn\_model=cnn\_model,

gaussian\_model=gaussian\_model

)

**config.py**

import datetime

import os

import json

API\_ID = '17332691'

API\_HASH = '626a6ab20ca7d1c151a4d1984448fd0c'

DATA\_FOLDER\_PATH = '/home/kinfi4/python/Propaganda-Analyzer/src/data'

NEWS\_DATA\_FOLDER\_PATH = os.path.join(DATA\_FOLDER\_PATH, 'news-posts')

PROCESSED\_DATA\_FOLDER\_PATH = os.path.join(DATA\_FOLDER\_PATH, 'processed-data')

CLASSIFIED\_NEWS\_FILE = os.path.join(PROCESSED\_DATA\_FOLDER\_PATH, 'classified-news.csv')

SENTIMENT\_DICTIONARY\_PATH = '/home/kinfi4/python/Propaganda-Analyzer/src/ETL/config/sentiment-words.csv'

TRAINED\_MODELS\_PATH = '/home/kinfi4/python/Propaganda-Analyzer/src/models/trained-models'

CNN\_MODEL\_PATH = os.path.join(TRAINED\_MODELS\_PATH, 'cnn-news-type-prediction.h5')

LSTM\_MODEL\_PATH = os.path.join(TRAINED\_MODELS\_PATH, 'lstm-news-type-prediction.h5')

SVC\_MODEL\_PATH = os.path.join(TRAINED\_MODELS\_PATH, 'svc-news-type-prediction.sav')

KNN\_MODEL\_PATH = os.path.join(TRAINED\_MODELS\_PATH, 'knn-news-type-prediction.sav')

GAUSSIAN\_MODEL\_PATH = os.path.join(TRAINED\_MODELS\_PATH, 'nb-news-type-prediction.sav')

SKLEARN\_VECTORIZER = os.path.join(TRAINED\_MODELS\_PATH, 'vectorizer.pk')

KERAS\_TOKENIZER = os.path.join(TRAINED\_MODELS\_PATH, 'keras-tokenizer.pk')

LAST\_POST\_PUBLISH\_DATE = 'LAST\_POST\_PUBLISH\_DATE'

FIRST\_POST\_PUBLISH\_DATE = 'FIRST\_POST\_PUBLISH\_DATE'

DATE\_FORMAT = '%Y-%m-%d %H:%M:%S%z'

MESSAGES\_MAX\_NUMBER\_LIMIT = 70\_000

MAX\_POST\_LEN\_IN\_WORDS = 20

LOCAL\_TIMEZONE = datetime.datetime.now(datetime.timezone.utc).astimezone().tzinfo

with open('/home/kinfi4/python/Propaganda-Analyzer/src/ETL/config/CHANNEL\_REGISTRY.json') as registry\_file:

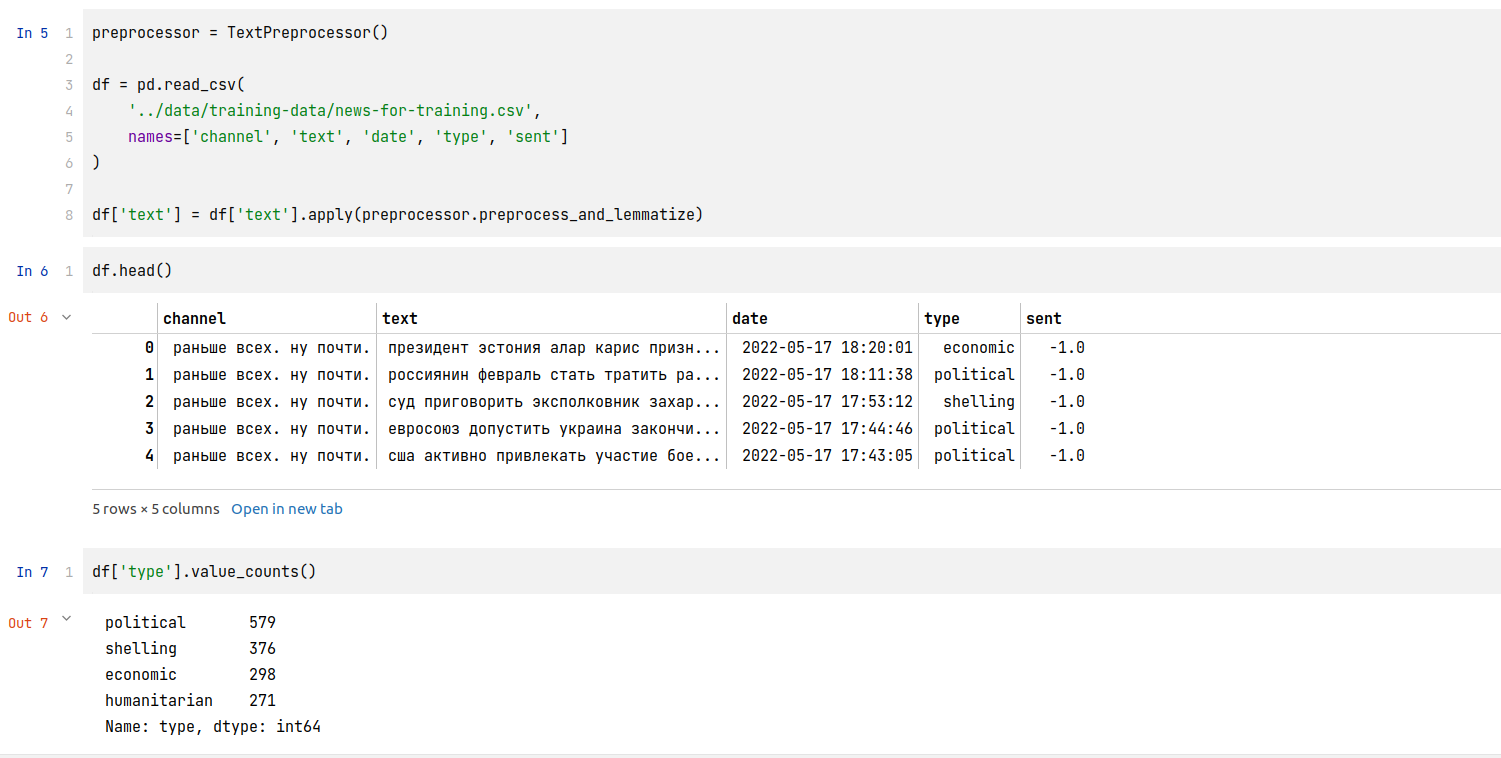
CHANNEL\_REGISTRY = json.load(registry\_file)

with open('/home/kinfi4/python/Propaganda-Analyzer/src/ETL/config/RUSSIAN\_STOP\_WORDS.json') as russian\_stop\_words\_file:

RUSSIAN\_STOP\_WORDS = json.load(russian\_stop\_words\_file)

RESTRICTED\_LEMMAS\_FOR\_WORDS\_CLOUD = ('ус', 'подписаться', 'rtrussian', 'украина', 'российский', 'украинский', 'российскоукраинский')

***Training NN models***

******

