МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
СУМСЬКИЙ ДЕРЖАВНИЙ УНІВЕРСИТЕТ
КАФЕДРА КОМП`ЮТЕРНИХ НАУК

КВАЛІФІКАЦІЙНА РОБОТА МАГІСТРА

НА ТЕМУ:

“Інформаційна технологія оцінки релевантності розміщення реклами в контексті сторінки новин”

Заівідувач випускаючої кафедри **Довбиш А.С.**

Керівник роботи **Шелехов І.В.**

Студент групи ІНм-91н **Тарасов О.О.**

СУМИ 2021

Сумський державний університет

(назва вузу)

Факультет *ЕлІТ* Кафедра *Комп'ютерних наук*

Спеціальність *122 Комп'ютерні науки*

Затверджую:

зав.кафедрою _____

_____

"_____" _____ 20___р.

ЗАВДАННЯ

НА ДИПЛОМНИЙ ПРОЕКТ (РОБОТУ) СТУДЕНТОВІ

Тарасову Олександру Олексійовичу

(прізвище, ім'я, по батькові)

1. Тема проекту (роботи) *Інформацыйна технологія оцінки релевантності розміщення реклами в контексті сторінки новин*

затверджую наказом по інституту від "____" _____ 20 ___ р. № _____

2. Термін здачі студентом закінченого проекту (роботи)_____

3. Вхідні данні до проекту (роботи) _____

_____

_____

4. Зміст розрахунково-пояснювальної записки (перелік питань, що їх належить розробити) *1) Огляд сучасного стану ринку Digital реклами в Україні, проблема негативного контексту; 2) Постановка завдання і формування етапів дослідження; 3)Огляд технологій для скрапінгу, парсингу, роботи із текстом та статистичного навчання на основі текстів; 4)Накопичення корпусу текстів для подальшої обробки; 5)Побудова та збереження моделей машинного навчання; 6) Розробка демоверсії веб-додатка; 7) Аналіз результатів.*

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

_____

_____

6. Консультанти до проекту (роботи), із значенням розділів проекту, що стосується їх

Розділ	Консультант	Підпис, дата	
		Завдання видав	Завдання прийняв

7. Дата видачі завдання _____

Керівник _____

(підпис)

Завдання прийняв до виконання _____

(підпис)

КАЛЕНДАРНИЙ ПЛАН

№ п/п	Назва етапів дипломного проекту (роботи)	Термін виконання проекту (роботи)	Примітка
1.	Огляд сучасного стану сфери Digital реклами	01.10.2020 –01.11.2020	
2.	Постановка задачі та формування завдань дослідження, вибір ресурсів новин	01.11.2020 –15.12.2020	
3.	Аналіз і вибір інструментів	15.12.2020 –02.01.2021	
4.	Створення модуля завантаження та розбору сторінок новин	02.01.2021 –14.01.2021	
5.	Накопичення корпусу документів	14.01.2021 –15.02.2021	
6.	Створення модуля очищення та лематизації тексту	15.02.2021 –01.03.2021	
7.	Вибір, створення та збереження моделей машинного навчання	01.03.2021 – 30.03.2021	
8.	Побудова демоверсії додатку	30.03.2021 – 12.04.2021	
9.	Аналіз результатів	12.04.2021 – 19.04.2021	
10.	Оформлення пояснювальної записки до кваліфікаційної магістерської роботи	19.04.2021 –16.05.2021	

Студент – дипломник _____

(підпис)

Керівник проекту _____

(підпис)

РЕФЕРАТ

Записка: 81 стор., 8 рис., 4 таблиці, 2 формули, 37 літературних джерел.

Об’єкт дослідження — слабоформалізований процес оцінки релевантності розміщення реклами в контексті сторінки новин

Мета роботи — розробити інформаційну технологію для оцінки релевантності розміщення рекламних матеріалів і сконструювати демоверсію веб-додатку.

Результати — було проаналізовано стан Digital реклами в Україні і зроблено висновок необхідності створення технології фільтрації новин із негативним контекстом, бо в іншому випадку контекст впливає на сприйняття користувачем рекламованого бренду. Спочатку було вирішено використовувати мову програмування Python та додаткові бібліотеки для вирішення поставлених завдань. В результаті досліджень було створено модуль для завантаження та розбору сторінок новин. Далі відбувався етап консолідації корпуса текстів для створення датасету на базі вибраних ресурсів новин. Після цього був розроблений модуль очищення та лематизації текстів. Наступним етапом було створення та збереження моделей машинного навчання, які дають можливість визначити негативність теми додатку та емоційну тональність. Для демонстрації роботи додатку було створено веб додаток на основі мікрофреймворку Flask. На заключному етапі було проаналізовано створений додаток і сформовано ряд пропозицій щодо розвитку. Дані пропозиції стосувалися не лише представлення даних та навчання моделей, а також і обслуговування користувачів веб додатком.

ІНФОРМАЦІЙНА ТЕХНОЛОГІЯ ФІЛЬТРАЦІЇ СТАТЕЙ ІЗ НЕГАТИВНИМ КОНТЕКСТОМ, LENTA, MEDUZA, RBK, ИЗВЕСТИЯ, РИА, GENSIM, LOGISTIC REGRESSION, TFIDF VECTORISER, BEAUTIFUL SOUP, FLASK

ЗМІСТ

ВСТУП

Сьогодні важко зустріти людину, яка б не цікавилася новинами. Посилання на такі ресурси часто присутні на початкових сторінках браузерів, у пошукових ресурсах та веб-сторінках із довільним контентом. Вони спокушають цікавими заголовками, перериваються на найцікавішому місці й зваблюють зайти та прочитати хоча б один абзац тексту. Зазвичай такі ресурси є безкоштовними завдяки аналізу смаків користувачів та розміщенню рекламних банерів. Природньо, що негативні новини викликають значну активність користувачів, що призводить до збільшення показу рекламних банерів у контексті статей про війни, вбивства, пограбування, згвалтування, крадіжки та шахрайство. Тепер розглянемо типові угоди на рекламу на інформаційних порталах. Замовник сплачує за кількістю демонстрацій рекламних банерів та кліків. Але при цьому рекламодавець не отримує даних, в якому контексті користувачі сприймають повідомлення про послуги, які він надає. Таким чином кошти на рекламу витрачаються недоцільно, оскільки потенційні клієнти вперше отримують рекламу про послуги у негативній емоційній призмі. Тому, доцільно буде створити технологію оцінки релевантності розміщення реклами в контексті сторінки новин. Метою даної роботи буде створення вказаної вище технології та робочого прототипу як мінімального життєздатного продукту. Дані заходи приведуть до оптимізації використання коштів на рекламу шляхом відмови від демонстрації рекламних банерів на новинних сторінках із небезпечним контекстом.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Дослідження актуальності проблеми

Почнемо із того, що ринок реклами в Україні поступово розширюється, про що свідчать дані із Таблиці 1.1.

Таблиця 1.1 – Стан рекламної сфери в Україні на грудень 2020 року. [2]

	Підсумки 2019р., млн грн	Підсумки 2020р., млн грн	Відсоток зміни 2020 до 2019	Прогноз на 2021р., млн грн	Відсоток зміни 2021 до 2020
ТБ-реклама, всього	**11 527**	**12 175**	**5.6%**	**13 392**	**10%**
Пряма реклама	*10 089*	*10 593*	*5%*	*11 652*	*10%*
Спонсорство	*1 438*	*1 582*	*10%*	*1 740*	*10%*
Реклама в пресі, всього	**1 850**	**1 466**	**- 20.8%**	**1 541**	**5.2%**
Національна преса	*1 106*	*866*	*- 21.7%*	*947*	*9.4%*
в т.ч. Спонсорство	*284*	*215*	*-24.4%*	*238*	*10.5%*
Регіональна преса	*320*	*243*	*-23.9%*	*262*	*7.5%*
Спеціалізована преса	*425*	*357*	*-16%*	*333*	*-6.7%*
Радіо реклама, всього	**717**	**717**	**0%**	**825**	**15%**
Національне	*518*	*512*	*-1%*	*595*	*16%*
Регіональне	*65*	*65*	*0%*	*75*	*15%*
Спонсорство	*134*	*140*	*4%*	*155*	*11%*
OOH Media, всього	**4 240**	**3 159**	**-25%**	**3 695**	**17%**
Зовнішня реклама	*3 283*	*2 433*	*-26%*	*2 799*	*15%*
Транспортна реклама	*600*	*351*	*-42%*	*402*	*15%*
DOOH	*205*	*291*	*42%*	*402*	*38%*
Indoor реклама	*152*	*85*	*-44%*	*92*	*8%*
Реклама в кінотеатрах	**58**	**20**	**-65%**	**26**	**30%**
Digital (Internet) Media реклама	**6 379**	**6 980**	**9%**	**8 977**	**29%**
Всього рекламний медіа ринок	**24 771**	**24 517**	**- 1%**	**28 456**	**16%**

За загальними сумами, що були витрачені на рекламу, можна судити, що найпопулярнішою серед рекламодавців залишається ТБ реклама. При цьому при зміні фокусу на швидкість росту лідером ринку в наших очах стає Digital реклама. Тепер пропонується розглянути дані про ріст саме вказаний реклами в Україні за 2012 – 2020 роки за даними, поданими на Рисунку 1.1.

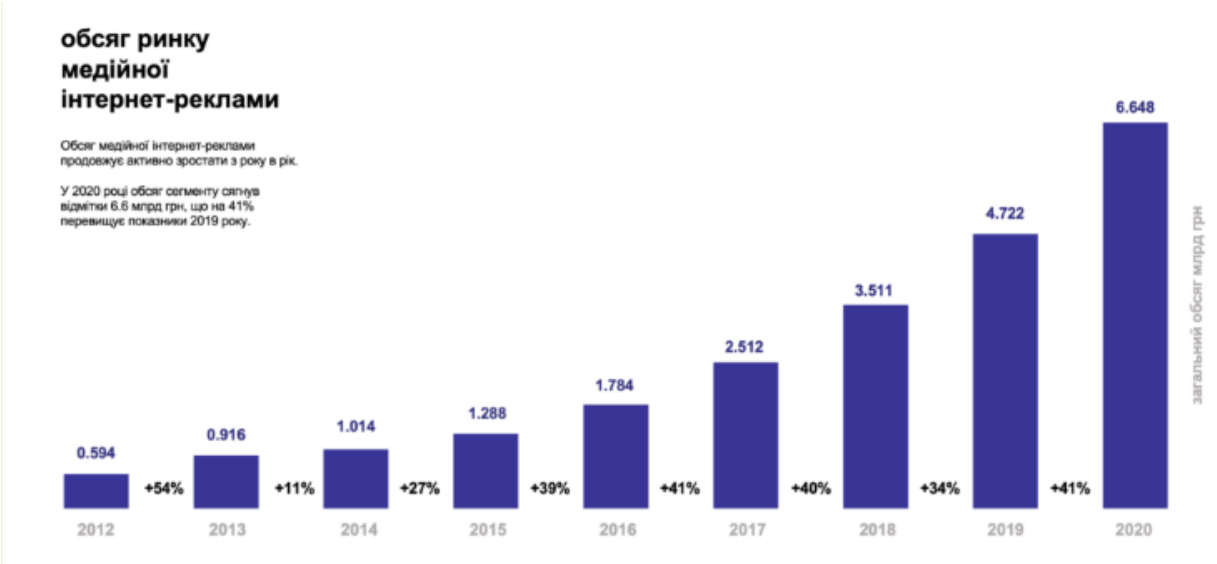

Рисунок 1.1 – Розвиток Digital реклами в Україні за 2012 – 2020 роки [3]

За даними на рис. 1.1 можна константувати бурхливе зростаня Digital реклами, напраму пов`язане із зростанням рівня діджиталізації споживачів. Але при цьому даний тип рекламної діяльності включає в різноманітні види реклами. Розглянемо дані наведені у Таблиці 1.2.

Таблиця 1.2 – Популярні види Digital реклами [1]

Вид реклами	Особливості	Відображення
Контекстна реклама (PPC, Pay Per Click)	Система оплати за клік, використання ключових слів	У пошуковому рядку, на сайтах партнерської мережі провайдера
Ремаркетинг	Можливості відслідковувати користувачів, які відвідали ваш сайт, і показувати їм оголошення повторно	На інших сайтах у медійній мережі
Пошукове просування (SEO, Search Engine Optimization)	Збільшення видимості сайту в пошукових системах, довгостроковий ефект	Пошукові системи
Медійна реклама (банерна)	Здійснюється оплата за тисячу показів чи за клік, можливість використовувати географічний і демографічний таргетинг	На сторонніх сайтах
Реклама в соціальних мережах	Формування та розвиток спільнот навколо сторінки бренду	Великі соціальні мережі
Тізерна реклама	Мікс із медійної і контекстної реклами, використовується яскравий заголовок, текст, який інтригує, і фото, яке привертає увагу	На сторонніх сайтах
CPA-реклама (Cost Per Action)	Оплата з боку рекламодавця проводиться тільки в разі здійснення певних дій із боку клієнта	На тематичних сайтах
Мобільна реклама (реклама на весь екран, продакт-плейсмент, покази на мобільних у PPC)	Звернення до аудиторії через смартфон, користування точними налаштуваннями демографії та географічного таргетингу	Мобільний телефон
Email-розсилка	Може містити спам	Електронна пошта
Створення та просування сайтів, які містять інформаційний і рекламний контент	Потреба в залученні читачів, повний контроль над змістом, безперервність дії	сайт-візитівка, промосайт, корпоративний сайт, портал, Інтернет-магазин

За наведеними даними можна судити, що часто реклама розташовується на сторонніх ресурсах (Ремаркетинг, Медійна та Тізерна реклама). Саме тому одним із способів отримання прибутку для власників інтернет ресурсів є здача в оренду певного місця на веб сторінках під рекламний контент.

Дійсно, із підвищенням рівня діджиталізації всіх сфер в Україні виникла можливість отримувати кошти на створенні медійного контенту. При цьому прибуток залежить від популярності ресурсу, тобто від кількості користувачів, зо користуються сервісом. І в сучасних реаліях одним із найпопулярніших видів контенту можна вказати новинні портали. Для доведення даного твердження наведені дані на табл. 1.3.

Таблиця 1.3 – Рейтинг найпопулярніших сайтів України березень 2021 [4]

CMeter Site: ТОП-25
Березень 2020

	Сайт	Лютий 2020,%	Березень 2020,%
1	google.com	93,43	93,56
2	youtube.com	72,51	73,95
3	facebook.com	64,79	67,92
4	wikipedia.org	53,51	55,94
5	privatbank.ua	41,98	43,04
6	rozetka.com.ua	42,71	41,36
7	ukr.net	39,73	40,52
8	olx.ua	38,96	40,39
9	prom.ua	42,93	40,07
10	rbc.ua	21,95	32,99
11	instagram.com	31,10	32,16
12	sinoptik.ua	32,86	31,29
13	nv.ua	15,70	28,84
14	pravda.com.ua	14,44	24,27
15	tsn.ua	19,02	23,34
16	blogspot.com	16,81	23,04
17	24tv.ua	19,61	22,80
18	minfin.com.ua	12,02	20,08
19	gismeteo.ua	15,25	19,50
20	livejournal.com	13,88	19,42
21	ivi.ru	15,46	18,56
22	obozrevatel.com	14,89	18,53
23	i.ua	16,74	18,01
24	novaposhta.ua	16,61	17,60
25	segodnya.ua	16,55	17,52

KANTAR

Охоплення, desktop та mobile інтернет- користувачі.

Із наведених популярних ресурсів можна виокремити rbc, nv, pravda, tsn. Дані ресурси відносяться до новинних, що доводить популярність даного

виду контенту, що призводить до частого використання цих сервісів, а тому збільшує охоплення аудиторії і підвищує рекламну привабливість.

Розглянемо новини із точки зору власника ресурсу. Найбільшої популярності набирає контент, який знаходить емоційний відгук у користувачів. Не важливо чи є він позитивним(народження слоненяти у національному парку Балі), чи негативним(війни, злочини). Тому відбувається тенденція збільшення долі новин із негативним контекстом. До нього можна віднести політичні скандали, шахрайства, різні злочини, війни, катастрофи. Дані теми підвищують активність користувачів на новинних ресурсах і тому збільшують прибутки від реклами.

Тепер же розглянемо контент із точки зору рекламодавців. Банери із рекламою призначені для спокушання користувачів перейти за посиланням на певний товар або послугу. Рекламу природньо розташовують на сторінках із найбільшою популярністю, що в контексті новинного ресурсу часто означає розміщення поряд із негативним контекстом. Але при цьому часто не враховується емоційний стан користувача при зіткненні із рекламними слоганами. Якщо контекст сторінки викликав у користувача негативні емоції, то цілком ймовірно проеціювання данних відчуттів на бренд, що призводить до зниження популярності контенту. Саме тому необхідно фільтрувати контекст розміщення рекламних оголошень на новиних ресурсах. На жаль, перелік україномовних новинних ресурсів обмежений, тому будемо аналізувати та фільтрувати саме російськомовні новинні ресурси. Для цього використовуватимемо медіа ЗМІ із різних позицій рейтингу російськомовних[36]:

- "Известия"
- Lenta
- Meduza
- "РИА Новости"
- РБК

1.2 Аналіз аналогів та визначення наявних проблем

На даний момент існують онлайн сервіси Microsoft Azure Text Analytics та Amazon Comprehend Features. Дані сервіси використовують глибоке машинне навчання для вирішення таких задач:

- Аналіз тональності
- Розпізнавання мови
- Вилучення головних сутностей
- Аналіз синтаксису
- Класифікація текстів
- Підтримка великої кількості мов

Вказані аналоги успішно вирішують ключові завдання[6, 7], але при цьому існують і складності для їх застосування:

- Багатомовність – у даній роботі концентрація відбувається на російськомовних джерелах, і спеціалізована модель, заточена спеціально під вказану мову, впорається краще, аніж мультимовна
- Пропрієтарність – необхідність пдатити за використання сервісу
- Кастомізація скрапінгового пайплайну – дана робота спрямована на аналіз текстів певних новинних ресурсів, тому необхідно знайти можливість отримання текстів із даного ресурсу, а таких можливостей вказані сервіси не надають
- Кастомізація очищення текстів – часто трапляється, що тексти новин можуть містити одночасто російську мову у комбінації із іншими у вигляді цитат, заяв, що може негативно впливати на результат
- Спрямованість не на конкретні ресурси – у разі створення власного рішення технологія буде навчатися на основі корпусів текстів саме вказаних новинних ресурсів на відміну від готових аналогів

Отже, створення кастомного вузьконаправленого рішення, що буде навчатися на основі корпусів текстів із вказаних російськомовних ресурсів вирішить вказані проблеми.

1.3 Огляд технологій

По-перше, важливою ланкою розробки проекту в даному випадку можна вважати спосіб отримання даних. Деякі ресурси можуть надавати користувачам доступ до архівів старого контенту, особливо якщо це стосується новинних платформ, де інформацію постійно оновлюється. Але при цьому постійно тримати на веб серверах архівовані дані не вигідно, бо вони просто займають місце на диску, що не є ліквідним із точки зору використання ресурсів. Також практична реалізація даної роботи вимагає наявного способу отримання даних, які більшістю не збережені в архіві. Саме тому для вказаних новинних ресурсів варто застосовувати технології скрапінгу та парсингу веб-сторінок. Звичайно можна власноруч завантажувати та розбирати html сторінки, фільтрувати медійний контент, службові теги та властивості, але зручніше це робити за допомогою спеціальних бібліотек(HTML, lxml). Частим явищем є також використання спеціалізованих API для отримання даних та подальшого аналізу JSON файлів.[34, с. 32 - 119] Також помітним явищем є використання власних або нерозповсюджених Front-end фреймворків. У даному випадку часто доводиться затосовувати кастомні прийоми розбору текстів за допомогою регулярних виразів.

По-друге, тексти новин часто містять у собі іншомовні слова або навіть цитати іншою мовою аніж цільова. Саме тому нобхідно очищувати дані перед тим, як використовувати їх. Першим кроком у даному процесі зазвичай вважають розбиття на токени. Зазвичай даний процес доволі легко провести, використовуючи пробільні символи за ідентифікатор кінця слова. Трохи складніше буває із розділенням слів і пунктуації, але у даному випадку нам на допомогу приходять заготовані словники пунктуаційних символів. Після очищення тексту зазвичай проходить і стандартизація форм слів. Даний процес неможливий без глибоких знань у лінгвістиці, тому частіше за все використовують вже сформовані морфологічні аналізатори та словники форм слів. В залежності від цілей можуть використовувати грубу стандартизацію,

що займає порівняно менше часу, але не є ефективною у перетворенні слів, використовуючи прийоми обрізання закінчень суфіксів та префіксів. Іншим, набагато коащим способом є лематизація, шо забирає порівняно багато часу, але при цьому дійсно приводить слово до стандартної форми.[8, с. 68-73] Усі описані дії для досягнення цілей можна проводити і за допомогою вже готових бібліотек, але при цьому при необхідності збереження певних токенів, використовують і частково кастомізазовані пайплайни очищення тексту.

По-третє, вирішимо, що нам необхідно використовувати людську здатність розуміння текстів у поєднанні із шкидкістю роботи комп`ютерної техніки. Саме тому звичайні алгоритми і структури даних не підійдуть, а доведеться обрати алгоритми машинного навчання. Вказані способи обробки текстів переважно працюють із текстовими даними, тому необхідно також використовувати способи представлення текстових даних у числовому форматі. До основних відносят:

- мішок слів – текст представлений у вигляді вектора, де кожна позиція означає певне слово, а значення в позиції – кількість входжень слова до документа
- TF-вектори [8] – кожна позиція у векторі означає певне слово, але при цьому значення показує частоту входжень до документа

$$TF = \frac{n_i}{\sum_k n_k} \tag{1.1}$$

- TF-IDF-вектори [8] – схожий на TF вектор, але при цьому слова мають коефіцієнти важливості, що обернено пропорційні до кількості входження до документів корпуса

$$IDF = log \frac{|D|}{|d_i \supset t_i|} \tag{1.2}$$

$$TF - IDF = TF \cdot IDF$$

- Відображення у вигляді щільних векторів – унітарне представлення слів у вигляді мішка слів, TF або TF-IDF створює матриці великих розмірностей, де кожен стовпець означає входження певного слова, кожен рядок – тексту у корпусі. Такі представлення корпусу текстів призводять до необхідності використання великих об`ємів обчислювальних ресурсів при тренуванні моделей машинного навчання. Набагато легше використовувати занурення ознак. Для використання цього методу необхідно спочатку виділити необхідні ознаки(слова), для кожної ознаки знайти спосіб перетворення у n-мірний вектор, а потім за допомогою суми, конкатенації або комбінування даних методів, застосованих до ознак-векторів, отримати фінальний вектор і подати його на вхід до нелінійного класифікатора(зазвичай ним є нейронна мережа).[9, с. 100 - 112]

По-четверте, як уже було сказано ми будемо використовувати технології машинного навчання. Для поставленої цілі будемо використовувати навчання із учителем. Даний розділ машинного навчання характирезується тим, що вхідні дані мають значення змінних(певних вимірів) та відгуків. Існує певна залежність між вимірами та відповідями і алгоритму необхідного його апроксимізувати із мінімальною похибкою. Існують параметричні і непараметричні методі алгоритми машинного навчання. Зазвичай параметричні моделі зосереджені на побудові певної поверхні у просторі ознак, яка допоможе вирішити питання класифікації або прогнозування певного значення цільової змінної. Дані моделі обирають, коли вже мають певні знання про природу досліджуваного явища. При цьому із збільшенням даних для навчання складність моделі ніяк не змінюється, лише підбираються кращі коефіцієнти. Яскравими прикладами таких моделей SVM, логістична регресія, перцептрон. Непараметричні моделі можуть працювати за відсутності знань про дані, але при цьому є великий

шанс пристосування до певного сету даних, що помилка на нових вимірюваннях буде досить значною. Такі випадки називаються перенавчанням. При цьому для гарного навчання непараметричних моделей необхідно використовувати дійсно велику кількість даних, але при цьому із порівняно невеликою розмірністю простору ознак. Часто для покращення результатів непараметричних моделей використовують їх ансамблі. Дані моделі є більш точними й ефективно протистоять перенавчанню, але при цьому потребують багато часу, обчислювальних потужностей і даних для навчання.[10, с. 96 - 145]

1.4 Висновки по розділу 1

Із усього вказаного вище можна зробити висновок, що необхідно створити технологію визначення релевантності розміщення рекламних матеріалів саме в контексті сторінок перелічених російськомовних новинних ресурсів. Існуючі рішення мають певні недоліки, які планую виправити у даній роботі. Будуть використані сучасні засоби скрапінгу, очищення та векторизації текстів. Векторизовані ознаки в свою чергу будуть використані для тренування моделей машинного навчання, які задовольняють критерії імітації сприйняття текстів людиною зі швидкістю роботи комп`ютерної техніки.

2 ОГЛЯД АРХІТЕКТУРИ ТА ВИБІР МЕТОДІВ РЕАЛІЗАЦІЇ

2.1 Постановка задачі

Розробити технологію оцінки безпечності контексту статті на російськомовних джерелах новин

- "Известия"
- Lenta
- Meduza
- "РИА Новости"
- РБК

Також представити демо версію веб-додатку із формою для введення посилання на статтю. Після обробки статті користувач має отримати інформацію щодо того чи є дана стаття безпечним контекстом для розміщення рекламних матеріалів.

2.2 Етапи роботи

Для ефективного ведення робота була розділена на етапи. Успішне завершення кожного попереднього етапу є необхідною умовою початку наступного. У списку нижче будуть подані основні етапи роботи, дії в них та умови завершення.

- Проектування скраппера новинних ресурів – на даному етапі розроблюється програмний продукт для накопичення та виділення текстів із html коду із вказаних російськомовних новинних ресурсів. Умовою завершення є робота на довільному посиланні не більше за 10 секунд.
- Консолідація текстів – у цьому проміжку часу відбувалося збирання текстів. Умовою завершення є збір не менш як 5 тисяч документів
- Розробка пайплайну очищення та лемматизації тексту – тексти очищуються від пуктуації, типових слів, очищення одного документа має тривати не більше за 10с

- Створення словників – створення моделі відображення слів у вектори не менш як розміром у 50 тисяч слів
- Розмітка данних – визначення ставлення у кожному із текстів, виділення тонального словника
- Моделювання – побудова векторизатора тексту, визначення та тренування моделі не менш як із 80% точністю
- Побудова веб-додатку – на даному етапі будується демо версія додатку із використанням натренованих моделей

2.3 Вибір мови програмування

Розглянемо вибір мови програмування для кожного із етапів роботи.

Найперше, що потрібно зробити за планом – це створити програмний модуль для збору даних із новинних ресурсів та розпарсити їх вміст. Однією із найпопулярніших у даній сфері є мова програмування Python із бібліотекою Beautiful Soup[15]. Також значним плюсом можна вважати асинхронне виконання запитів за допомогою пакету Aiohttp[28].

Враховуючи подальшу роботу із машинним навчанням, матрицями та векторами чисел, будемо також використовувати мову програмування Python, яка містить в собі бібліотеку Numpy для роботи із великими масивами даних[16], scikit-learn для моделей машинного навчання та створення TF, TF-IDF векторів на основі корпусу текстів[18], genism для створення відображень слів у вектори та пошуку схожостей між словами[24].

Третій і заключний етап побудування демо версії додатку для використання даної технології. Нам необхідна всього одна веб сторінка для запиту у користувача посилання на статтю і демонстрації розультатів обчислень. При цьому необхідно швидко розпаковувати та використовувати створені за допомогою Python моделі машинного навчання та вектори, проводити завантаження та парсинг текстів статей за допомогою уже створених модулів, очищення їх від непотрібних даних. Природньо, що для цих цілей також буде обрана мова програмування Python. Зважаючи на

розмір додатка будемо використовувати мікрофреймворк Flask, оскільки він дає можливість швидко і гнучко запустити вказаний веб додаток, не конфігуруючи інші середовища розробки та запуску[30].

2.4 Інструменти

По-перше, дані із вказаних ресурсів потрібно буде завантажувати в першу чергу для аналізу та тренування моделей. Також у фінальній демонстрації додатка планується отримувати від користувача саме посилання на сторінку, контент якої варто перевірити. Саме тому необхідний модуль для вилучення тексту статті. Таким чином нам необхідні бібліотеки завантаження даних і ми використаємо requests для спроб синхронних завантажень[13], aiohttp для аналогічних асинхронних дій[28]. Для парсингу html сторінок планується використання бібліотеки Beautiful Soup[15].

По-друге, збір даних на початкових етапах роботи потребує різних підходів. Деякі ресурси надають API доступ, інші можуть вказати посилання на зібраний в архів корпус статей. Але також існують ресурси, для яких необхідо використовувати саморобний кроулер, який планується створити із використанням кастомного скаппера та бібілотеки Selenium для імітації дій браузера [14].

По-третє, після збору корпусу текстів необхідно буде провести аналіз отриманих документів. Для цього використовуватимемо бібліотеку NLTK як сховище даних для стоп слів і пунктуації[17]. Бібліотеки Pandas та Numpy використовуватимуться для зручного збереження та оперування даними[33, 16]. Нарешті Scikit-learn містить у собі набір векторизаторів для зручного представлення частот слів[18], що буде використано при аналізі найпопулярніших слів і фраз. Для зручної ілюстрації буде використано бібліотеку Seaborn[35].

По-четверте, отримані тексти необхідно буде очистити від нецільової мови та знаків пунктуації. Для таких цілий підійде вже згадана бібліотека NLTK. При цьому планується поекспереметувати із швидкодією

саморобного модуля очищення та вже наявних рішень із бібліотек Catboost[23], stop-words[21] та Natasha[25]. Окреми завданням стоятиме стандартизація токенів і тому планується поекспереметувати із використанням стемерів та лематизаторів із уже вказаних Nltk та Natasha. Також перспективними у даному питанні виглядають Pymorphy2[22] та Pymystem[20].

По-п`яте, на основному етапі планується використати бібліотеку Gensim[24] для представлення слів та текстів у вигляді щільних векторів та Scikit-learn для тренування моделей машинного навчання.

Останнім етапом проекту можна назвати конструювання демоверсії. Для цих цілей найкраще підійде синхронний мікрофреймворк Flask[30], так як планується використання лише однієї сторінки. Для валідації даних всередині програми використаємо Pydantic[26].

Отже, загаьльний список використаних ресурсів:

- requests
- Aiohttp
- BeautifulSoup
- selenium
- Numpy
- Pandas
- Seaborn
- NLTK(Natural Language Toolkit)
- Scikit-learn
- Natasha
- Stop-words
- CatBoost
- Pymystem
- Pymorphy2
- Gensim

- Pydantic
- Flask

2.5 Висновки по розділу 2

У даному розділі було поставлено головну задачу роботи – створення технології визначення релевантності розміщення рекламного контенту у контексті сторінки російськомовних новин перелічених раніше ресурсів. Було визначено основні етапи роботи, що являють собою створення засобів сбору корпусу текстів, збір даних, розмітка, формування словників, навчання та збереження моделей, демонстрація у вигляді демо. У розділі наведено причини вибору мови програмування Python для реалізації кожного із етапів і суміщення їх у фінальній версії. Для кожного із етапів було наведено необхідні бібліотеки, які планується використовувати.

3 ПРАКТИЧНА РЕАЛІЗАЦІЯ

3.1 Побудова модулів для скрапінгу

Для скрапінгу даних для кожного обраного російськомовного ресурсу було побудовано свій модуль, який приймає на вхід посилання на сторінку новин і повертає текст статті. Для отримання посилань на статті було використано бібліотеку Selenium. Завантажувачем даних на початку розробки виступала синхронна бібліотека requests, але згодом вона була замінена на асинхронну aiohttp. Для парсингу використовувалася бібліотека Beautiful Soup. Деякі ресурси мають API для завантаження, що також було передбачено, наприклад, для сервісу Meduza . Приклади написаних модулів знаходяться в Додатку.

3.2 Збір корпусу текстів

Після створення модулю для завантаження і парсингу веб сторінок обраних ресурсів новин природньо використати даний написаний код в дії. В результаті роботи було консолідовано 8674 статей новин. У Додатку можна знайти перші 400 посилань на статті із сервісу Lenta. Для кращого розуміння датасету проведемо його поверхневий аналіз та здобудемо основні статистичні показники. Побудуємо для початку таблицю, що відобразить кількість унікальних та типових слів, кількість букв, знаків пунктуації, слів із великої літери, тощо. Далі проілюструємо топ 10 типових і нетипових слів корпусі. Але при цому окремі слова самі по собі мало що можуть означати. Для цього також наведемо дані топ найбільш вживаних біграм. Такі ж дії зробимо і для триграм. Ілюстрація виразів уже із 3, 4 , 5, ... не має серйозного значення, тому не наводитимемо їх рейтинги.

Таблиця 3.1 – Описова статистикаі отриманного корпусу документів.

	Кількість слів	Кількість слів із великої літери	Кількість унікальних слів	Кількість типових слів	Кількість букв	Кількість знаків пунктуації
count	8674.000000	8674.000000	8674.000000	8674.000000	8674.000000	8674.000000
mean	255.963454	37.670625	174.052686	70.537699	1841.712359	65.874107
std	668.084705	97.928443	336.817141	184.712055	4807.146135	173.225634
min	43.000000	4.000000	39.000000	5.000000	298.000000	8.000000
25%	128.000000	17.000000	104.000000	33.000000	924.000000	31.000000
50%	157.000000	23.000000	124.000000	43.000000	1133.500000	40.000000
75%	193.000000	30.000000	149.000000	55.000000	1392.000000	51.000000
max	6440.000000	931.000000	3180.000000	1764.000000	46902.000000	1763.000000

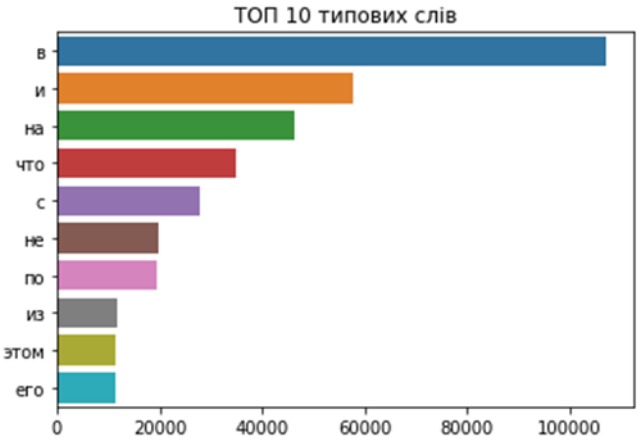

Рисунок 3.1 – Найпопулярніші типові слова

Рисунок 3.2 – Найпопулярніші нетипові слова

Рисунок 3.3 – Найпопулярніші біграми

Рисунок 3.4 – Найпопулярніші триграми

3.3 Розбір пайплайну очищення тексту

Як уже було продемонстровано у попередньому пункті отриманий корпус текстів містить багато пунктуаційних знаків та типових слів. Ці домішки можуть спотворити результат навчання, тому їх необхідно вилучити. Для цього використовуємо бібліотеку nltk. Спочатку завантажуємо дані пунктуації та стоп-слів. Перевіряємо наявні токени із тексту на входження в дані множини. Токени, що не належать до цих масивів залишаються у тексті. Після вилучення домішок необхідно привести різні форми слів до єдиного виду. Для цього ідеально підходить процедура

лематизації, для якої використовуємо морфологічні аналізатори Pymystem та Pymorphy2.

3.4 Створення словників

Даний процес відбувався нероздільно із маркуванням текстів. Під час маркування відбувався розбір текстів і виявлялися слова, які вживаються переважно у контексті описутоксичних тем. Саме такі слова були винесені в окремий словник.

3.5 Розмітка корпуса

Були проаналізовані тексти корпуса і кожному була присвоєна одна із міток. Негативна тональність(засудження), нейтральна тональність (констатація фактів), позитивна тоналність (підтримка). Як уже було описано у попередньому пункті детальний огляд текстів був спрямований до побудови словників негативного контексту, які надалі будуть використовуватися для визначення конексту статті новин.

3.6 Моделювання

У даному пункті будуть використані дані із попередніх кроків для тренування моделей машинного навчання. Для початку подумаємо над тим, як саме визначити, що контекст веб сторінки є неприйнятним для розміщення рекламних матеріалів? По-перше, сама тема статті може викликати негативні емоції у читача. До таких тем можна сміливо віднести війну, злочини, політичні скандали, пропаганду. По-друге, важливу роль відіграє емоційне забарвлення тексту. Зазвичай новини мають нейтральний характер, але все ж трапляються статті із різкою підтримкою, або навпаки запереченням. Загальний настрій автора легко передається читачу, тому необхідно враховувати і даний параметр. Ось чому потрібно розбити аналіз тесту статті на два етапи:

- визначення ступеня токсичності теми
- емоційне забарвлення

Для того, щоб визначити рівень токсичності теми будемо використовувати концепцію вкладень слів. Кожному слову буде відповідати вектор фіксованного розміру. Відстань між векторами слів відповідає на питання наскільки відрізняються за сенсом слова. Для організації вказанного принципу використовуємо клас Word2Vec із бібліотеки genism. Для навчання данної моделі використовуємо неочищений корпус, оскільки Word2Vec оснований на нейронній мережі, то модель має сама розібратися із зв’язком побудови слова та його вектором.

Для визначення тональності будемо використовувати стандартні алгоритми навчання із учителем. Наразі виділяють два типи вказанних моделей:

- параметричні
- непараметричні

Для навчання необхідно очистити та лематизумати текст за допомогою вже створеного пайплайну. Наступним кроком буде створення TfIdf моделі для представлення кожного тексту у вигляді вектора частот входжень слів. Отже, матимемо велику кількість фіч. Ала при цьому нагадаємо, що зібрано лише 8674 тексти, що є порівняно малою кількістю для навчання. Саме тому відпадає можливість використання непараметричних моделей через “прокляття розмірності”. Тому нам доведеться використовувати саме параметричні моделі машинного навчання. В процесі відбору була обрана модель логістичної регресії. Для реалізації навчання була використана модель LogisticRegression із бібліотеки scikit-learn. У результаті крос-валідації за F2 мірою було отримало модель із точністю 81%. Побудовані моделі були збережені для демонстрації у демо-додатку.

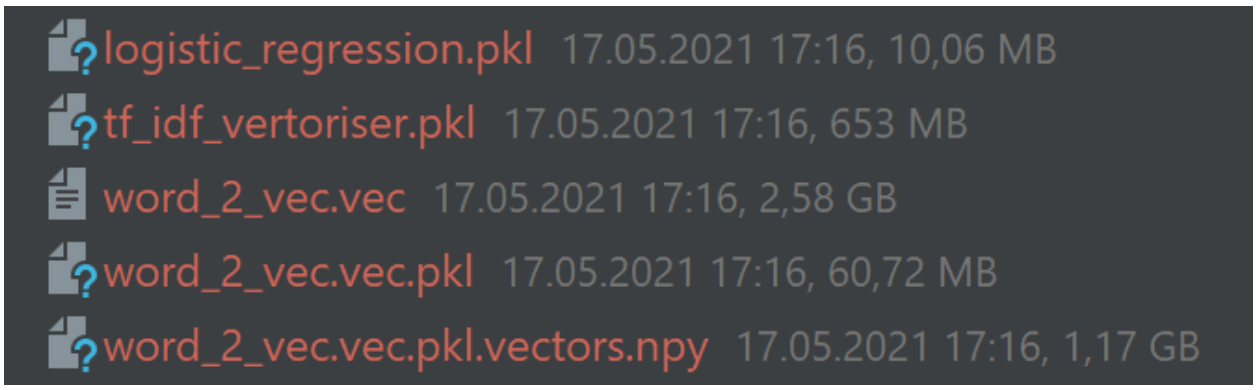

Рисунок 3.5 – Створені моделі

3.7 Програмна реалізація та тестування демоверсії

Для сторення демонстрації використовуватимемо мікрофреймворк Flask. На початку роботи розпаковуватимемо моделі в оперативну пам`ять і вони залишатимуться там до зупинки програми. За планом користувач має отримати спочатку форму для введення посилання на сторінку, яку необхідно перевірити. Після відправлення форми через деякий час користувач отримає на екрані три прогрес бари із:

- оцінкою безпечності теми
- найважливішими сутностями тексту
- оцінкою тональності тексту

Визначення порогу релевантності розміщення реклами залишається за користувачем, тобто наскільки небезпечні статті користувач готовий прийняти для збільшення кількості показів рекламних матеріалів.

В результаті практичної реалізації даної роботи було створено модулі для скраппінгу та парсингу веб-сторінок вказаних російськомовних новинних сервісів. У результаті роботи даного модуля було отримано корпус текстів із 8674 статті. Було створено модулі для очищення, видалення типових слів та лематизації текстів. Після проведення всіх необхідних підготовчих заходів було навчено та збережено моделі

- Word2Vec із бібліотеки genism
- TfIdfVectoriser та LogisticRegression із бібліотеки scikit-learn.

Для демонстрації роботи із даними моделями було створено демоверсію додатку на основі мікрофреймворка Flask.

При першому відвідуванні сторінки отримуємо веб форму показану на рис. 4.1. Форма містить поле для вводу посилання на необхідну сторінку та кнопку для ініціації обробки.

Рисунок 4.1 – Стартова сторінка додатку для введення посилання на статтю

Для тесту використаємо https://lenta.ru/news/2021/05/19/poehali/.

В результаті отрумуємо результати обробки ілюстровані на рис. 4.2.

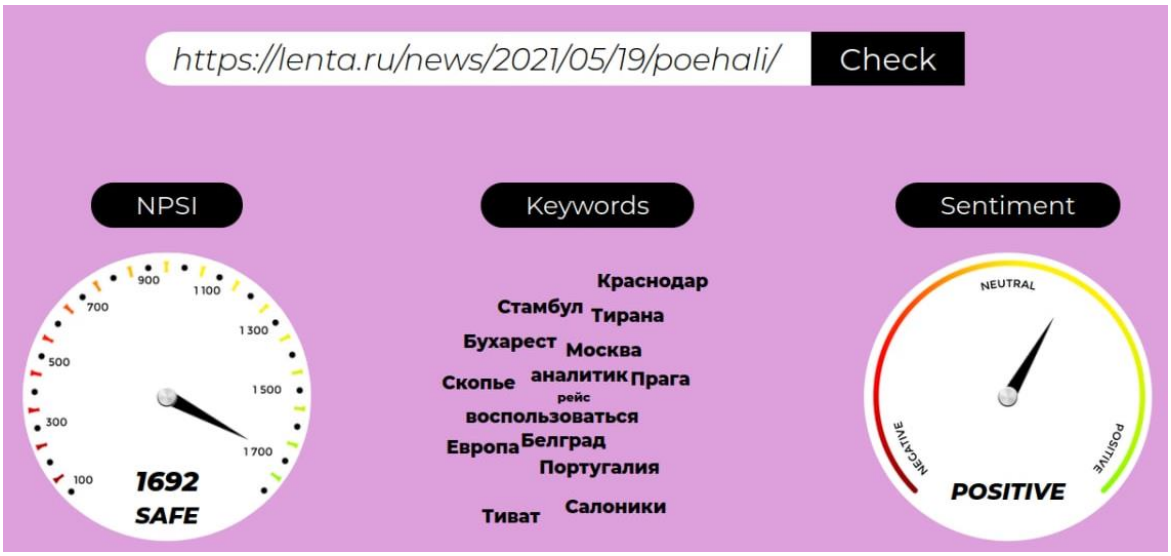

Рисунок 4.2 – Результат обробки статті

На сторінці ми бачимо три параметри. Перший NPSI(news posting safety index) ілюструє наскільки стаття схожа на небезпечну. Градація відбувається за такими правилами:

- 30-600 – небезпечна

- 600-1200 – нейтральна
- 1200 – 1800 – позитивна

Параметр Keywords ілюструє найважливіші слова серед тексту. Параметр Sentiment означає емоційне забарвлення тексту.

3.8 Висновки по розділу 3

У даному розділі було розглянуто роботу демо версії технології визначення релевантності розміщення рекламних матеріалів в контексті статті нових із перелічених російськомовних новинних ресурсів. На початку роботи користувач вводить посилання на статтю в відповідне поле для вводу. Далі варто натиснунути клавішу і чекати. Результат містить у собі три параметри. Перший – NPSI(news posting safety index) – означає безпечність теми статті. Другий параметр Keywords ілюструє головні слова у даному тексті. Третій параметр Sentiment показує емоційне забарвлення. Остаточне рішення щодо розміщення рекламних оголошень все ж приймає користувач на основі параметрів відповіді.

ВИСНОВКИ

За результатами аналізу ринку Digital реклами було виявлено проблему розміщення рекламних матеріалів у контексті статей новин із негативною темою та емоційним забарвленням. Таким чином було вирішено створити технологію визначення релевантності показу банерів реклами на певних сторінках новинних сервісів. Зацільові ресурси було обрано російськомовні сервіси:

- "Известия"
- Lenta
- Meduza
- "РИА Новости"
- РБК

Під час аналізу інструментів було обрано мову програмування Python із широким набором бібліотек, що повністю відповідають потребам кожного із етапів створення проекту.

Під час проектування технології було створено програмні модулі для скрапінгу та парсингу веб сторінок вказаних ресурсів. Наступним етапом роботи було формування корпусу текстів для навчання алгоритмів. У процесі обробки отриманого датасету було створено пайплайн для очищення від пунктуації та типових слів, а також для лематизації. В результаті моделювання було створено та збережено

- модель Word2Vec для вимірювання близькості даного тексту до негативного контексту із використанням словника негатичних слів
- модель TfIdfVectoriser для відображення очищеного тексту у вектори частот
- модель LogisticRegression для класифікації емоційного забарвлення із 80% точністю

Для ілюстрації роботи технології було побудовано демоверсію у вигляді веб додатка на баз'і мікрофреймворка Flask. Враховуючи результати наведемо шляхи покращення даного додатку

- використання більшого корпусу текстів для навчання
- врахування позицій слів у тексті шляхом застосування LSTM або GRU архітектур нейронних мереж для навчання тонального аналізатора (GRU архітектура простіша за LSTM але і дані запам`ятовуються на коротший час)
- у зв`язку із планами на використання нейронних мереж зупинити видалення немовних знаків із початкових статей за причиною можливого кодування інформаціїї в них(смайли, псевдографіка, тощо)
- глибше занурення у лінгвістику, використання разом позицій, міток частин мови та представлення щільними векторами
- створення асинхронного веб додатку
- побудова API і додавання підтримки JWS токена для контролю використання
- можливість аналізу великого масиву посилань на статті
- можливість аналізу введеного із клавіатури тексту

СПИСОК ЛІТЕРАТУРИ

1. Tendencies of development of the advertising and communication market of Ukraine – Butorina V. B., УДК 659.118(477)
2. Результати дослідження обсягу інтернет-реклами в Україні [електронний ресурс] – http://adcom.inau.ua/2021/03/05/internet-market-research-2020/
3. Об`єми рекламно-комунікаційного ринку України 2020 і прогноз на 2021 [електронний ресурс] - https://vrk.org.ua/news-events/2020/ad-volume-2020.html
4. Рейтинг найпопулярніших сайтів України за березень 2021 [електронний ресурс] – https://ms.detector.media/onlain-media/post/24463/2020-04-08-reytyng-naypopulyarnishykh-saytiv-za-berezen-mesendzhery-novyny-poshuk-likiv/
5. Анализируй это. Lenta.ru [електронний ресурс] – https://habr.com/ru/post/343838/
6. Microsoft Azure Text Analytics [електронний ресурс] – https://azure.microsoft.com/en-us/services/cognitive-services/text-analytics/
7. Amazon Comprehend Features – https://aws.amazon.com/comprehend/features/
8. Natural Language Processing in Action – Hobson Lane, Cole Howard, Hannes Max Hapke, Manning, 03/2019, 576 с.; ISBN: 9781617294631
9. Neural Network Methods for Natural Language Processing – Yoav Goldberg, Morgan&Claypool Publishers, 04/2017, 282 с.; ISBN: 9781627052986
10. Deep Learning – Ian Goodfellow, Yoshua Bengio, Aaron Courville, The MIT Press, 2017, 654 с.; ISBN: 9780262035613
11. Natural Language Processing with Python – Steven Bird, Ewan Klein, Edward Loper, O'Reilly Media, Inc., 06/2009; ISBN: 9780596516499

12. Sentiment Analysis: Modern Approaches and existing problems – Semina T. A., 2020
13. Requests: HTTP for humans [електронний ресурс] - https://docs.python-requests.org/en/master/
14. Selenium with Python [електронний ресурс] - https://selenium-python.readthedocs.io/
15. Beautiful Soup Documentation [електронний ресурс] - https://www.crummy.com/software/BeautifulSoup/bs4/doc/
16. Numpy [електронний ресурс] - https://numpy.org/
17. Natural Language Toolkit [електронний ресурс] - https://www.nltk.org/
18. Scikit-learn: Machine Learning in Python [електронний ресурс] - https://scikit-learn.org/stable/
19. Лемматизируй это быстрее [електронний ресурс] - https://habr.com/ru/post/503420/
20. A Python wrapper of the Yandex Mystem 3.1 morphological analyzer [електронний ресурс] - https://github.com/nlpub/pymystem3
21. Python Stop Words [електронний ресурс] - https://github.com/Alir3z4/python-stop-words
22. Морфологический анализатор pymorphy2 [електронний ресурс] - https://pymorphy2.readthedocs.io/en/stable/
23. Yandex catboost [електронний ресурс] - https://github.com/catboost/catboost
24. Gensim is for similarity retrieval with large corpora [електронний ресурс] - https://radimrehurek.com/gensim/
25. Проект Natasha [електронній ресурс] - https://habr.com/ru/post/516098/
26. Pydantic [електронний ресурс] - https://pydantic-docs.helpmanual.io/
27. Async Timeout [електронний ресурс] - https://github.com/aio-libs/async-timeout
28. Aiohttp [електронний ресурс] - https://docs.aiohttp.org/en/stable/

29. Word2Vec в картинках [електронний ресурс] - https://habr.com/ru/post/446530/
30. Flask [електронний ресурс] - https://flask.palletsprojects.com/en/2.0.x/
31. Flask Web Development, 2nd Edition – Miguel Grinberg, O`Reilly Media Inc., 03/2018; ISBN: 9781491991732
32. Learning Python, 5th Edition – Mark Lutz, O`Reilly Media Inc, 06/2013; ISBN: 9781449355739
33. Pandas, Python data scientist library [електронний ресурс] - https://pandas.pydata.org/
34. Web Scraping with Python 2nd Edition – Ryan Mitchell, O`Reilly Media Inc., 06/2015; ISBN: 9781491910276
35. Seaborn: statistical data visualization [електронний ресурс] - https://seaborn.pydata.org/
36. Рейтинг 100 лучших русских интернет СМИ [електронний ресурс] – https://wsjournal.ru/top-russian-internet-media-rating-russia-ukraine-belarus-smi-wsj/
37. Оценка классификатора(точность, полнота, F-мера) [електронний ресурс] – http://bazhenov.me/blog/2012/07/21/classification-performance-evaluation.html

ДОДАТОК

Програмний код

Providers

Модуль використовується для отримання даних зі статті

iz.py

Пакет для розбору даних із ресурсу "Известия"

```
import json
import os
import time

import requests
import selenium
from bs4 import BeautifulSoup
from selenium import webdriver
from selenium.common.exceptions import ElementClickInterceptedException
from selenium.webdriver.chrome.options import Options
from selenium.webdriver.common.by import By
from selenium.webdriver.support import expected_conditions as EC
from selenium.webdriver.support.ui import WebDriverWait

chrome_options = Options()
chrome_options.add_argument('--headless')
chrome_options.add_argument('--no-sandbox')
chrome_options.add_argument('--disable-dev-shm-usage')

X_PATH_BUTTON = "/html/body/div[4]/div[1]/div[3]/div[2]/div[1]/div/div[2]/div/div/div/div/ul/li/a"


def save_file(path, data):
    with open(path, 'w', encoding='utf-8') as outfile:
        json.dump(data, outfile, ensure_ascii=False)
```

```
def get_links(site_links,n_click):
    driver = webdriver.Chrome(executable_path ="/usr/lib/chromium-browser/chromedriver", chrome_options=chrome_options)
    driver.get(site_links)
    time.sleep(5)
    for _ in range(n_click):
        try:
            WebDriverWait(driver, 5).until(EC.element_to_be_clickable((By.XPATH, X_PATH_BUTTON))).click()
            time.sleep(5)
        except selenium.common.exceptions.ElementClickInterceptedException:
            pass
        except selenium.common.exceptions.WebDriverException:
            pass
    soup = BeautifulSoup(driver.page_source, 'lxml')
    links_lst = []
    for link in soup.find_all('a', class_='lenta_news__day__list__item show_views_and_comments'):
        links_lst += ['https://iz.ru'+link.get('href')]
    driver.close()
    return links_lst


def get_text(links_lst):
    res = []
    for links in links_lst:
        request = requests.get(links)
        soup2 = BeautifulSoup(request.text, 'lxml')
        dictionary = dict.fromkeys(['url', 'title', 'article', 'tags'])
        dictionary['url'] = links
        # title
        try:
           dictionary['title'] = soup2.find('h1', itemprop='headline').get_text()
        # text
        except:
           dictionary['title']=''
        tmp = []
        for elem in soup2.find_all('div',itemprop='articleBody'):
            tmp += [elem.get_text()]
        dictionary['article'] = tmp
        #tags
        try:
```

```
            dictionary['tags'] = soup2.find(class_="hash_tags").get_text()
        except:
            dictionary['tags'] = ''
        res+=[dictionary]
    return res


def parse(site=None, category=None, clicks=30, path_to_output=None):
    final_input_path = os.path.join(site, category)
    links = get_links(final_input_path, clicks)
    res_dic= get_text(links)
    data = save_file(path_to_output, res_dic)
    return data


def one_page_parse_iz(link):
    """Function for extracting text from an article
                Parameter:
                link (str): link to article from https://iz.ru/

                Returns:
                (str): 'text of the article'  """
    request = requests.get(link)
    soup = BeautifulSoup(request.text, 'lxml')
    try:
        text = str(soup.find('div', itemprop='articleBody').get_text())
    except:
        text = ""
        for elem in soup.find_all('div', itemprop='description'):
            text = text + elem.get_text()
    article = ' '.join(text.split())
    return article
```

lenta.py

Пакет для розбору даних із ресурсу Lenta

```
from copy import copy
```

```
import requests
from bs4 import BeautifulSoup


def get_raw_text(link):
    """return str"""
    request = requests.get(link)
    soup = BeautifulSoup(request.text, 'lxml')
    try:
        res = ' '.join(soup.find('div', class_='b-text clearfix js-topic__text').get_text().split())
    except:
        res = ' '.join(soup.find('div', itemprop="articleBody").get_text(" ").split())
    return res


def get_additional_materials(link):
    """ return list or None"""
    request = requests.get(link)
    soup = BeautifulSoup(request.text, 'lxml')
    additional_materials = []
    for elem in soup.find_all('aside', class_='b-inline-topics-box'):
        additional_materials += [elem.get_text(' ')]
    if len(additional_materials) == 0:
        return None
    else:
        return additional_materials


def get_additional_information(link):
    """ return list or None"""
    request = requests.get(link)
    soup = BeautifulSoup(request.text, 'lxml')
    additional_information = []
    all = []

    all += [" ".join([item.text for item in soup.select(".b-box b")])]
    all += [" ".join([item.text for item in soup.select(".b-box i")])]

    temp_text = ""
    for elem in soup.find_all('a', class_='covid'):
```

```
        temp_text += elem.get_text()
    all += [" ".join(temp_text.split())]

    temporary_text = ""
    for elem in soup.find_all('div', class_='social-snippet'):
        temporary_text += elem.get_text()
    all += [" ".join(temporary_text.split())]

    for element in all:
        if len(element) != 0:
            additional_information += [element]

    if len(additional_information) == 0:
        return None
    else:
        return additional_information


def remove_substring(substring, string):
    """function to remove extra material and information from a string
    return str"""
    substring = substring.replace(u'\xa0', u' ')
    string = string.replace(u'\xa0', u' ')
    left = substring[0:17]
    right = substring[-17:]
    begin_str = string[:string.index(left)]
    if string.index(left) + len(substring) > len(string):
        return begin_str
    else:
        end_str = string[len(right) + string.index(right):]
        return begin_str + end_str


def get_cleaned_text(text, additional_materials, additional_information):
    """ return str"""
    text_ = copy(text)
    if additional_materials == None and additional_information == None:
        return text

    elif additional_materials == None:
```

```
        for information in additional_information:
            text_ = remove_substring(information, text_)
        return text_

    elif additional_information == None:
        for materials in additional_materials:
            text_ = remove_substring(materials, text_)
        return text_
    else:
        for materials in additional_materials:
            text_ = remove_substring(materials, text_)
        for information in additional_information:
            text_ = remove_substring(information, text_)
        return text_


def one_page_parse_lenta(link):
    """ Parser, return str """
    text = get_raw_text(link)
    materials = get_additional_materials(link)
    info = get_additional_information(link)
    return get_cleaned_text(text, materials, info)


def check_url_and_get_article(link):
    site = "https://lenta.ru"
    url = requests.get(link)
    if url.status_code == 200:
        if site == link[:len(site)]:
            return one_page_parse_lenta(link)
        else:
            return "Please choose another parser"
    else:
        return "Please check the url"
```

meduza_parse.py

Пакет для розбору даних ресурсу Meduza

```
"""A simple Python module that wraps the meduza.io API."""

import gzip
import json
from urllib.parse import urlencode, urljoin
from urllib.request import urlopen
import requests
from bs4 import BeautifulSoup
from copy import copy


__all__ = [
    "LANGUAGES",
    "EN_SECTIONS",
    "RU_SECTIONS",
    "EN_TAGS",
    "RU_TAGS",
    "stocks",
    "get",
    "section",
    "tag",
    "reactions_for",
    "latest_push",
]


LANGUAGES = ["ru", "en"]

EN_SECTIONS = ["news"]

RU_SECTIONS = ["news", "articles", "shapito", "razbor", "games", "podcasts"]

EN_TAGS = ["news", "like it or not", "games"]

RU_TAGS = [
    "новости",
    "истории",
    "разбор",
    "шапито",
    "игры",
```

```
    "подкасты",
    "партнерский материал",
]


_BASEURL = "https://meduza.io"
_V3 = "api/v3"
_W4 = "api/w4"
_MISC = "api/misc"
_SEARCH_API = f"{_BASEURL}/{_W4}/search?"
_SOCIAL_API = f"{_BASEURL}/{_MISC}/social?"
_STOCK_API = f"{_BASEURL}/{_MISC}/stock/all"
_LATEST_PUSH = f"{_BASEURL}/{_V3}/push/chrome/latest"


# maps tags to sections (russian)
_rus_section_from = {
    "новости": "news",
    "истории": "articles",
    "шапито": "shapito",
    "разбор": "razbor",
    "игры": "games",
    "подкасты": "podcasts",
    "партнерский материал": "news",
}

# maps tags to sections (english)
_eng_section_from = {"news": "news", "games": "news", "like it or not": "news"}


# open URL and return JSON response (as dict)
def _GET(url):
    # open url
    response = urlopen(url)
    headers = dict(response.headers)
    data = response.read()
    # if the data is compressed using gzip, then decompress this.
    # (u is a gzip file if the first two bytes are '0x1f' and '0x8b')
    if headers.get("Content-Encoding") == "gzip":
        data = gzip.decompress(data)
```

```
    # remove all non-breaking spaces
    data = data.decode("utf-8").replace("\xa0", " ")
    return json.loads(data)


def stocks(key=None) -> dict:
    """Returns stocks."""
    response = _GET(_STOCK_API)
    return response if key is None else response[key]


def get(url: str) -> dict:
    """Gets the article for the url.

    `url` - the url of a page."""
    if not url.startswith(_BASEURL):
        url = urljoin(_BASEURL, url)
    if _W4 not in url:
        url = url.replace(_BASEURL, f"{_BASEURL}/{_W4}")
    return _GET(url)["root"]


def section(section: str, *, n=24, lang="ru", page=0):
    """Gets articles from the `section`.

    `section` - Section name (see `meduza.EN_SECTIONS` and q
    `meduza.RU_SECTIONS` constants);
    `n` - How many articles to return;
    `lang` - Russian or English version of meduza.io (see
    `meduza.LANGUAGES`)."""

    def _article_urls(url):
        documents = _GET(url)["documents"]
        for url in documents:
            if url == "nil":
                url = documents["nil"]["root"]["url"]
            yield url

    payload = {"chrono": section, "locale": lang, "page": page, "per_page": n}
    for url in _article_urls(_SEARCH_API + urlencode(payload)):
```

```
        yield get(url)


def _choose_section_if_tag(tag, *, lang):
    return _rus_section_from[tag] if lang == "ru" else _eng_section_from[tag]


def tag(tag, *, n=24, lang="ru"):
    """Gets articles with the `tag`.

    `tag` - An article tag. Same as in `article['tag']['name']` (see
    `meduza.EN_TAG`S and `meduza.RU_TAGS` constants);
    `n`  -- How many articles to return;
    `lang` -- Russian or English version of meduza.io (see
    `meduza.LANGUAGES`)."""
    section_name = _choose_section_if_tag(tag, lang=lang)
    page = 0
    m = n
    while m > 0:
        for article in section(section_name, n=n, lang=lang, page=page):
            if m > 0 and article["tag"]["name"] == tag:
                yield article
                m -= 1
        page += 1


def reactions_for(*urls) -> dict:
    """Gets number of reactions in social networks (and number of
    comments on meduza.io) for urls."""
    url = _SOCIAL_API + urlencode({"links": json.dumps([*urls])})
    return _GET(url)


def latest_push() -> dict:
    """Gets the latest push."""
    return _GET(_LATEST_PUSH)["notification"]


def get_raw_text_meduza(link):
    """Function for extracting text from an article
```

```
            Parameter:
            link (str): link to article from https://meduza.io/

            Returns:
            (str): 'text of the article'  """
    request = requests.get(link)
    soup = BeautifulSoup(request.text, 'lxml')
    text = ""
    try:
        text = soup.find('div', class_='GeneralMaterial-article').get_text(' ')
        for elem in soup.find_all('p', class_='SimpleBlock-p'):
            text = text + elem.get_text(' ')
    except:
        #4
        for elem in soup.find_all('p', class_='SimpleBlock-lead SimpleBlock-center'):
            text = text + elem.get_text(' ')

        for elem in soup.find_all('p', class_='SimpleBlock-lead SimpleBlock-center'):
            text = text + elem.get_text(' ')

        for elem in soup.find_all('p', class_='SimpleBlock-lead SimpleBlock-center SimpleBlock-game'):
            text = text + elem.get_text(' ')

        for elem in soup.find_all('blockquote', class_='SimpleBlock-blockquote SimpleBlock-center SimpleBlock-game'):
            text = text + elem.get_text(' ')

        #2
        for elem in soup.find_all('p', class_='SimpleBlock-lead SimpleBlock-slide'):
            text = text + elem.get_text(' ')

        for elem in soup.find_all('p', class_='SimpleBlock-p SimpleBlock-slide'):
           text = text + elem.get_text(' ')

    return text.replace('\n',' ')



def get_add_materials(link):
    request = requests.get(link)
    soup = BeautifulSoup(request.text, 'lxml')
```

```
sub_material = []

# 'напишите нам'
for elem in soup.find_all('span', class_='ToolbarButton-text'):
    if elem.get_text(' ') != '':
        sub_material += [elem.get_text(' ')]
    else:
        continue

# ссылки на другие статьи
for elem in soup.find_all('div', class_='RelatedBlock-root RelatedBlock-center RelatedBlock-rich'):
    sub_material += [elem.get_text(' ')]

# ссылки на другие статьи
for elem in soup.find_all('h3', class_="RelatedBlock-header"):
    sub_material += [elem.get_text(' ')]

# ссылки на другие статьи
for elem in soup.find_all('span', class_="RelatedBlock-first"):
    sub_material += [elem.get_text(' ')]

# ссылки на другие статьи
for elem in soup.find_all('span', target="_blank"):
    sub_material += [elem.get_text(' ')]

# ссылки на другие статьи
for elem in soup.find_all('a', "span"):
    sub_material += [elem.get_text(' ')]

# ссылки на другие статьи
for elem in soup.find_all('span', class_="RelatedBlock-rich"):
    sub_material += [elem.get_text(' ')]

# заметка из блокнота
for elem in soup.find_all('blockquote', class_='SimpleBlock-blockquote'):
    sub_material += [elem.get_text(' ')]

# подписи к фото 2
for elem in soup.find_all('div', class_='MediaCaption-credit'):
    sub_material += [elem.get_text(' ')]
```

```
    return sub_material


def remove_substring(substring, string):
    """function to remove extra material and information from a string
    return str"""
    substring = substring.replace(u'\xa0', u' ')
    string = string.replace(u'\xa0', u' ')
    if len(string) <= 12:
        left = substring[0:3]
        right = substring[-3:]
    else:
        left = substring[0:17]
        right = substring[-17:]
    begin_str = string[:string.index(left)]
    if string.index(left) + len(substring) > len(string):
        return begin_str
    else:
        end_str = string[len(right) + string.index(right):]
        return begin_str + end_str


def get_cleaned_text(text, additional_materials):
    """ return str"""
    text_ = copy(text)
    if additional_materials == '':
        return text
    else:
        for materials in additional_materials:
            try:
                text_ = remove_substring(materials, text_)
            except:
                text_ = text_ + ''
        return text_


def one_page_parse_meduza(link):
    text = get_raw_text_meduza(link)
    add_material = get_add_materials(link)
    return get_cleaned_text(text, add_material)
```

rbk_ru.py

Пакет для розбору даних ресурсу РБК

```
import requests
from bs4 import BeautifulSoup
from copy import copy

def get_raw_text_rbk(link):
    request = requests.get(link)
    soup = BeautifulSoup(request.text, 'lxml')
    text = soup.find('div', itemprop='articleBody').get_text(' ')
    article = ' '.join(text.split())
    return article


def get_add_materials_rbk(link):
    request = requests.get(link)
    soup = BeautifulSoup(request.text, 'lxml')
    sub_material = []
    for elem in soup.find_all('div', class_='article__main-image__title'):
        sub_material += [elem.get_text(' ').replace('        ', ' ')]

    for elem in soup.find_all('span', class_='article__main-image__author'):
        sub_material += [elem.get_text(' ').replace('        ', ' ')]

    for elem in soup.find_all('div', class_='article__picture__title'):
        sub_material += [elem.get_text(' ').replace('        ', ' ')]

    for elem in soup.find_all('span', class_='article__inline-item__title'):
        sub_material += [elem.get_text(' ').replace('        ', ' ')]

    for elem in soup.find_all('a', class_='article__inline-item__category'):
        sub_material += [elem.get_text(' ').replace('        ', ' ')]

    for elem in soup.find_all('a', class_='article__photoreport__category'):
        sub_material += [elem.get_text(' ').replace('        ', ' ')]
```

```
    for elem in soup.find_all('a', class_='article__photoreport__title'):
        sub_material += [elem.get_text(' ').replace('        ', ' ')] #.replace('\n','')

    for elem in soup.find_all('span', class_='article__photoreport__more'):
        sub_material += [elem.get_text(' ')]

    for elem in soup.find_all('div', class_='article__special_container'):
        sub_material += [elem.get_text(' ')]

    for elem in soup.find_all('strong'):
        sub_material += [elem.get_text(' ')]

    for elem in soup.find_all('a', target='_blank'):
        sub_material += [elem.get_text(' ')]

    for elem in soup.find_all('div', class_='article__inline-video__author js-insert-video-author'):
        sub_material += [elem.get_text(' ')]

    for elem in soup.find_all('div', class_='article__inline-video__text js-insert-video-desc'):
        sub_material += [elem.get_text(' ')]

    clean_sub_materials = []
    for material in sub_material:
        material = material.replace('\n', '')
        material = material.replace('        ', '')
        material = material.replace('     ', ' ')
        material = material.replace('  ', ' ')
        material = material.replace(u'\xa0', u' ')
        if material.startswith(' '):
            material = material[1:]
        if material.endswith(' '):
            material = material[:-1]
        clean_sub_materials += [material]
    return clean_sub_materials


def remove_substring(substring, string):
    """function to remove extra material and information from a string
    return str"""
    string = string.replace(u'\xa0', u' ')
```

```
    if len(string) <= 12:
        left = substring[0:3]
        right = substring[-3:]
    else:
        left = substring[0:17]
        right = substring[-17:]
    begin_str = string[:string.index(left)]
    if string.index(left) + len(substring) > len(string):
        return begin_str
    else:
        end_str = string[len(right) + string.index(right):]
        return begin_str + ' ' + end_str


def get_cleaned_text(text, additional_materials):
    """input: text is a string, additional_materials is a list.  return str"""
    text_ = copy(text)
    if additional_materials == '':
        return text
    else:
        for materials in additional_materials:
            try:
                text_ = remove_substring(materials, text_)
            except:
                text_ = text_ + ''
        return text_


def one_page_parse_rbk(link):
    article = get_raw_text_rbk(link)
    add_materials = get_add_materials_rbk(link)
    return get_cleaned_text(article, add_materials)
```

ria_parse.py

Пакет для розбору даних із ресурсу "РИА Новости"

```
import json
import os
```

```
import re
from datetime import datetime, timedelta

import requests
from bs4 import BeautifulSoup


def save_file(path, data):
    with open(path, 'w', encoding='utf-8') as outfile:
        json.dump(data, outfile, ensure_ascii=False)


def generate_time_period(date_start, date_end):
    """example of date_start:"2012-07-09"   """
    start = datetime.strptime(date_start, "%Y-%m-%d")
    end = datetime.strptime(date_end, "%Y-%m-%d")
    date_generated = [start + timedelta(days=x) for x in range(0, (end-start).days)]
    return date_generated


def get_links(dates_lst, category):
    links = []
    for date in dates_lst:
        date = date.strftime("%Y%m%d")
        site = 'https://ria.ru/services/tagsearch/?date_start=00&date=00&tags%5B%5D='
        tmp = site + category
        final_input_path = re.sub(r"00", date, tmp)
        request = requests.get(final_input_path)
        soup = BeautifulSoup(request.text, 'lxml')
        for link in soup.find_all('a', 'list-item__title color-font-hover-only'):
            links += ['https://radiosputnik.ria.ru' + link.get('href')]
    return links


def get_text(lst_links):
    res = []
    for link in lst_links:
        dictionary = dict.fromkeys(['url', 'title', 'article', 'tags'])
        dictionary['url'] = link
        request2 = requests.get(link)
```

```
        soup2 = BeautifulSoup(request2.text, 'lxml')

        for i in soup2.find_all('h1', class_='article__title'):
            dictionary['title'] = i.get_text()
        temp = []
        for elem in soup2.find_all('div', class_='article__text'):
            temp += [elem.get_text()]
        dictionary['article'] = temp

        try:
            for i in soup2.find_all('div', class_='article__quote-text m-small'):
                dictionary['article'] += [i.get_text()]
            for i in soup2.find_all('div', itemprop='keywords'):
                dictionary['tags'] = i.get_text()
        except:
            pass
        res += [dictionary]
    return res


def parse(date_start, date_end, category_, output_path):
    """date_start="2012-01-14", date_end="2020-07-09"  """
    time_period = generate_time_period(date_start, date_end)
    links = get_links(time_period, category_)
    result = get_text(links)
    data = save_file(output_path, result)
    return data


def one_page_parse_ria(link):
    """Function for extracting text from an article
         Parameter:
         link (str): link to article from https://ria.ru/

         Returns:
         (str): 'text of the article'  """
    request = requests.get(link)
    soup = BeautifulSoup(request.text, 'lxml')
    text = ""
    for elem in soup.find_all('div', class_='article__text'):
```

```
        text = text + elem.get_text()

    for element in soup.find_all('div', class_='article__quote-text m-small'):
        text = text + element.get_text()

    article = ' '.join(text.split())
    return article
```

desc_statistics

Модуль описової статистики

clening_data.ipynb

Ноутбук для очищення перед статистичним аналізом

```
#%%

import nltk
nltk.download("punkt")
nltk.download("stopwords")
nltk.download("wordnet")

#%%

import pandas as pd
from functools import reduce
from nltk.tokenize import word_tokenize
from nltk.corpus import stopwords
from nltk.stem.wordnet import WordNetLemmatizer

#%%

data_files = [
    "lenta.csv",
    "meduza.csv",
    "rbc.csv",
    "ria.csv",
    "iz.csv"
```

```
    ]

#%%

news_df = pd.concat(map(pd.read_csv, data_files))
news_df = news_df.drop_duplicates(subset=["link"], keep="first")


#%%

news_df = news_df[news_df["article"].notna()]

#%%

# tokenized corpus
tok_corp = news_df["article"].apply(word_tokenize)

#%%

# lowercase tokenized corpus
low_corp = tok_corp.apply(lambda doc: list(map(lambda token: token.lower(), doc)))

#%%

# add token to final list if is alpha
def reducer_alphas(prev_list, current_token):
    if current_token.isalpha():
        prev_list.append(current_token)
    return prev_list

# only alpha lowercase corpus
alpha_corp = low_corp.apply(lambda doc: reduce(reducer_alphas, doc, []))

#%%

stopwords = stopwords.words("russian")

# add token to final list if not stop word
def reducer_stop_words(prev_list, cur_token):
    if cur_token not in stopwords:
```

```
        prev_list.append(cur_token)
    return prev_list

# alpha lowercase tokenized corpus without stop words
no_stop_words_corp = alpha_corp.apply(lambda doc: reduce(reducer_stop_words, doc, []))

#%%

lemma = WordNetLemmatizer()

# stemmed lowercase tokenized corpus without stop words
lem_corp = no_stop_words_corp.apply(lambda doc: list(map(lemma.lemmatize, doc)))

#%%

full_cleaned_corp = lem_corp.apply(' '.join)
```

descriptive_statistics.ipynb

Ноутбук для проведення статистичного аналізу текстів

```
#%%

import pandas as pd
from functools import reduce
import seaborn as sns
from collections import Counter
import import_ipynb

#%%

from cleaning_data import tok_corp, low_corp, alpha_corp, no_stop_words_corp, full_cleaned_corp, stopwords

#%%

# words statistics
words_numbers = alpha_corp.apply(len)
```

```
#%%

# unique words statistics
unique_words_number = alpha_corp.apply(lambda doc: len(set(doc)))

#%%

# chars statistics
chars_number = alpha_corp.apply(lambda doc: len(' '.join(doc)))

#%%

def reducer_stop_words_defining(prev_list, token):
    if token in stopwords:
        prev_list.append(token)
    return prev_list
# stop words statistics
stop_words_in_corpus = alpha_corp.apply(lambda doc: reduce(reducer_stop_words_defining, doc, []))
stop_words_number = stop_words_in_corpus.apply(len)


#%%

# punctuations statistics
punctuations_number = low_corp.apply(len) - words_numbers



#%%

def reducer_is_title(prev_val, token):
    if token.istitle():
        prev_val += 1
    return prev_val

# title words statistics
title_words_number = tok_corp.apply(lambda doc: reduce(reducer_is_title, doc, 0))

#%%
```

```
# make data frame
news_features_df = pd.DataFrame({"clean_text": full_cleaned_corp,
                    "Кількість слів": words_numbers,
                    "Кількість слів із великої літери": title_words_number,
                    "Кількість унікальних слів": unique_words_number,
                    "Кількість типових слів": stop_words_number,
                    "Кількість букв": chars_number,
                    "Кількість знаків пунктуації": punctuations_number})

#%%

news_features_df.describe()

#%%

def visualize_top_words_in_dataset(dataset, plot_title):
    tokens_counter = Counter()
    dataset.apply(tokens_counter.update)
    most_common = dict(tokens_counter.most_common(10))
    words = list(most_common.keys())
    counts = list(most_common.values())
    sns.barplot(counts, words).set_title(plot_title)

#%%

visualize_top_words_in_dataset(alpha_corp, "Top 10 words")

#%%

visualize_top_words_in_dataset(stop_words_in_corpus, "ТОП 10 типових слів")

#%%

visualize_top_words_in_dataset(no_stop_words_corp, "ТОП 10 нетипових слів")
```

extracting_n-grams.ipynb

Ноутбук для аналізу цілих фраз

```
#%%

import import_ipynb
from sklearn.feature_extraction.text import TfidfVectorizer
import pandas as pd
import seaborn as sns

#%%

from cleaning_data import full_cleaned_corp

#%%

def get_n_grams(corpus, n_gram_size=2):
    learned_vocabulary = TfidfVectorizer(ngram_range=(n_gram_size, n_gram_size), smooth_idf=True,
use_idf=True).fit(corpus)
    tf_idf_vector = learned_vocabulary.transform(corpus)
    tokens_sum = tf_idf_vector.sum(axis=0)
    ranks = [(word, tokens_sum[0, index]) for word, index in learned_vocabulary.vocabulary_.items()]
    return pd.DataFrame.from_records(ranks, columns=["n-gram", "rank"]).sort_values("rank", ascending=False).head(10)

#%%

best_bi_grams = get_n_grams(full_cleaned_corp, 2)
sns.barplot(data=best_bi_grams, x="rank", y="n-gram").set_title("Найчастіщі біграми")

#%%

best_three_grams = get_n_grams(full_cleaned_corp, 3)
sns.barplot(data=best_three_grams, x="rank", y="n-gram").set_title("Найчастіші триграми"
```

nlp

Модуль для опрацювання текстів

preprocessing/cleanups.py

Пакет для очищення тексту

```
import re
import nltk
from nltk.corpus import stopwords
from pymystem3 import Mystem
from sklearn.feature_extraction.text import TfidfVectorizer
import numpy as np
import pymorphy2

stop_words = stopwords.words("russian")
mystem = Mystem()
tfidf_vectorizer = TfidfVectorizer()
morph = pymorphy2.MorphAnalyzer()


def clean_text(text):
    data = text.lower()
    data = re.sub(r'[^\w\s]', ' ', data, flags=re.UNICODE)
    data = re.sub(r'[\d]', ' ', data, flags=re.UNICODE)
    return data


def tokenize(text):
    return nltk.word_tokenize(clean_text(text))


def remove_stopwords(line):
    return [word for word in line if word not in stop_words]


def lemmatize(text):
    return [mystem.lemmatize(w)[0] for w in remove_stopwords(text)]


def cleanups_pipeline(text):
    data = clean_text(text)
    data = tokenize(data)
    data = lemmatize(data)
    return ' '.join(data)
```

```
def get_words(text, n = 20):
    data = lemmatize(tokenize(text))
    response = tfidf_vectorizer.fit_transform(data)
    feature_array = np.array(tfidf_vectorizer.get_feature_names())
    tfidf_sorting = np.argsort(response.toarray()).flatten()[::-1]
    top_n = feature_array[tfidf_sorting][:n]
    tags = [w for w in top_n.tolist() if 'NOUN' in morph.parse(w)[0].tag]
    return list(set(tags))
```

preprocessing/sentiment_preprocessing.py

Пакет для очищення тексту перед визначенням його тональності

```
import re
import nltk
from stop_words import get_stop_words
from pymorphy2 import MorphAnalyzer
from pymystem3 import Mystem
from catboost.text_processing import Tokenizer  # Catboost version==0.24!

morpher = MorphAnalyzer()  # lemmatizer
mystem = Mystem()
stemmer = nltk.stem.snowball.RussianStemmer()

stop_words_row = get_stop_words("ru")
stop_words = set([stemmer.stem(word) for word in stop_words_row])
stop_words = stop_words - set(['не'])

tokenizer = Tokenizer(  # tokenizer
    lowercasing=True,  # lowercase the text
    number_process_policy='Skip',  # skip digits
    separator_type='BySense',  # separate yourself)
    skip_empty=True,
    token_types=['Word']  # leave only words
)


def preprocess_text_stemmer(txt):
    txt = str(txt)
```

```
    txt = txt.lower()
    txt = re.sub('@[\w]*', ' ', txt)  # remove usernames
    txt = re.sub('[^а-я]+', ' ', txt)  # remove foreign words
    txt = [word for word in tokenizer.tokenize(txt) if word not in stop_words]
    txt = [stemmer.stem(word) for word in txt]
    txt = [word for word in txt if len(word) > 2]
    return ' '.join(txt)


def preprocess_text_pymorphy(txt):
    '''Much faster'''
    txt = txt.lower()
    txt = re.sub('@[\w]*', ' ', txt)  # remove usernames
    txt = re.sub('[^а-я]+', ' ', txt)  # remove foreign words
    txt = [morpher.parse(word)[0].normal_form for word in tokenizer.tokenize(txt)]
    txt = [word for word in txt if word not in stop_words_row and len(word) > 2]

    return ' '.join(txt)


def preprocess_text_pymystem(txt):
    '''Slow and boring. Never use on large dataset'''
    txt = re.sub('@[\w]*', ' ', txt)
    txt = " ".join(tokenizer.tokenize(txt))
    txt = " ".join([token for token in mystem.lemmatize(txt)
                    if token not in stop_words_row and len(token) > 2])

    return txt
```

sentiment/sentiment_combined_logreg.py

Пакет для визначення тональності тексту

```
from nlp.preprocessing.sentiment_preprocessing import preprocess_text_stemmer
from core.models_extracting import extract_sentiment_vectorizer, extract_sentiment_prediction_model

sentiment_dict = {
    0: 'negative',
    1: 'neutral',
```

```
    2: 'positive'
}

tf_idf_vectorizer = extract_sentiment_vectorizer()
model_logreg = extract_sentiment_prediction_model()


def get_sentiment(txt):
    result = model_logreg.predict(tf_idf_vectorizer.transform([preprocess_text_stemmer(txt)]))
    sentiment = sentiment_dict[result[0]]
    proba = model_logreg.predict_proba(tf_idf_vectorizer.transform([preprocess_text_stemmer(txt)]))
    return [sentiment, proba[0][0]]
```

npsi/npsi.py

Пакет для визначення індексу релевантності

```
from collections import Counter

from core.models_extracting import extract_npsi_model
from nlp.preprocessing.cleanups import get_words, cleanups_pipeline
from nlp.helpers.utils import CATEGORIES

big_model = extract_npsi_model()


def get_occurences(text):
    tags = get_words(text)
    cleaned_text = cleanups_pipeline(text)
    d = {}
    for e in tags:
        d.update({e: cleaned_text.count(e)})
    return d


def tma_calc(text):
    occurences = get_occurences(text)
    c = Counter(occurences)
    mc = c.most_common(3)
    qk = [x[1] for x in mc]
    tk = sum(occurences.values())
```

```
    tma = [x / tk for x in list(qk)]
    return tma


def npsi_calc(text):
    occurences = get_occurences(text)
    c = Counter(occurences)
    mc = c.most_common(3)
    tags = [x[0] for x in mc]
    to_flat = [list(x.values()) for x in CATEGORIES]
    seq = []
    for x, i in enumerate(to_flat):
        seq += [to_flat[x][0]]
    flat_list = [item for sublist in seq for item in sublist]
    tag_tm = []
    words = filter(lambda x: x in big_model.vocab, tags)
    for tag in words:
        tm = big_model.distances(tag, flat_list)
        tag_tm += [sum(tm)]
    tma = tma_calc(text)
    return sum(tma) * sum(tag_tm)
```

keywords/keywords.py

Пакет для визначення найважливіших слів у тексті

```
from nlp.sentiment.sentiment_combined_logreg import tf_idf_vectorizer as tfidf_transformer
import pickle

import numpy as np
import pymorphy2

from nlp.helpers.ner_extractor import *
from nlp.preprocessing.sentiment_preprocessing import preprocess_text_pymorphy


morph = pymorphy2.MorphAnalyzer()
GROUPING_SPACE_REGEX = re.compile('([^\w _-]|[+])', re.U)
```

```
def sort_coo(coo_matrix):
    """
    Sort a matrix
    :param coo_matrix: matrix to work with from the vectorizer
    :return: tuples with vector values
    """
    tuples = zip(coo_matrix.col, coo_matrix.data)
    return sorted(tuples, key=lambda x: (x[1], x[0]), reverse=True)


def extract_topn_from_vector(feature_names, sorted_items, topn=10):
    """
    Extracting topn words from the vector created from the original document
    :param feature_names: top features
    :param sorted_items: top sorted otems from the vector
    :param topn: numerical value
    :return: feature and score from the vector
    """
    sorted_items = sorted_items[:topn]
    score_vals = []
    feature_vals = []
    for idx, score in sorted_items:
        score_vals.append(round(score, 20))
        feature_vals.append(feature_names[idx])
        results = {}
    for idx in range(len(feature_vals)):
        results[feature_vals[idx]] = score_vals[idx]
    return results


def get_abbrevs(text):
    """
    Get an abbreviation, need to be reviewed
    :param text: string
    :return: abbreviations list
    """
    abbrevs = []
    article = [t for t in GROUPING_SPACE_REGEX.split(text)
               if t and not t.isspace()]
```

```
    for w in article:
        abbr = re.sub('[^А-Я]', '', w)
        if len(abbr) >= 2:
            abbrevs += [abbr]
    return abbrevs


def get_pos(w):
    """
    Getting a part of speech
    :param w: word to get part of speech
    :return: only nouns or verbs as most informative pos
    """
    p = morph.parse(w)[0]
    pos = p.tag.POS
    if pos == 'NOUN' or pos == 'PRTS':  # Adding participl
        return w


def get_keywords(doc):
    """
    Getting a keywords from the given string
    :param doc: text input, string
    :return: dict with a keyword and random (meanwhile - need to fix! distribution)
    """
    processed_doc = preprocess_text_pymorphy(doc)
    tf_idf_vector = tfidf_transformer.transform([processed_doc])  # Pre-trained tf-idf
    sorted_items = sort_coo(tf_idf_vector.tocoo())
    feature_names = tfidf_transformer.get_feature_names()
    keywords = extract_topn_from_vector(feature_names, sorted_items, 10)
    entities = get_entities_corrected(doc)
    vals = [k for k in keywords if len(k.split()) == 1]  # Choose unigramms
    normalized_vals = list(
        set(
            filter(None, [get_pos(a) for a in vals])) - set([x.lower() for x in entities]  # Remove NERs
                                                            )
    )
    total_keywords = list(set().union(normalized_vals, entities))
    filtered_keywords = list(filter(
        lambda x: [x for i in total_keywords if x in i.lower() and x != i.lower()] == [], total_keywords
```

```
    ))

    vocabulary = tfidf_transformer.vocabulary_  # A dictionary: word-->num
    keys_freqs = dict(sorted_items)  # A dictionary: num-->score

    freqs = []  # Scores
    kwords = []  # Words
    for key in filtered_keywords:
        if key.lower() in feature_names:
            freqs.append(keys_freqs[vocabulary[key.lower()]])  # Getting scores from sorted_items
        else:
            freqs.append(np.random.normal(3, 4, 1)[0])  # If the word is not in feature_names, generate score
        kwords.append(key)
    freqs = np.array(freqs)
    freqs_scaled = ((freqs - freqs.min()) / (freqs.max() - freqs.min())) * 10 + 1
    keywords_dict = dict(zip(kwords, freqs_scaled))
    return keywords_dict
```

helpers/corrector.py

Пакет для корекції закінчень в лемах

```
def correct_last_name(tag):
    if tag[-2:] == 'ым':
        return tag.replace('ым', 'ый')
    elif tag[-2:] == 'ой':
        return tag.replace('ой', 'ая')
    elif tag[-3:] == 'ому':
        return tag.replace('ому', 'ый')
    elif tag[-3:] == 'ого':
        return tag.replace('ого', 'ый')
    else:
        return tag
```

helpers/ner_extractor.py

Пакет для корегування форм слів

```
import re
from pymorphy2 import MorphAnalyzer
morpher = MorphAnalyzer()

from natasha import (
    Segmenter,
    MorphVocab,

    NewsEmbedding,
    NewsMorphTagger,
    NewsNERTagger,

    NamesExtractor,
    Doc
)

from nlp.helpers.corrector import *
from providers.lenta.lenta import *

segmenter = Segmenter()
morph_vocab = MorphVocab()

emb = NewsEmbedding()
morph_tagger = NewsMorphTagger(emb)
ner_tagger = NewsNERTagger(emb)

names_extractor = NamesExtractor(morph_vocab)


def correct_complex_words(entities):
    corrected = []
    for ner in entities:
        if re.findall(r'[a-zA-Z0-9_а-яА-Я]*-[а-яА-Я]*$', ner):
            fixed_ner = ner.split('-')
            corrected.append(''.join([fixed_ner[0].strip()] + ['-'] + [morpher.parse(fixed_ner[1])[0].normal_form]))
        else:
            corrected.append(ner)
    return corrected
```

```
def clean_entities(entities):
    entities = [re.sub('[«»]*', '', span) for span in entities]
    return entities


def get_entities(text):
    doc = Doc(text)
    doc.segment(segmenter)
    doc.tag_morph(morph_tagger)
#     doc.parse_syntax(syntax_parser)  # это не нужно для извлечения NER
    doc.tag_ner(ner_tagger)
    for span in doc.spans:
        span.normalize(morph_vocab)
    entities = set({_.normal for _ in doc.spans})
    entities = correct_complex_words((clean_entities(entities)))
    corrected = sorted(list(set([correct_last_name(entity) for entity in entities])))
    return corrected


def get_entities_corrected(text):
    entities = get_entities(text)

    return list(filter(lambda x: [x for i in entities if x in i and x != i] == [], entities))
```

core

Модуль для основної бізнес логіки додатка

build_response.py

Пакет для побудови основних відповідей для користувача

```
from copy import copy

import numpy as np
from PIL import Image
from wordcloud import WordCloud

from core.fetch_data import scrap_article
```

```
from nlp.npsi.npsi import npsi_calc
from nlp.keywords.keywords import get_keywords
from nlp.sentiment.sentiment_combined_logreg import get_sentiment
from resource_paths import mask_path, font_path


def build_result_data(primary_data):
    processed_data = copy(primary_data)
    processed_data["npsi_angle"] = get_npsi_angle(processed_data)
    processed_data["npsi_verdict"] = get_npsi_verdict(processed_data)
    processed_data["npsi"] = "%0.2f" % processed_data["npsi"]
    processed_data["sentiment_angle"] = get_sentiment_angle(processed_data)
    processed_data["word_cloud"] = build_word_cloud(processed_data["keywords"])
    return processed_data


def get_npsi_angle(data):
    max_scale = 1800
    npsi_value = data["npsi"]
    npsi_value_scale_context = npsi_value if npsi_value < max_scale else max_scale
    return npsi_value_scale_context * 0.15 + 250 * 0.15


def get_npsi_verdict(data):
    npsi_value = data["npsi"]
    if npsi_value <= 900:
        npsi_verdict = 'danger'
    elif npsi_value <= 1150:
        npsi_verdict = 'grey zone'
    else:
        npsi_verdict = 'safe'
    return npsi_verdict


def get_sentiment_angle(data):
    sentiment_result, sentiment_value = data["sentiment"][:2]
    sentiment_start_angle = 250 * 0.15 + sentiment_value * 139.5
    if sentiment_result == 'negative':
        return sentiment_start_angle
    elif sentiment_result == 'neutral':
```

```
        return sentiment_start_angle + 139.5
    else:
        return sentiment_start_angle + 250


def build_word_cloud(words):
    def color_func(word, font_size, position, orientation, random_state=None, **kwargs):
        return 'rgb(0, 0, 0)'

    mask = np.array(Image.open(mask_path))
    wc = WordCloud(font_path=font_path, max_words=100, mask=mask,
                   background_color='none', mode='RGBA', margin=4, prefer_horizontal=1, relative_scaling=0.5)
    wc.generate_from_frequencies(words)
    wc.recolor(color_func=color_func)
    svg = wc.to_svg()
    return svg


def get_npsi_keywords_sentiment(text):
    npsi = npsi_calc(text)
    keywords = get_keywords(text)
    sentiment = get_sentiment(text)
    return {
        "npsi": npsi,
        "keywords": keywords,
        "sentiment": sentiment
    }


def eval_article(article_url):
    """
    Method for making all article evaluating process. It scraps article by the url, eval it and return.
    If the url is None it throws UrlNotPassed exception, if it is about unknown provider - UnknownProvider exception.
    :param article_url: string with url of article inside
    :return: dict with npsi, keywords and sentiment keys
    """
    article_text = scrap_article(article_url)
    article_eval_data = get_npsi_keywords_sentiment(article_text)
    return article_eval_data
```

fetch_data.py

Пакет для розв`язання завдання розбору даних із різних ресурсів

```
import asyncio
import re

import aiohttp
import async_timeout

from app.exceptions import UrlNotPassed, UnknownProvider
from providers.izvestiya.iz import one_page_parse_iz
from providers.lenta.lenta import one_page_parse_lenta
from providers.meduza.meduza_parse import one_page_parse_meduza
from providers.rbk.rbk_ru import one_page_parse_rbk
from providers.ria.ria_parse import one_page_parse_ria

loop = asyncio.get_event_loop()


def scrap_article(article_url):
    article_text = loop.run_until_complete(get_input(article_url))
    return article_text


def check_url(link):
    try:
        url_verbose = re.sub('http(s?):\/\/', '', link).split('/')[0]
    except:
        raise UrlNotPassed()
    return url_verbose


async def fetch(url):
    async with aiohttp.ClientSession() as session, async_timeout.timeout(10):
        async with session.get(url) as response:
            return await response.text()


async def get_input(url):
```

```
    url_verbose = check_url(url)
    if url_verbose == 'iz.ru':
        return one_page_parse_iz(link=url)
    elif url_verbose == 'ria.ru':
        return one_page_parse_ria(link=url)
    elif url_verbose == 'meduza.io':
        return one_page_parse_meduza(link=url)
    elif url_verbose == 'lenta.ru':
        return one_page_parse_lenta(link=url)
    elif 'rbc.ru' in url_verbose:
        return one_page_parse_rbk(link=url)
    else:
        raise UnknownProvider()
```

model_etracting.py

Пакет для розпакування готових моделей

```
from pickle import load

from gensim.models.keyedvectors import KeyedVectors

from resource_paths import sentiment_logreg_model_path, \
    sentiment_vectorizer_path, lenta_lemmatize_model_path


def extract_sentiment_vectorizer():
    with open(sentiment_vectorizer_path, "rb") as vect_file:
        tfidf_vectorizer = load(vect_file)
    return tfidf_vectorizer


def extract_sentiment_prediction_model():
    with open(sentiment_logreg_model_path, "rb") as model_file:
        model_logreg = load(model_file)
    return model_logreg


def extract_npsi_model():
```

```
    big_model = KeyedVectors.load_word2vec_format(lenta_lemmatize_model_path, binary=False)
    return big_model
```

app

Веб модуль додатку

__init__.py

Пакет для створення веб додатку

```
from flask import Flask
from flask_restful import Api

flask_app = Flask(__name__)
flask_api = Api(flask_app)

from app.api import routes
from app import api

"""This file loads always first that is why we have to init here all variables for whole package.
We make init of flask app and api service. We make afterwards import to avoid import conflicts because
in other module`s files like routes.py and api.py we import back app and api variables"""
```

routes.py

Пакет для реєстрації основного шляху для головоної форми

```
from flask import render_template, request

from app import flask_api
from app import flask_app
from app.api.api import NewsApi
from core.build_response import build_result_data
from core.build_response import eval_article
from app.exceptions import WrongUrl
from app.swagger import SWAGGER_UI_BLUEPRINT, SWAGGER_URL
```

```
flask_app.register_blueprint(SWAGGER_UI_BLUEPRINT, url_prefix=SWAGGER_URL)


@flask_app.route('/', methods=['GET', 'POST'])
def index():
    if request.method == "POST":
        article_url = request.form.get("url")
        try:
            primary_eval_data = eval_article(article_url)
        except WrongUrl:
            return render_template('form.html', open=True)
        processed_eval_data = build_result_data(primary_eval_data)
        return render_template('response.html', data=processed_eval_data)
    return render_template('form.html', open=False)
```

api.py

Пакет для створення API

```
from copy import copy

from flask import jsonify
from flask_restful import Resource, request

from core.build_response import eval_article
from core.build_response import get_npsi_verdict
from app.exceptions import WrongUrl


class NewsApi(Resource):
    """
    Class that represents API resource
    """

    def get(self):
        """
        Method get GET api request and process it
        :return: json object
        """
```

```
        article_url = request.args.get("article_url")
        return self.build_response(article_url)

    def post(self):
        """
        Method get POST api request and process it
        :return: json object
        """
        article_url = request.json.get("article_url")
        return self.build_response(article_url)

    def build_response(self, article_url):
        """
        Method get article url from request by key and process article on the url by eval_article function.
        If url is None or article is from unknown provider method return error message to user.
        :return: json object
        """
        try:
            primary_data = eval_article(article_url)
            processed_data = self.build_user_friendly_data(primary_data)
            return jsonify(processed_data)
        except WrongUrl as err:
            return jsonify(err.cause)

    def build_user_friendly_data(self, primary_data):
        """
        Method takes object with "npsi", "keywords" and "sentiment" keys and process the data for better user
understanding.
        The result object has the same keys but the value of the npsi is the dict with "value" and "verdict" keys.
        :param primary_data: dict with "npsi", "keywords" and "sentiment" keys
        :return: dict with "npsi", "keywords" and "sentiment" keys
        """
        processed_data = copy(primary_data)
        npsi_object = dict()
        npsi_verdict = get_npsi_verdict(processed_data)
        npsi_object["safe"] = self.is_npsi_safe(npsi_verdict)
        npsi_object["value"] = processed_data["npsi"]
        processed_data["npsi"] = npsi_object
        return processed_data
```

```
def is_npsi_safe(self, npsi_verdict):
    npsi_versicts = {"safe": "true",
                     "danger": "false"}
    try:
        return npsi_versicts[npsi_verdict]
    except:
        return "skip"
```

Перші 400 використані посилання із Lenta

https://lenta.ru/news/2020/09/23/alive/
https://lenta.ru/news/2020/09/23/oleg/
https://lenta.ru/news/2020/09/23/razgon/
https://lenta.ru/news/2020/09/23/tr/
https://lenta.ru/news/2020/09/23/migrants/
https://lenta.ru/news/2020/09/23/tanto/
https://lenta.ru/news/2020/09/23/paschi/
https://lenta.ru/news/2020/09/23/rearrangement/
https://lenta.ru/news/2020/09/23/topsecret/
https://lenta.ru/news/2020/09/23/plan/
https://lenta.ru/news/2020/09/23/climatecov/
https://lenta.ru/news/2020/09/23/prichina/
https://lenta.ru/news/2020/09/23/insurance/
https://lenta.ru/news/2020/09/23/sh/
https://lenta.ru/news/2020/09/23/pobeg/
https://lenta.ru/news/2020/09/23/android/
https://lenta.ru/news/2020/09/23/nasilnik/
https://lenta.ru/news/2020/09/23/reason/
https://lenta.ru/news/2020/09/23/ustal/
https://lenta.ru/news/2020/09/23/zakon/
https://lenta.ru/news/2020/09/23/tonnel/
https://lenta.ru/news/2020/09/23/bel_housing/
https://lenta.ru/news/2020/09/23/kaia/
https://lenta.ru/news/2020/09/23/naval/
https://lenta.ru/news/2020/09/23/mos_vac/
https://lenta.ru/news/2020/09/23/full/
https://lenta.ru/news/2020/09/23/greece/
https://lenta.ru/news/2020/09/23/new_built/
https://lenta.ru/news/2020/09/23/stydno/
https://lenta.ru/news/2020/09/23/shtepa/
https://lenta.ru/news/2020/09/23/cosmetics/
https://lenta.ru/news/2020/09/23/inn/
https://lenta.ru/news/2020/09/23/father/
https://lenta.ru/news/2020/09/23/oldgremlin/
https://lenta.ru/news/2020/09/23/neg_tarif/
https://lenta.ru/news/2020/09/23/withdrawal/
https://lenta.ru/news/2020/09/23/protest/
https://lenta.ru/news/2020/09/23/peskov/
https://lenta.ru/news/2020/09/23/indania/
https://lenta.ru/news/2020/09/23/razgovor/
https://lenta.ru/news/2020/09/23/akchiz/
https://lenta.ru/news/2020/09/23/how_it_works/
https://lenta.ru/news/2020/09/23/ntechlab/
https://lenta.ru/news/2020/09/23/st/
https://lenta.ru/news/2020/09/23/borscheva_comedy/
https://lenta.ru/news/2020/09/23/karpovich/
https://lenta.ru/news/2020/09/23/iri/
https://lenta.ru/news/2020/09/23/matkap/
https://lenta.ru/news/2020/09/23/egg/
https://lenta.ru/news/2020/09/23/win/
https://lenta.ru/news/2020/09/23/bad_sale/
https://lenta.ru/news/2020/09/23/obed/
https://lenta.ru/news/2020/09/23/bez/
https://lenta.ru/news/2020/09/23/pozner_zvanie/
https://lenta.ru/news/2020/09/23/rodstvennik/
https://lenta.ru/news/2020/09/23/maski_show/
https://lenta.ru/news/2020/09/23/mrot/
https://lenta.ru/news/2020/09/23/elephant/
https://lenta.ru/news/2020/09/23/spravka/
https://lenta.ru/news/2020/09/23/bestemyanova/
https://lenta.ru/news/2020/09/23/tainaya_lukashenko/
https://lenta.ru/news/2020/09/23/gerluka/
https://lenta.ru/news/2020/09/23/effect/
https://lenta.ru/news/2020/09/23/meat/
https://lenta.ru/news/2020/09/23/up/
https://lenta.ru/news/2020/09/23/msp/
https://lenta.ru/news/2020/09/23/heart_problems/
https://lenta.ru/news/2020/09/23/chinovniki/
https://lenta.ru/news/2020/09/23/dryan/
https://lenta.ru/news/2020/09/23/delo/
https://lenta.ru/news/2020/09/23/wedding/
https://lenta.ru/news/2020/09/23/spies/
https://lenta.ru/news/2020/09/23/domoi/
https://lenta.ru/news/2020/09/23/comment/
https://lenta.ru/news/2020/09/23/ndpi/
https://lenta.ru/news/2020/09/23/abyzov/
https://lenta.ru/news/2020/09/23/malysheva_moshenniki/
https://lenta.ru/news/2020/09/23/dollar/

https://lenta.ru/news/2020/09/23/station/
https://lenta.ru/news/2020/09/23/nagrada/
https://lenta.ru/news/2020/09/23/baikal/
https://lenta.ru/news/2020/09/23/ozenil/
https://lenta.ru/news/2020/09/23/superapps/
https://lenta.ru/news/2020/09/23/vereschuk/
https://lenta.ru/news/2020/09/23/stuart/
https://lenta.ru/news/2020/09/23/pamyatnikihab/
https://lenta.ru/news/2020/09/23/lecar/
https://lenta.ru/news/2020/09/23/roman/
https://lenta.ru/news/2020/09/23/long_term/
https://lenta.ru/news/2020/09/23/applerus/
https://lenta.ru/news/2020/09/23/photosu/
https://lenta.ru/news/2020/09/23/koffi/
https://lenta.ru/news/2020/09/23/legs/
https://lenta.ru/news/2020/09/23/time/
https://lenta.ru/news/2020/09/23/peyte_deti_moloko/
https://lenta.ru/news/2020/09/23/cia/
https://lenta.ru/news/2020/09/23/dagestann/
https://lenta.ru/news/2020/09/23/gaz/
https://lenta.ru/news/2020/09/23/dimon/
https://lenta.ru/news/2020/09/23/desyatki/
https://lenta.ru/news/2020/09/23/xandra/
https://lenta.ru/news/2020/09/23/ximki/
https://lenta.ru/news/2020/09/23/mur_mur_mur/
https://lenta.ru/news/2020/09/23/harry/
https://lenta.ru/news/2020/09/23/artem/
https://lenta.ru/news/2020/09/23/deputat/
https://lenta.ru/news/2020/09/23/notpresident/
https://lenta.ru/news/2020/09/23/million/
https://lenta.ru/news/2020/09/23/taxi/
https://lenta.ru/news/2020/09/23/fighter/
https://lenta.ru/news/2020/09/23/analyzy/
https://lenta.ru/news/2020/09/23/barbie/
https://lenta.ru/news/2020/09/23/polsha/
https://lenta.ru/news/2020/09/23/legsandarms/
https://lenta.ru/news/2020/09/23/antibiotik/
https://lenta.ru/news/2020/09/23/un4/
https://lenta.ru/news/2020/09/23/musk/
https://lenta.ru/news/2020/09/23/nevinovnaya_ya/
https://lenta.ru/news/2020/09/23/roza_uspekh/
https://lenta.ru/news/2020/09/23/another_uber/
https://lenta.ru/news/2020/09/23/idish/
https://lenta.ru/news/2020/09/23/swe/
https://lenta.ru/news/2020/09/23/pension/
https://lenta.ru/news/2020/09/23/ba/
https://lenta.ru/news/2020/09/23/sektant/
https://lenta.ru/news/2020/09/23/snowden/
https://lenta.ru/news/2020/09/23/trrr/
https://lenta.ru/news/2020/09/23/uae/
https://lenta.ru/news/2020/09/23/baiden/
https://lenta.ru/news/2020/09/23/antiross/
https://lenta.ru/news/2020/09/23/bankrot/
https://lenta.ru/news/2020/09/23/kazn/
https://lenta.ru/news/2020/09/23/liza/
https://lenta.ru/news/2020/09/23/devito/
https://lenta.ru/news/2020/09/23/huawei/
https://lenta.ru/news/2020/09/23/camera/
https://lenta.ru/news/2020/09/23/marihuana/
https://lenta.ru/news/2020/09/23/healthy_fish/
https://lenta.ru/news/2020/09/23/vrez/
https://lenta.ru/news/2020/09/23/vina/
https://lenta.ru/news/2020/09/23/no_rush/
https://lenta.ru/news/2020/09/23/pandemiacov/
https://lenta.ru/news/2020/09/23/sberzvooq/
https://lenta.ru/news/2020/09/23/blood/
https://lenta.ru/news/2020/09/23/mono/
https://lenta.ru/news/2020/09/23/time_putin/
https://lenta.ru/news/2020/09/23/macdonalds_invest/
https://lenta.ru/news/2020/09/23/rasstrel/
https://lenta.ru/news/2020/09/23/tsvet_nacyi/
https://lenta.ru/news/2020/09/23/sosed_timon/
https://lenta.ru/news/2020/09/23/obstacle/
https://lenta.ru/news/2020/09/23/tomato/
https://lenta.ru/news/2020/09/23/gnezdo/
https://lenta.ru/news/2020/09/23/pollution/
https://lenta.ru/news/2020/09/23/razon/
https://lenta.ru/news/2020/09/23/so_slezami_na_glazah/
https://lenta.ru/news/2020/09/23/vmf/
https://lenta.ru/news/2020/09/23/oplata/
https://lenta.ru/news/2020/09/23/raschlen/
https://lenta.ru/news/2020/09/23/china/
https://lenta.ru/news/2020/09/23/khamzat/
https://lenta.ru/news/2020/09/23/farce/

https://lenta.ru/news/2020/09/23/genderfluid/
https://lenta.ru/news/2020/09/23/appleb14/
https://lenta.ru/news/2020/09/23/samba/
https://lenta.ru/news/2020/09/23/giprosvyaz/
https://lenta.ru/news/2020/09/23/eiffeltower/
https://lenta.ru/news/2020/09/23/tikhanovskaya/
https://lenta.ru/news/2020/09/23/ocheredi_no/
https://lenta.ru/news/2020/09/23/shkola/
https://lenta.ru/news/2020/09/23/esli_hochesh_ostatsya/
https://lenta.ru/news/2020/09/23/zhvania/
https://lenta.ru/news/2020/09/23/insure/
https://lenta.ru/news/2020/09/23/petroleum/
https://lenta.ru/news/2020/09/23/zub/
https://lenta.ru/news/2020/09/23/msp2/
https://lenta.ru/news/2020/09/23/zdravoohranenie/
https://lenta.ru/news/2020/09/23/prim/
https://lenta.ru/news/2020/09/23/planroad/
https://lenta.ru/news/2020/09/23/wildberries/
https://lenta.ru/news/2020/09/23/no_corona_only_pentagona/
https://lenta.ru/news/2020/09/23/police/
https://lenta.ru/news/2020/09/23/nologi/
https://lenta.ru/news/2020/09/23/tu160/
https://lenta.ru/news/2020/09/23/elle/
https://lenta.ru/news/2020/09/23/no_help/
https://lenta.ru/news/2020/09/23/krim/
https://lenta.ru/news/2020/09/23/fns/
https://lenta.ru/news/2020/09/23/kompleks/
https://lenta.ru/news/2020/09/23/vyslala/
https://lenta.ru/news/2020/09/23/kugran/
https://lenta.ru/news/2020/09/23/perepel_hity/
https://lenta.ru/news/2020/09/23/writer_nen/
https://lenta.ru/news/2020/09/23/eskimo/
https://lenta.ru/news/2020/09/23/khamstvo/
https://lenta.ru/news/2020/09/23/ts4/
https://lenta.ru/news/2020/09/23/imposed/
https://lenta.ru/news/2020/09/23/novichok/
https://lenta.ru/news/2020/09/23/solovyev_parody/
https://lenta.ru/news/2020/09/23/libya/
https://lenta.ru/news/2020/09/23/sizo/
https://lenta.ru/news/2020/09/23/ruble/
https://lenta.ru/news/2020/09/23/noseless/
https://lenta.ru/news/2020/09/23/spad/
https://lenta.ru/news/2020/09/23/dominatrix/
https://lenta.ru/news/2020/09/23/creepy_catfish/
https://lenta.ru/news/2020/09/23/vostochny/
https://lenta.ru/news/2020/09/23/son/
https://lenta.ru/news/2020/09/23/russia_blacklist/
https://lenta.ru/news/2020/09/23/jellyfish/
https://lenta.ru/news/2020/09/23/sabo/
https://lenta.ru/news/2020/09/23/deputat_umer/
https://lenta.ru/news/2020/09/23/yetanotherevent/
https://lenta.ru/news/2020/09/23/gas/
https://lenta.ru/news/2020/09/23/affair/
https://lenta.ru/news/2020/09/23/perehvat/
https://lenta.ru/news/2020/09/23/fear/
https://lenta.ru/news/2020/09/23/arest/
https://lenta.ru/news/2020/09/23/obeshaniye/
https://lenta.ru/news/2020/09/23/legitimniy/
https://lenta.ru/news/2020/09/23/ya_ne/
https://lenta.ru/news/2020/09/23/nlm/
https://lenta.ru/news/2020/09/23/film/
https://lenta.ru/news/2020/09/23/horoshie_palchiki/
https://lenta.ru/news/2020/09/23/usa_plane/
https://lenta.ru/news/2020/09/23/ninada/
https://lenta.ru/news/2020/09/23/prison_guard/
https://lenta.ru/news/2020/09/23/111/
https://lenta.ru/news/2020/09/23/stop_tovar/
https://lenta.ru/news/2020/09/23/pomoshnik/
https://lenta.ru/news/2020/09/23/risk/
https://lenta.ru/news/2020/09/23/hairloss/
https://lenta.ru/news/2020/09/23/svyaz/
https://lenta.ru/news/2020/09/23/zavet/
https://lenta.ru/news/2020/09/23/gazzaev/
https://lenta.ru/news/2020/09/23/weightloss/
https://lenta.ru/news/2020/09/23/zakrytiye/
https://lenta.ru/news/2020/09/23/cheap/
https://lenta.ru/news/2020/09/23/propal/
https://lenta.ru/news/2020/09/23/soveti/
https://lenta.ru/news/2020/09/23/pitanie/
https://lenta.ru/news/2020/09/23/whatsapp/
https://lenta.ru/news/2020/09/23/upskirt/
https://lenta.ru/news/2020/09/23/spiker/
https://lenta.ru/news/2020/09/23/inauguraciya/
https://lenta.ru/news/2020/09/23/cifra/

https://lenta.ru/news/2020/09/23/neptune/
https://lenta.ru/news/2020/09/23/malysheva_nos/
https://lenta.ru/news/2020/09/23/lottery/
https://lenta.ru/news/2020/09/23/taiyna/
https://lenta.ru/news/2020/09/23/versace/
https://lenta.ru/news/2020/09/23/sposob/
https://lenta.ru/news/2020/09/23/oymyakon/
https://lenta.ru/news/2020/09/23/otsenka/
https://lenta.ru/news/2020/09/23/minobr/
https://lenta.ru/news/2020/09/23/popravki/
https://lenta.ru/news/2020/09/23/icebox/
https://lenta.ru/news/2020/09/23/su30/
https://lenta.ru/news/2020/09/23/checkra1n/
https://lenta.ru/news/2020/09/23/avast/
https://lenta.ru/news/2020/09/23/drugs/
https://lenta.ru/news/2020/09/23/nacia/
https://lenta.ru/news/2020/09/23/high/
https://lenta.ru/news/2020/09/23/zaderzhany/
https://lenta.ru/news/2020/09/23/potolok/
https://lenta.ru/news/2020/09/23/arbooz/
https://lenta.ru/news/2020/09/23/window/
https://lenta.ru/news/2020/09/23/nazval_chislo/
https://lenta.ru/news/2020/09/23/vacuna/
https://lenta.ru/news/2020/09/23/holden/
https://lenta.ru/news/2020/09/23/growth/
https://lenta.ru/news/2020/09/23/neluka/
https://lenta.ru/news/2020/09/23/ndfl/
https://lenta.ru/news/2020/09/23/tesak/
https://lenta.ru/news/2020/09/23/voin/
https://lenta.ru/news/2020/09/23/mesto/
https://lenta.ru/news/2020/09/23/dopolnit/
https://lenta.ru/news/2020/09/23/oil_russia/
https://lenta.ru/news/2020/09/23/javelin/
https://lenta.ru/news/2020/09/23/kamenskikh/
https://lenta.ru/news/2020/09/23/queso/
https://lenta.ru/news/2020/09/23/otstoyali/
https://lenta.ru/news/2020/09/23/tinkoff/
https://lenta.ru/news/2020/09/23/lubovnichek/
https://lenta.ru/news/2020/09/23/devchulya/
https://lenta.ru/news/2020/09/23/kontrol/
https://lenta.ru/news/2020/09/23/explanation/
https://lenta.ru/news/2020/09/23/centres/
https://lenta.ru/news/2020/09/23/remont/
https://lenta.ru/news/2020/09/23/pashaev_deputat/
https://lenta.ru/news/2020/09/23/noah_purvis/
https://lenta.ru/news/2020/09/23/bezhentsy/
https://lenta.ru/news/2020/09/23/advvv/
https://lenta.ru/news/2020/09/23/zabor/
https://lenta.ru/news/2020/09/23/poimaly/
https://lenta.ru/news/2020/09/23/darroch/
https://lenta.ru/news/2020/09/23/rasteryalas/
https://lenta.ru/news/2020/09/23/rus_vaccine/
https://lenta.ru/news/2020/09/23/kodeks/
https://lenta.ru/news/2020/09/23/not_good/
https://lenta.ru/news/2020/09/23/rudkovskaya/
https://lenta.ru/news/2020/09/23/avtozak/
https://lenta.ru/news/2020/09/23/gotovnost/
https://lenta.ru/news/2020/09/23/razvedka/
https://lenta.ru/news/2020/09/23/advokat/
https://lenta.ru/news/2020/09/23/putin/
https://lenta.ru/news/2020/09/23/trillionp/
https://lenta.ru/news/2020/09/23/weight/
https://lenta.ru/news/2020/09/23/risk_kapusta/
https://lenta.ru/news/2020/09/23/leszhenya/
https://lenta.ru/news/2020/09/23/minsk/
https://lenta.ru/news/2020/09/23/heal/
https://lenta.ru/news/2020/09/23/vodomet/
https://lenta.ru/news/2020/09/23/epstein_didnt_youknowwhar/
https://lenta.ru/news/2020/09/23/dtp/
https://lenta.ru/news/2020/09/23/transpot/
https://lenta.ru/news/2020/09/23/sooslov/
https://lenta.ru/news/2020/09/23/raneny/
https://lenta.ru/news/2020/09/23/renat/
https://lenta.ru/news/2020/09/23/su57/
https://lenta.ru/news/2020/09/23/jump/
https://lenta.ru/news/2020/09/23/startups/
https://lenta.ru/news/2020/09/23/cycycy/
https://lenta.ru/news/2020/09/23/slow/
https://lenta.ru/news/2020/09/23/reper/
https://lenta.ru/news/2020/09/23/vstrecha/
https://lenta.ru/news/2020/09/23/meri/
https://lenta.ru/news/2020/09/23/neeeee/
https://lenta.ru/news/2020/09/23/bogdan/
https://lenta.ru/news/2020/09/23/isolation/

https://lenta.ru/news/2020/09/22/narko/
https://lenta.ru/news/2020/09/22/fsb/
https://lenta.ru/news/2020/09/22/alkosekta/
https://lenta.ru/news/2020/09/21/guliev/
https://lenta.ru/news/2020/09/21/mdm/
https://lenta.ru/news/2020/09/21/karkanye/
https://lenta.ru/news/2020/09/21/worst/
https://lenta.ru/news/2020/09/22/ne_mogu_smotret/
https://lenta.ru/news/2020/09/22/moderator/
https://lenta.ru/news/2020/09/22/predatelstvo/
https://lenta.ru/news/2020/09/22/dacha/
https://lenta.ru/news/2020/09/22/pivas/
https://lenta.ru/news/2020/09/22/serd/
https://lenta.ru/news/2020/09/22/samaya/
https://lenta.ru/news/2018/06/03/medinsky/
https://lenta.ru/news/2018/05/31/palibin/
https://lenta.ru/news/2018/04/19/kp/
https://lenta.ru/news/2020/09/22/tochki_rosta/
https://lenta.ru/news/2020/09/22/svrdlvskobl/
https://lenta.ru/news/2020/09/22/chechnya/
https://lenta.ru/news/2020/09/22/granti/
https://lenta.ru/news/2020/09/22/sputnik/
https://lenta.ru/news/2020/09/22/minzdrav/
https://lenta.ru/news/2020/09/22/hospitals/
https://lenta.ru/news/2020/09/22/vaccine/
https://lenta.ru/news/2020/09/22/efr/
https://lenta.ru/news/2020/09/22/second_wave/
https://lenta.ru/news/2020/09/22/pensiya/
https://lenta.ru/news/2020/09/22/ambrosio/
https://lenta.ru/news/2020/09/22/zona/
https://lenta.ru/news/2020/09/22/iran_sanctions/
https://lenta.ru/news/2020/09/22/gd_oon/
https://lenta.ru/news/2020/09/22/harry_meg/
https://lenta.ru/news/2020/09/22/infection/
https://lenta.ru/news/2020/09/22/conserve/
https://lenta.ru/news/2020/09/22/bogaty/
https://lenta.ru/news/2020/09/22/summatrat/
https://lenta.ru/news/2020/09/22/lider/
https://lenta.ru/news/2020/09/22/cat/
https://lenta.ru/news/2020/09/22/gosdolg/
https://lenta.ru/news/2020/09/22/svoboda/
https://lenta.ru/news/2020/09/22/duh/
https://lenta.ru/news/2020/09/22/banki/
https://lenta.ru/news/2020/09/22/mos_vtorichka/
https://lenta.ru/news/2020/09/22/harlamov_isolation/
https://lenta.ru/news/2020/09/22/sprotein/
https://lenta.ru/news/2020/09/22/tt/
https://lenta.ru/news/2020/09/22/nitsoi/
https://lenta.ru/news/2020/09/22/bride/
https://lenta.ru/news/2020/09/22/uuutechka/
https://lenta.ru/news/2020/09/22/chudo/
https://lenta.ru/news/2020/09/22/paket/
https://lenta.ru/news/2020/09/22/dead_referee/
https://lenta.ru/news/2020/09/22/human_are_not_needed/
https://lenta.ru/news/2020/09/22/fil/
https://lenta.ru/news/2020/09/22/reaction_rpc/
https://lenta.ru/news/2020/09/22/loss/
https://lenta.ru/news/2020/09/22/hospital_spb/
https://lenta.ru/news/2020/09/22/tsunamiturki/
https://lenta.ru/news/2020/09/22/drogas/
https://lenta.ru/news/2020/09/22/ojidaemo/
https://lenta.ru/news/2020/09/22/dry/
https://lenta.ru/news/2020/09/22/russiabelorussia/
https://lenta.ru/news/2020/09/22/kovalev/
https://lenta.ru/news/2020/09/22/rt_trump/
https://lenta.ru/news/2020/09/22/iphone_mini/
https://lenta.ru/news/2020/09/22/treniruysya/
https://lenta.ru/news/2020/09/22/emma/
https://lenta.ru/news/2020/09/22/borrel/