Information technology for identifying disinformation sources and inauthentic chat users' behaviours based on machine learning

Victoria Vysotska[1,†], Lyubomyr Chyrun[2,†], Sofia Chyrun[1,†] and Ilia Holets[1,*,†]

[1] *Lviv Polytechnic National University, Stepan Bandera 12, 79013 Lviv, Ukraine*
[2] *Ivan Franko National University of Lviv, University 1, 79000 Lviv, Ukraine*

Abstract

For the study, the main tasks were formulated as recognition of propaganda messages and analysis of the spread of propaganda messages between groups, thereby finding propaganda networks. The relevance, object and subject of the research are described. The influencing public opinion means are studied, in particular propaganda, in particular russian propaganda. A data search and review for the study is conducted. Several datasets of different structures are selected for further research. The tagged data quality, the tags balance and the empty values presence are checked. Several regularities of specific datasets are deduced (regarding the length of messages, and the use of certain emoticons and keywords). Processing of text data is carried out using a combination of different methods (conversion to lowercase, extraction of characters through regular expressions, extraction of stop words, stemming). The unique words present in the texts of the dataset are reviewed. A word frequencies table is compiled, which demonstrates some regularities for certain words. The best parameters for the binary classification of propaganda on machine learning models are selected, such as logistic regression (1-2-3-grams, 5000 features). A classification accuracy of ~0.85 is achieved. Binary classification is tested using artificial neural networks, namely: a simple fully connected neural network with count vectors at the input, a neural network with embeddings at the input, and a transformer. Different methods of measuring the similarity (distance) between texts are tested: cosine similarity with different types of input vectors, Jaccard similarity, Leventschein distance, cosine similarity on loaded word2vec-google-news-300 embeddings. The last method proved to be the best, so it is used further. On several datasets, part of the messages was compared with each other, using the chosen method of distance measurement. Similar messages are found, which most likely were retranslated, both within the same group and between different groups. A method of calculating the number of reinterpretations between groups has been created, which allows propaganda distribution networks recognition.

Keywords

Disinformation, fake, propaganda, linguistic analysis, natural language processing, machine learning, cyber warfare, artificial intelligence, semantic analysis, information security

MoDaST-2024: 6th International Workshop on Modern Data Science Technologies, May, 31 - June, 1, 2024, Lviv-Shatsk, Ukraine

[*] Corresponding author.
[†] These authors contributed equally.

victoria.a.vysotska@lpnu.ua (V. Vysotska); Lyubomyr.Chyrun@lnu.edu.ua (L. Chyrun); sofiia.chyrun.sa.2022@lpnu.ua (S. Chyrun); illia.holets.sa.2020@lpnu.ua (I. Holets)
0000-0001-6417-3689 (V. Vysotska); 0000-0002-9448-1751 (L. Chyrun); 0000-0002-2829-0164 (S. Chyrun); 0000-0003-4747-3684 (I. Holets)

1. Introduction

The spread of disinformation, propaganda, and fake information has become a particularly acute problem with the spread of the Internet and social media [1-3]. Now anyone can create their site or group on a social network and share almost any information. The problem has already been studied, but this area of research is still quite young. In addition, the methods of creating and spreading dishonest information are constantly changing and improving. In addition, during the war, the spread of russian propaganda is a big problem for Ukraine and the whole world [4-6].

Misinformation is defined as "factually incorrect information that is not supported by evidence." Disinformation in social networks has become an urgent and vital problem, especially in areas related to the war in Ukraine. Such information obtained from social media, including thematic online communities, can influence the results of public opinion formation, control public sentiment, and accordingly affect the course of war as a whole [1-3]. Concerns about misinformation have grown with the rise of requests for relevant information on social media. The lack of safeguards during discussions in online communities contributes to the spread and reinforcement of misinformation. The existing literature mostly focuses on the detection of fake reviews and fake news; however, the literature lacks a comprehensive theoretical framework designed to detect misinformation, especially in the context of an online community. Considering the huge amount of misinformation about the war in Ukraine that is spreading in the relevant online communities, there is a need to develop an effective model to achieve automatic detection of disinformation in the context of identification of inauthentic behaviour (bots) of coordinated groups. Stopping the spread of disinformation in social networks during an information war has long been a public concern, as the spread of such disinformation can hurt the population as a consumer of this content and, accordingly, the course of the war itself. Usually, the detection of thematic online disinformation is based on the linguistic features of the content of the textual content of the publications. But they multiply and spread faster than they can be identified and blocked. Therefore, identifying the sources of similar content, potential authors, and distribution mechanisms, i.e. analysis and identification of the behaviour of potential generators of fakes is a priority task for improving the means of cyber-fighting against disinformation on the Internet. Features of online disinformation can be classified into two levels: central (including features of the topic) and peripheral (including linguistic features, features of attitudes and features of user behaviour). Behavioural features need to be found to reflect user interaction characteristics: discussion initiation, interaction involvement, sphere of influence, relational mediation, and informational independence.

To build models and methods for identifying misinformation on the Internet, many researchers have devoted themselves to identifying the features of misinformation. Misinformation on social media can be seen as messages that are posted to persuade other users. To identify effective misinformation detection functions in online health communities, it is necessary to use a model that can help understand how misinformation on the Internet, particularly in social networks and online communities, persuades users. Users usually construct an attitude toward a message through both central and peripheral routes. In the central route, users scrutinize the quality and strength of information; whereas in the peripheral route, users care more about surface factors such as source reputation, visual appeal, and presentation. In addition to the content of the message, some secondary information (for example, the number of likes and stars) significantly increases the validity and reliability of messages. Therefore, message central-level functions persuade users based on message content, while peripheral-level functions persuade users through the influence of message authors. The best features for detecting misinformation in social networks may be those that look at user characteristics, messages, topics, and user behaviour. The creation of a misinformation detection model that combines central-level functions (in particular, topic features) and peripheral-level functions (in particular, linguistic features, mood features, and user behaviour features) requires further research. Based on these features, it is necessary to evaluate their ability to automatically distinguish misinformation from truth within a topical online community using various machine learning techniques. The developed system for rapid identification of sources of disinformation should be based on the analysis of the inauthentic behaviour of participants in the distribution of fakes. The results have not only demonstrated the effectiveness of behavioural features in disinformation detection but also offered both methodological and theoretical contributions to disinformation detection in terms of integrating features of messages as well as features of message authors. The project is aimed at the application of artificial intelligence for the development and improvement of cyber warfare tools, in particular for the fight against disinformation on the Internet, namely for the automatic detection of sources of disinformation and inauthentic behaviour (bots) of coordinated groups.

The goal of the project is to increase the level of information security of the state by developing mathematical models, methods and means of cyber-fighting against disinformation, in particular, automatic detection of sources of disinformation and inauthentic behaviour (bots) of coordinated groups on the Internet based on stylistic analysis and linguistic processing of the text of fakes and propaganda, their features distribution and reposting, as well as machine learning methods.

The main tasks of the project are to develop methods and tools for monitoring and detecting misinformation on the Internet, in particular:

1) stylistic analysis and linguistic processing of disinformation to identify common characteristic features of fakes of the same author's collective;

2) identification of disinformation that is potentially similar in style to form a set of potential authors and participants in the dissemination of propaganda;

3) identification of primary sources of the publication of disinformation based on the analysis of the results of the search for distribution routes of thematically and content-like texts to determine a set of criteria for evaluating the inauthentic behaviour of a group of participants;

4) analysis of inauthentic behaviour of chat users to form their informational portraits with their classification, in particular, into people/bots;

5) implementation of an information system for identifying sources of misinformation and inauthentic behaviour of chat users, its experimental testing, collection/processing/analysis of the obtained results to calculate the accuracy/efficiency of functioning.

The purpose of the research is to recognize propaganda texts and ways of their distribution on the Internet, in particular, in social networks. The objectives of the research are:

1. Propaganda recognition in the text using computer linguistics and machine learning.
2. Recognition of similar propaganda messages.
3. Identification of propaganda distribution networks.

The object of the research is the processes and mechanisms of influence on public opinion through the dissemination of disinformation, propaganda and fake in mass media.

The subject of the study is the methods and means of spreading propaganda (especially russian) on the Internet.

The detection of propaganda in itself is not new, but it is also not fully researched, and in the context of the war in Ukraine and the fight against russian information attacks, this issue is more relevant than ever. Most of the research in this field was conducted on American data, such as the analysis of the situation of the election of Trump as the president of the United States. This study aims to adapt current knowledge to improve the situation in Ukraine.

2. The current state of the problem

Online media and social networks allow rapid exchange of information, including misinformation, both purposefully and randomly/chaotically [1-6]. Along with the main advantage as an organization of quick access for all those who want operational and up-to-date information, online media are often used to spread deliberately misleading content such as fakes and propaganda about specific events, people or organizations, including governments [7] Recently, vivid examples of the spread disinformation is the russian government's attempts to control information during the war in Ukraine since 2014, for example, the MH17 plane crash [8]. In parallel, much online information are subject to regional censorship in certain territorial regions due to political, economic, social, religious and other factors, for example, to control/manage the opinion of the people of that region, for example, in the occupied territories of russia to control the future voters of the bunker president. It is easy for an average person to get lost and navigate in this mass of content flow with contrary facts and causes of events/phenomena [9]. It is unethical, illegal and impractical to control what to show/hide (censor) Internet content to the average user in democracies without direct evidence of the presence of disinformation/fake/propaganda for a purposeful violation of the information security of an organization/country. This is one of the first steps in the transition to totalitarianism. Providing information, for example, to journalists about a possible thematic fake for conducting a journalistic investigation or warning the average reader about the possibility of disinformation in this content/resource is, on the one hand, support for freedom of speech, on the other hand, giving a person the opportunity to choose what to believe and what not. This makes it possible to gain an understanding of events and orientation in the flow of information both for solving everyday tasks and adjusting business strategies, etc. [10-15].

Blocking disinformation and sources of its dissemination, as well as identification of potential authors based on analysis of inauthentic behaviour, are usually the functional responsibilities of authorized bodies, especially during information warfare. But it is now so quickly and efficiently generated/distributed based on the use of modern information technologies and artificial intelligence that no one can cope with this task 100% without the use of new methods and tools based on machine learning [16-27]. Significant and massive dissemination of (dis)information against the background of the war in Ukraine without appropriate analysis potentially leads to panic among the relevant strata/region of the population, significantly affecting the process of adjusting plans/strategies of business, social services, etc. Against the background of the information war, a lot of time and resources are spent on the appropriate collection, analysis and formation of appropriate conclusions regarding the content of the relevant content. This is also influenced by the language of the information, which may partially/significantly change the content when translated. The system will not be able to completely replace human activity in this direction. But it can be a significant helper for the rapid formation of relevant bases of such content, stylistic and linguistic analysis of the disinformation text to form an informational portrait of the authors, search for authors and distributors based on the analysis of inauthentic behaviour and the results of the analysis of the style of content writing, as well as responding to local changes or dynamics changes in the flow of content, marking certain content as potentially fake in a certain percentage.

There will not be enough resources to fully analyze all new/old content. And by the time the analysis is done, the disinformation itself will become obsolete. And here is the quick formation/modification/replenishment of databases/databases of content marked as blocked/unblocked in a certain region, sorted by appropriate metrics (time, topic, blocking region, language, etc.) from relevant/relevant to less relevant for further analysis by methods/technologies NLP/ML will significantly speed up the process of navigating through the chaos of new information on the Internet. Determining the topic/reason for content blocking (censorship) in a certain region will improve the quality of identifying fakes/propaganda/disinformation on the relevant topic. Therefore, it is urgent and necessary to develop a system for the automatic detection of sources of disinformation and inauthentic behaviour of chat users in cyberspace. The system should be implemented based on new principles of information security (data monitoring, threat detection, forecasting), which will make it possible to identify, monitor, report on the threat level and predict cyber threats, as well as the degree of probable informational and psychological impact on public opinion. Because of this, this project is relevant, relevant, timely and promising for increasing the degree of information security of the state based on the identification, monitoring, forecasting and analysis of threats in the cyberspace of Ukraine.

3. The project's novelty

The scientific novelty consists in the development of the following methods:

- stylistic analysis and linguistic processing of disinformation to identify common characteristic features of fakes of one author's collective based on methods of processing natural language and artificial intelligence, linguistic analysis of information messages, classification/clustering of text, etc. to identify linguistic signs of destructive and manipulative attempts to influence the reader;

- detection of potentially similar disinformation in terms of style to form a set of potential authors and participants in the dissemination of propaganda based on the collection/monitoring/detection/classification of information threats in the Internet space;

- identification of primary sources of disinformation publication based on the analysis of the results of the search for distribution routes of thematically and content-similar texts to determine a set of criteria for evaluating the inauthentic behaviour of a group of participants based on the analysis of social networks through graph theory and intelligent data analysis.

The practical novelty consists in the development of an information system for detecting sources of misinformation and inauthentic behaviour of chat users, experimental testing, collection/processing/analysis of the obtained results to calculate the accuracy/efficiency of functioning based on the implementation of the following software modules as:

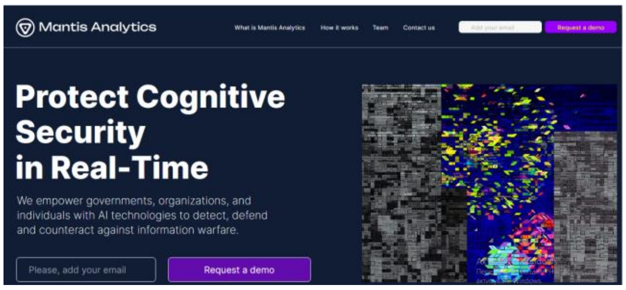
- a module for intelligent search, collection, marking, linguistic analysis and classification of information messages for the further formation of a set of potential fakes, as well as monitoring, management, detection and tracking of information threat data based on machine learning;

- a module of stylistic analysis of a set of fakes for identification of similar styles for one author collective with subsequent classification (human/bot) based on methods of machine learning and linguistic statistical data analysis;

- a module for the analysis of inauthentic behaviour of chat users to form their information portraits with their classification, in particular, into people/bots through the study of optimization models of attackers' actions based on the graph of reposts, optimization models and scenarios of inauthentic behaviour of participants, methods of intelligent search for disinformation distribution routes.

4. Related works

The only off-the-shelf solution similar enough to this research is Mantis Analytics (Fig. 1). Link: https://mantisanalytics.com/.

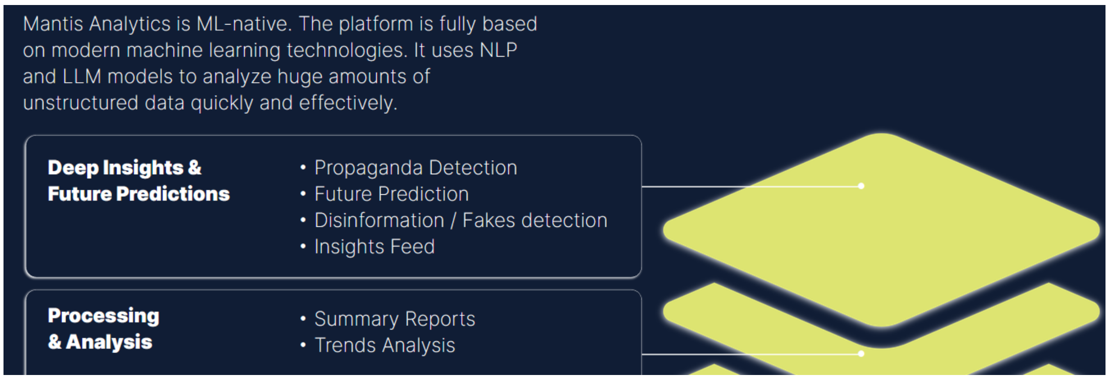


Protect Cognitive Security in Real-Time

We empower governments, organizations, and individuals with AI technologies to detect, defend and counteract against information warfare.

Figure 1: Mantis Analytics Home Page

The website states that the program uses machine learning, Natural Language Processing and Large Language Models to analyze a large amount of unstructured data (Fig. 2). Although the website describes how the program works and what tasks it can perform (Fig. 3), to use it, you need to enter your email address and request access to the demo version. Most likely, the program is not available to the general public, so it will not be considered further.

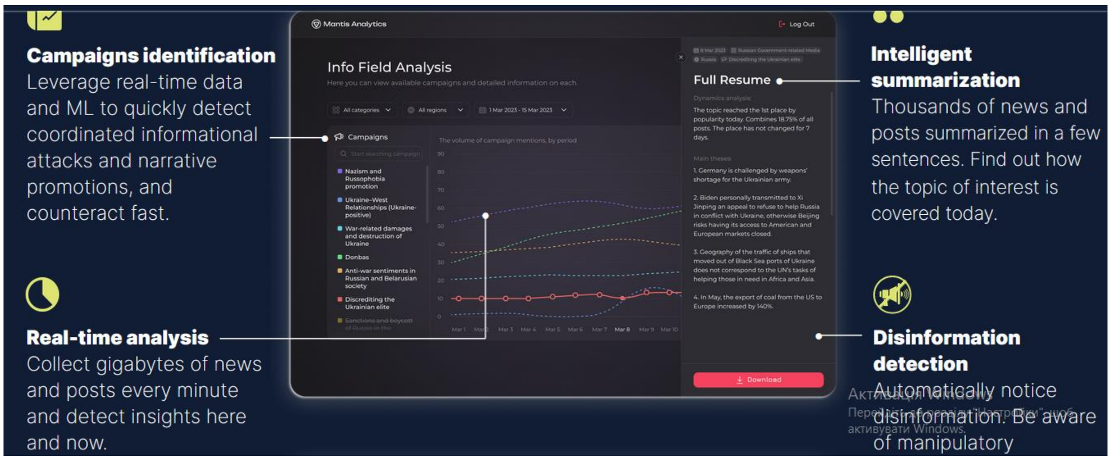


Mantis Analytics is ML-native. The platform is fully based on modern machine learning technologies. It uses NLP and LLM models to analyze huge amounts of unstructured data quickly and effectively.

Deep Insights & Future Predictions
- Propaganda Detection
- Future Prediction
- Disinformation / Fakes detection
- Insights Feed

Processing & Analysis
- Summary Reports
- Trends Analysis

Figure 2: A description of the program from Mantis Analytics



Campaigns identification
Leverage real-time data and ML to quickly detect coordinated informational attacks and narrative promotions, and counteract fast.

Real-time analysis
Collect gigabytes of news and posts every minute and detect insights here and now.

Intelligent summarization
Thousands of news and posts summarized in a few sentences. Find out how the topic of interest is covered today.

Disinformation detection
Automatically notice disinformation. Be aware of manipulatory

Mantis Analytics Log Out

Info Field Analysis
Here you can view available campaigns and detailed information on each.

All categories | All regions | 1 Mar 2023 - 15 Mar 2023

Campaigns
- Nazism and Russophobia promotion
- Ukraine-West Relationships (Ukraine-positive)
- War-related damages and destruction of Ukraine
- Donbas
- Anti-war sentiments in Russian and Belarusian society
- Discrediting the Ukrainian elite
- Sanctions and boycott of Russia in the

The volume of campaign mentions, by period

Full Resume
Dynamics analysis:
The topic reached the 1st place by popularity today. Combines 18.75% of all posts. The place has not changed for 7 days.

Main theses:
1. Germany is challenged by weapons' shortage for the Ukrainian army.
2. Biden personally transmitted to Xi Jinping an appeal to refuse to help Russia in conflict with Ukraine, otherwise Beijing risks having its access to American and European markets closed.
3. Geography of the traffic of ships that moved out of Black Sea ports of Ukraine does not correspond to the UN's tasks of helping those in need in Africa and Asia.
4. In May, the export of coal from the US to Europe increased by 140%.

Download

Figure 3: Features of Mantis Analytics

5. Research Methodology

As the basis of the research methodology, we offer a synthesized technology based on the methods of artificial intelligence, computer linguistics, machine learning, intelligent data analysis, statistical data processing, systems theory and system analysis, computer and simulation modelling, etc. The problem consists of two main components – identifying a set of information as fake and, based on it, finding sources and analyzing the inauthentic behaviour of participants.

The principle of operation of the information system of automatic detection of sources of disinformation and inauthentic behaviour of chat users:

Stage 1. Defining a set of information as fake:

Step 1.1. Collection and integration of the content of the relevant language from relevant resources in the Data Store.

Step 1.2. Checking whether content is blocked from a specific resource in a specific region.

Step 1.3. Marking of each content as blocked/unblocked in a certain region with corresponding additional metrics (time, resource, frequency of appearance of blocked/unblocked duplicates, presence of relevant marked words in the title/digest/annotation, for example, proper names, etc.).

Step 1.4. Formation of an intermediate database of branded sorted data.

Step 1.5. Applying top content NLP methods to calculate the potential of a fake and/or topic as a reason for blocking content in a certain region based on dictionaries and a set of metrics. NLP diagram of the content topic definition process:

1.5.1. Definition of a set of keywords of the relevant content and a set of available marker words (proper names, abbreviations, top words of the relevant topic, etc.). Determination, if possible, of the topic of the content (method of text classification).

1.5.2. If it is difficult to determine the topic by keywords - identify persistent phrases. Define if possible the content topic.

1.5.3. If it is difficult to define a topic based on persistent keywords, perform a semantic analysis and build an ontology. Define if possible the content topic.

1.5.4. If, according to the results of the semantic analysis, it is impossible to do this, mark it accordingly and transfer it to the list for the work of the content moderator.

1.5.5. For a specific topic, if the content is marked as blocked, check with the list of previously blocked topics in this region. If not, update the list. If there is to renew the number of blocks of this topic as censorship in a specific region.

Step 1.6. Applying ML technologies to improve data analysis/labelling/NLP. Pre-training ML models on a validated training dataset.

Step 1.7. Generating models/patterns of potential fakes to update the list of labelled content sorting metrics in step 1.3 and metrics/dictionaries for NLP.

Step 1.8. Constant updating of the intermediate database of branded sorted data and transfer of outdated content to the archive.

Step 1.9. Updating the training dataset to improve ML models. General scheme of the training and training process of the disinformation analysis module:

Pipeline 1.9.1Pre-Labeled Data $\rightarrow$ NLP $\rightarrow$ ML $\rightarrow$ Models/Patterns/Metrics.

Pipeline 1.9.2. Input new data $\rightarrow$ Data labelling (blocked/unblocked) $\rightarrow$ NLP $\rightarrow$ ML $\rightarrow$ Content labelling (fake/not fake) or finding a potential reason for blocking (not fake, but this particular event/topic is banned in a certain region for the average audience). The general scheme of the system is presented in Fig. 4.

Stage 2. Identification of sources and analysis of inauthentic behaviour of participants.

Step 2.1. Creation of a disinformation detection model that combines central-level features (in particular, topic features) and peripheral-level features (in particular, linguistic features, mood features, and user behaviour features).

Step 2.2. Evaluating the ability of central and peripheral features to automatically distinguish misinformation from truth within a topical online community using different machine learning techniques.

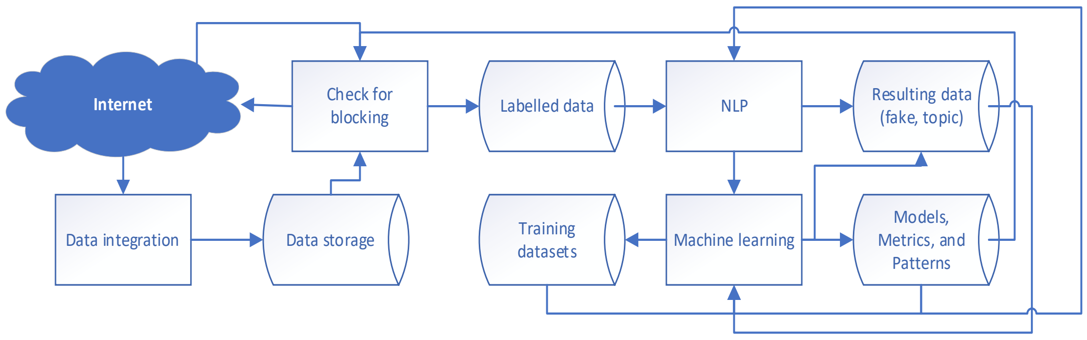
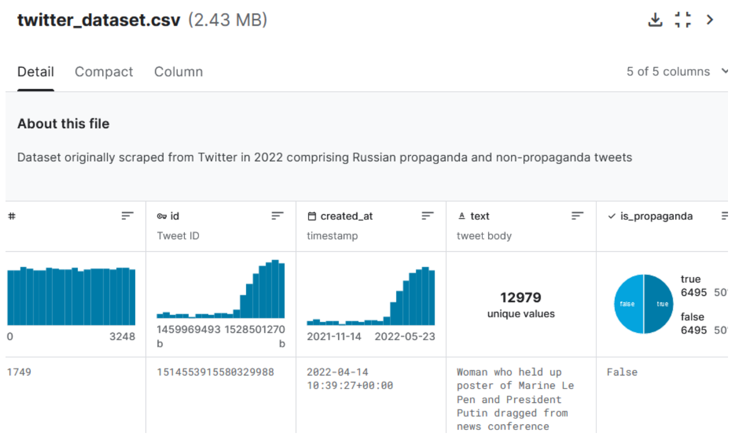


Figure 4: General scheme of the system for identifying disinformation, fakes and propaganda

Step 2.3. Intelligent search for fakes based on machine learning.
Step 2.4. Finding a set of stylistically similar fakes for one author.
Step 2.5. Finding the sources of the fake on the main analysis of the distribution graph.
Step 2.6. Analysis of the behaviour of the author/team/bot over a long period to form a set of main characteristic behavioural traits.
Step 2.7. Finding other fakes of the author by his writing style and behaviour.
Step 2.8. Formation of a portrait of the author's behaviour and behaviour prediction models.
Step 2.9. Based on the analysis of information portraits of various authors, form forecasts of the development and spread of fakes (frequency, density, subject matter), for example for informational and psychological operation (PSYOP).

6. Experiments, results and discussions

The goal of the project is to recognize propaganda and ways of its dissemination. To do this, we will analyze several datasets to study the parameters and markers of disinformation. The first dataset (Fig. 5) is the Twitter ru Propaganda Classification. The author is Bohdan Mynzar. The data are available in two copies - in English and Ukrainian. Among the important features: it contains almost 13 thousand records and has three useful features - date + time of creation, message text, and label (propaganda / not propaganda). Link to the dataset: https://www.kaggle.com/datasets/bohdanmynzar/twitter-propaganda-classification?resource=download. The second dataset (Fig. 6) is russian propaganda tweets. The author is Darius Alexandru. The dataset consists of two files - one contains only propaganda messages, and the other contains only non-propaganda messages. There are no labels in the dataset, but the very knowledge of broken files covers their absence. Of the important features, the dataset has 22,000 records, more than 30 features, but most of them are not very useful because they have a lot of empty values (90+%). Important and filled-in features are the date of posting the message, the time of posting, the name of the group that posted, and the text of the message. Link to the dataset: https://www.kaggle.com/datasets/dariusalexandru/russian-propaganda-tweets-vs-western-tweets-war?select=russian_propaganda_tweets.csv.


twitter_dataset.csv (2.43 MB)

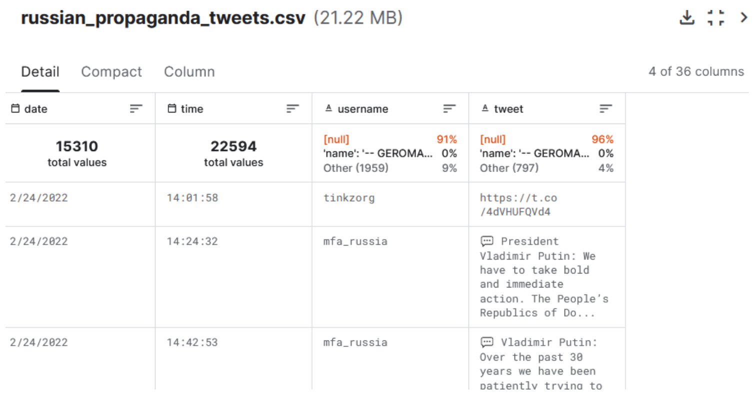
Detail Compact Column 5 of 5 columns

About this file

Dataset originally scraped from Twitter in 2022 comprising Russian propaganda and non-propaganda tweets

#	id	created_at	text	is_propaganda
	Tweet ID	timestamp	tweet body	
0 3248	1459969493 b 1528501270 b	2021-11-14 2022-05-23	12979 unique values	true 6495 50 false 6495 50
1749	1514553915580329988	2022-04-14 10:39:27+00:00	Woman who held up poster of Marine Le Pen and President Putin dragged from news conference	False

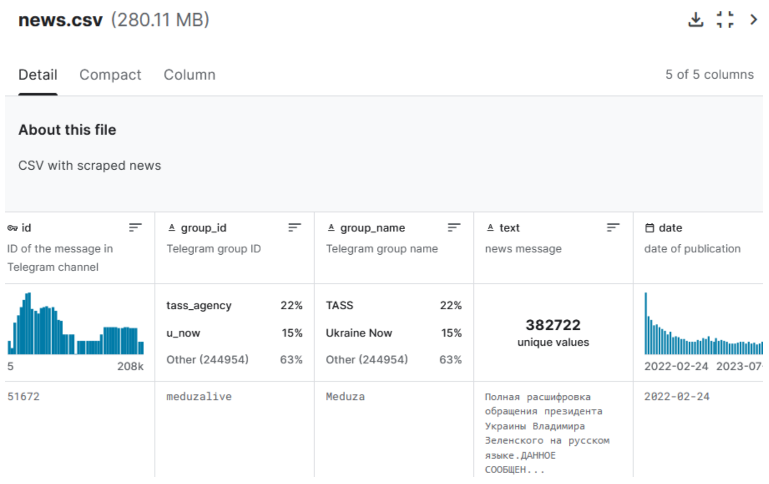
Figure 5: View of the first dataset


russian_propaganda_tweets.csv (21.22 MB)

Detail Compact Column 4 of 36 columns

date	time	username	tweet
15310 total values	22594 total values	[null] 91% 'name': '-- GEROMA... 0% Other (1959) 9%	[null] 96% 'name': '-- GEROMA... 0% Other (797) 4%
2/24/2022	14:01:58	tinkzorg	https://t.co /4dVHUFQVd4
2/24/2022	14:24:32	mfa_russia	President Vladimir Putin: We have to take bold and immediate action. The People's Republics of Do...
2/24/2022	14:42:53	mfa_russia	Vladimir Putin: Over the past 30 years we have been patiently trying to

Figure 6: View of the second dataset


news.csv (280.11 MB)

Detail Compact Column 5 of 5 columns

About this file

CSV with scraped news

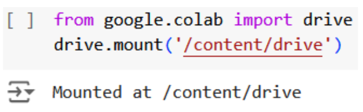
id	group_id	group_name	text	date
ID of the message in Telegram channel	Telegram group ID	Telegram group name	news message	date of publication
5 208k	tass_agency 22% u_now 15% Other (244954) 63%	TASS 22% Ukraine Now 15% Other (244954) 63%	382722 unique values	2022-02-24 2023-07
51672	meduzalive	Meduza	Полная расшифровка обращения президента Украины Владимира Зеленского на русском языке.ДАННОЕ СООБЩЕН...	2022-02-24

Figure 7: View of the third dataset

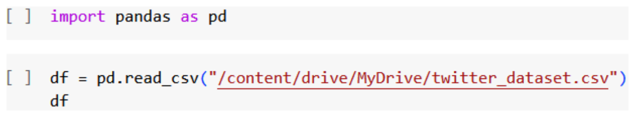
The third dataset (Fig. 7) – russian invasion of Ukraine | Live News. The author is Hladkiy Ivan. The dataset consists of one file containing messages from the Telegram social network. Contains more than 400 thousand records. Contains three important features - group name, publication date, and message text. Link: https://www.kaggle.com/datasets/falloutbabe/russian-invasion-of-ukraine-live-news-dataset. For simplicity, the first dataset will be used at the beginning, as it contains enough, but not too many records and features, and has English text and labels.

The project is implemented in the Google Colab environment in Python Notebook. The advantages of working in Python Notebook are the ability to conveniently run small blocks of code and write program comments in separate text blocks. The advantages of using Google Colab are the ability to easily connect to various Google services and the ability to work from any place where you can enter your browser and your account.

First, we will download the datasets to our Google Drive, and then use google.colab library + the drive function and the pandas library, we will transfer the first dataset from the drive to the collab notebook.

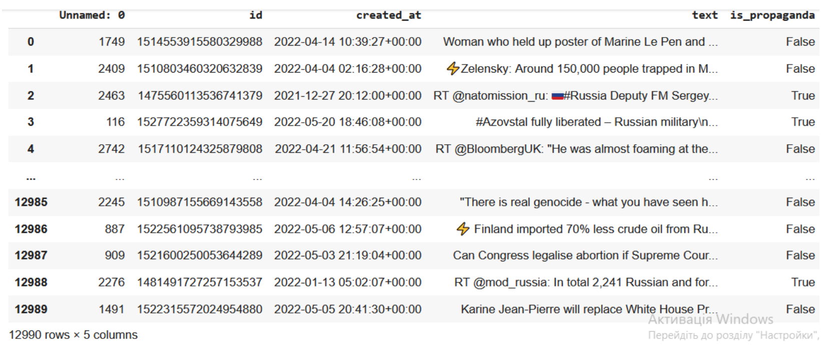


```
[ ] from google.colab import drive
    drive.mount('/content/drive')

Mounted at /content/drive
```



```
[ ] import pandas as pd

[ ] df = pd.read_csv("/content/drive/MyDrive/twitter_dataset.csv")
    df
```

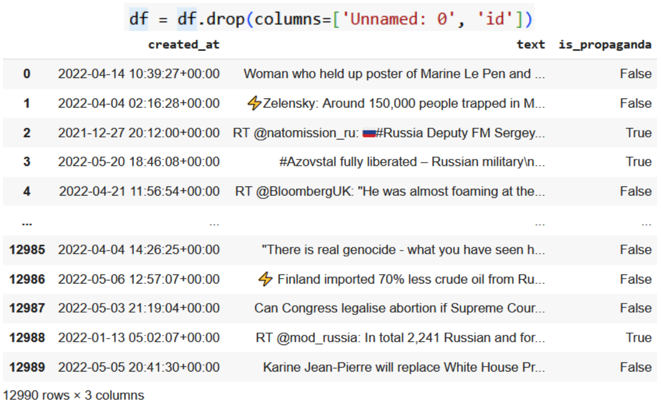
This is what the dataset looks like in the notebook:



	Unnamed: 0	id	created_at	text	is_propaganda
0	1749	1514553915580329988	2022-04-14 10:39:27+00:00	Woman who held up poster of Marine Le Pen and ...	False
1	2409	1510803460320632839	2022-04-04 02:16:28+00:00	⚡Zelensky: Around 150,000 people trapped in M...	False
2	2463	1475560113536741379	2021-12-27 20:12:00+00:00	RT @natomission_ru: 🇷🇺#Russia Deputy FM Sergey...	True
3	116	1527722359314075649	2022-05-20 18:46:08+00:00	#Azovstal fully liberated – Russian military\n...	True
4	2742	1517110124325879808	2022-04-21 11:56:54+00:00	RT @BloombergUK: "He was almost foaming at the...	False
...	...	...	...	...	...
12985	2245	1510987155669143558	2022-04-04 14:26:25+00:00	"There is real genocide - what you have seen h...	False
12986	887	1522561095738793985	2022-05-06 12:57:07+00:00	⚡ Finland imported 70% less crude oil from Ru...	False
12987	909	1521600250053644289	2022-05-03 21:19:04+00:00	Can Congress legalise abortion if Supreme Cour...	False
12988	2276	1481491727257153537	2022-01-13 05:02:07+00:00	RT @mod_russia: In total 2,241 Russian and for...	True
12989	1491	1522315572024954880	2022-05-05 20:41:30+00:00	Karine Jean-Pierre will replace White House Pr...	False

12990 rows × 5 columns

In the dataset, the first two columns do not carry important information, so we discard them.

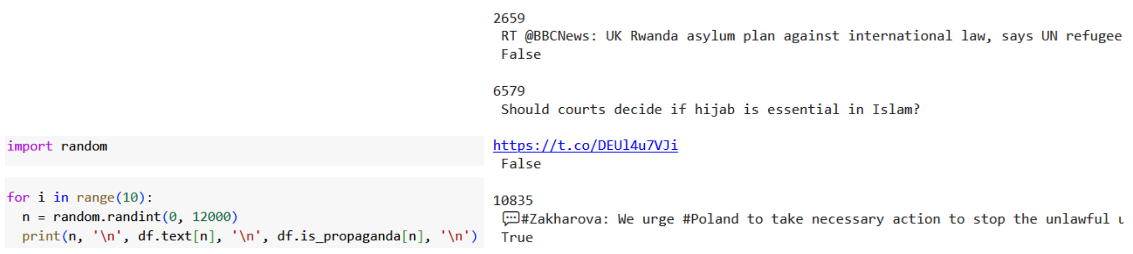


```
df = df.drop(columns=['Unnamed: 0', 'id'])
```

	created_at	text	is_propaganda
0	2022-04-14 10:39:27+00:00	Woman who held up poster of Marine Le Pen and ...	False
1	2022-04-04 02:16:28+00:00	⚡Zelensky: Around 150,000 people trapped in M...	False
2	2021-12-27 20:12:00+00:00	RT @natomission_ru: 🇷🇺#Russia Deputy FM Sergey...	True
3	2022-05-20 18:46:08+00:00	#Azovstal fully liberated – Russian military\n...	True
4	2022-04-21 11:56:54+00:00	RT @BloombergUK: "He was almost foaming at the...	False
...	...	...	...
12985	2022-04-04 14:26:25+00:00	"There is real genocide - what you have seen h...	False
12986	2022-05-06 12:57:07+00:00	⚡ Finland imported 70% less crude oil from Ru...	False
12987	2022-05-03 21:19:04+00:00	Can Congress legalise abortion if Supreme Cour...	False
12988	2022-01-13 05:02:07+00:00	RT @mod_russia: In total 2,241 Russian and for...	True
12989	2022-05-05 20:41:30+00:00	Karine Jean-Pierre will replace White House Pr...	False

12990 rows × 3 columns

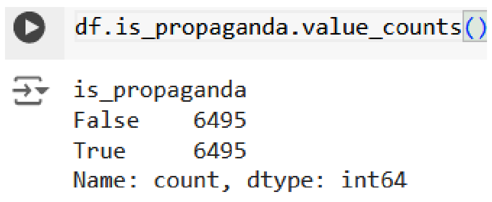
Let's check how well-marked the data are, since the assessment of whether a message is propaganda is a subjective decision of the author of the dataset. To do this, we output ten random messages, analyze their text and evaluate whether the label is well placed.



```
import random

for i in range(10):
  n = random.randint(0, 12000)
  print(n, '\n', df.text[n], '\n', df.is_propaganda[n], '\n')
```

```
2659
 RT @BBCNews: UK Rwanda asylum plan against international law, says UN refugee
 False

6579
 Should courts decide if hijab is essential in Islam?
https://t.co/DEUl4u7VJi
 False

10835
 💬#Zakharova: We urge #Poland to take necessary action to stop the unlawful u
 True
```

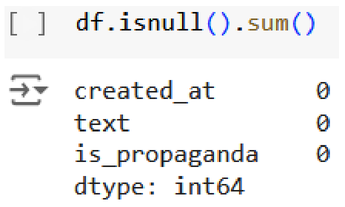
In our opinion, the dataset is labelled adequately. Next, we check whether the dataset is balanced.



```
df.is_propaganda.value_counts()
```

```
is_propaganda
False     6495
True      6495
Name: count, dtype: int64
```

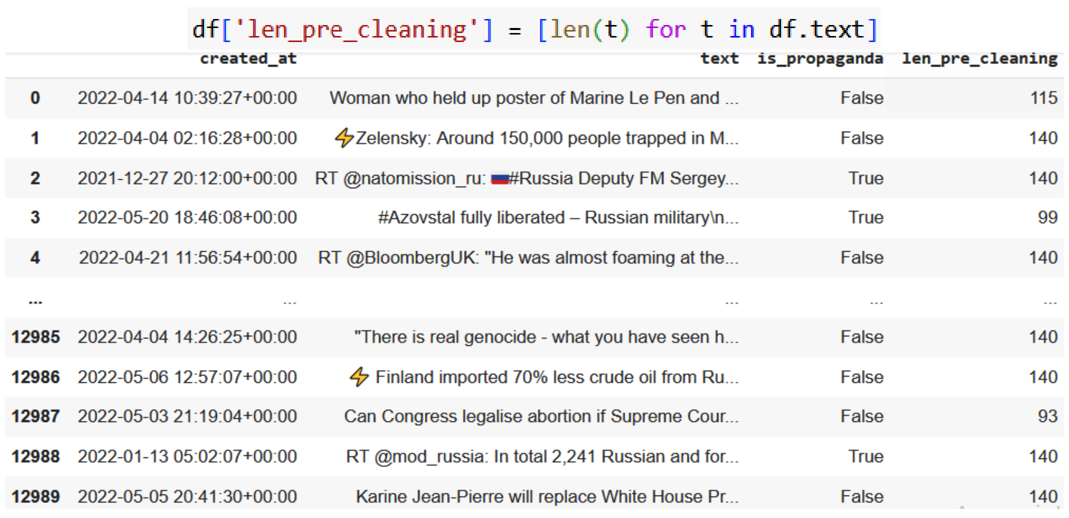
The dataset contains an equal number of positive and negative labels, so it is perfectly balanced.



```
[ ]  df.isnull().sum()
```

```
created_at       0
text             0
is_propaganda    0
dtype: int64
```

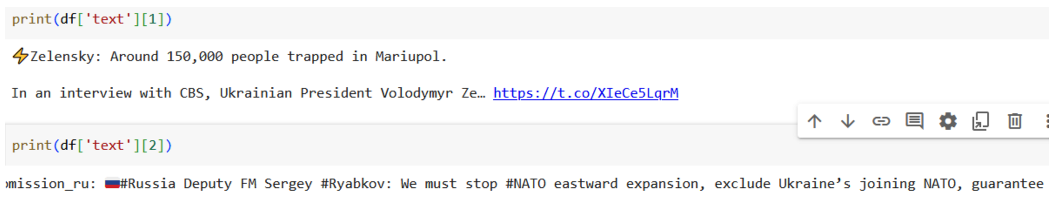
It also does not contain empty values. For further checks, add a column with the length of the message to the dataframe.



```
df['len_pre_cleaning'] = [len(t) for t in df.text]
```

	created_at	text	is_propaganda	len_pre_cleaning
0	2022-04-14 10:39:27+00:00	Woman who held up poster of Marine Le Pen and ...	False	115
1	2022-04-04 02:16:28+00:00	⚡Zelensky: Around 150,000 people trapped in M...	False	140
2	2021-12-27 20:12:00+00:00	RT @natomission_ru: 🇷🇺#Russia Deputy FM Sergey...	True	140
3	2022-05-20 18:46:08+00:00	#Azovstal fully liberated – Russian military\n...	True	99
4	2022-04-21 11:56:54+00:00	RT @BloombergUK: "He was almost foaming at the...	False	140
...	...	...	...	...
12985	2022-04-04 14:26:25+00:00	"There is real genocide - what you have seen h...	False	140
12986	2022-05-06 12:57:07+00:00	⚡ Finland imported 70% less crude oil from Ru...	False	140
12987	2022-05-03 21:19:04+00:00	Can Congress legalise abortion if Supreme Cour...	False	93
12988	2022-01-13 05:02:07+00:00	RT @mod_russia: In total 2,241 Russian and for...	True	140
12989	2022-05-05 20:41:30+00:00	Karine Jean-Pierre will replace White House Pr...	False	140

An interesting feature is that many messages are 140 characters long.



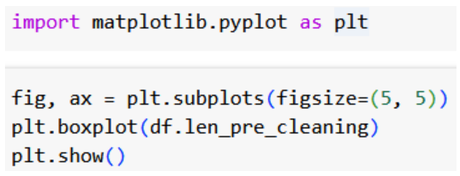
```
print(df['text'][1])
```

```
⚡Zelensky: Around 150,000 people trapped in Mariupol.

In an interview with CBS, Ukrainian President Volodymyr Ze… https://t.co/XIeCe5LqrM
```

```
print(df['text'][2])
```

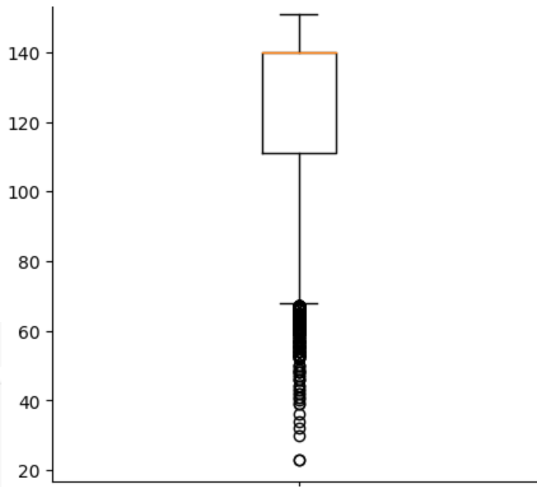
```
mission_ru: 🇷🇺#Russia Deputy FM Sergey #Ryabkov: We must stop #NATO eastward expansion, exclude Ukraine's joining NATO, guarantee
```

The texts of the messages are probably truncated. This may be caused either by the desire of the author of the dataset or by the fact that at one time the maximum message length on Twitter was 140 characters. Also interesting is that Google Colab understands emoticons from messages. Using the matplotlib library, we will display the average length of messages as a graph.

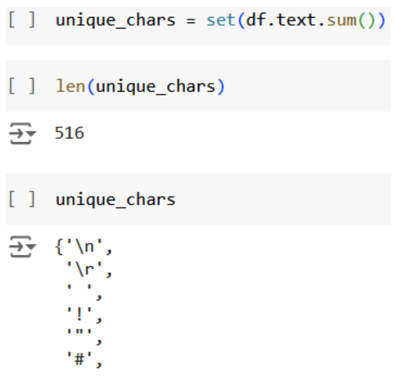


```
import matplotlib.pyplot as plt
```

```
fig, ax = plt.subplots(figsize=(5, 5))
plt.boxplot(df.len_pre_cleaning)
plt.show()
```

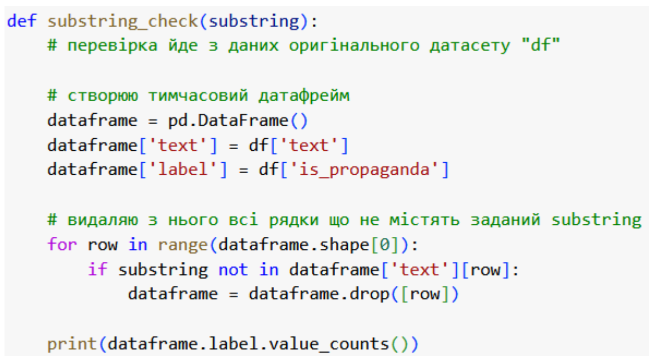


Most messages are close to the maximum length. This is probably because, in the news, proper names, first names and surnames of politicians are often used, which takes up a lot of characters. We are interested in emoticons from those messages, so let's analyze them a little. First, let's output all the unique characters that are in the texts.

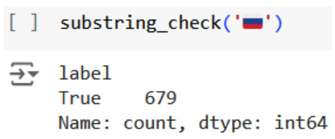


```
[ ]  unique_chars = set(df.text.sum())

[ ]  len(unique_chars)

     516

[ ]  unique_chars

     {'\n',
      '\r',
      ' ',
      '!',
      '"',
      '#',
```

In total, there are 516 unique symbols, most of them are letters, numbers, all kinds of symbols and emoticons. We do not see the point in analyzing them all, but we will analyze some of them through the author's substring_check function, which accepts a certain string as input, not necessarily a smiley, counts all the texts of messages containing this string, and outputs how many of such messages are propaganda and how many not.

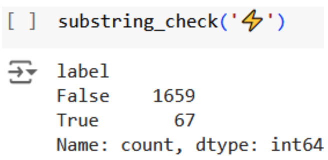


```
def substring_check(substring):
    # перевірка йде з даних оригінального датасету "df"

    # створюю тимчасовий датафрейм
    dataframe = pd.DataFrame()
    dataframe['text'] = df['text']
    dataframe['label'] = df['is_propaganda']

    # видаляю з нього всі рядки що не містять заданий substring
    for row in range(dataframe.shape[0]):
        if substring not in dataframe['text'][row]:
            dataframe = dataframe.drop([row])

    print(dataframe.label.value_counts())
```



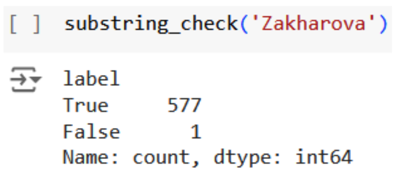
```
[ ]  substring_check('🇷🇺')

     label
     True     679
     Name: count, dtype: int64
```

For example, all 679 messages containing the russian flag emoticon are propaganda.



```
[ ]  substring_check('⚡')

     label
     False    1659
     True       67
     Name: count, dtype: int64
```

From my own experience, the lightning emoticon is often found in Ukrainian groups, so it is expected that the proportion of normal messages will be higher.

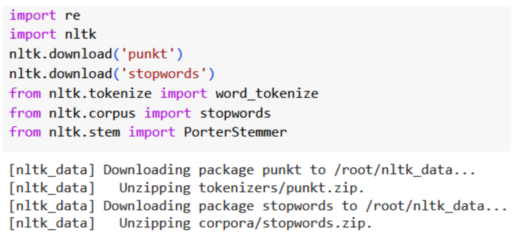


```
[ ]  substring_check('Zakharova')

     label
     True      577
     False       1
     Name: count, dtype: int64
```

99% of messages containing the surname "Zakharova" are propaganda.

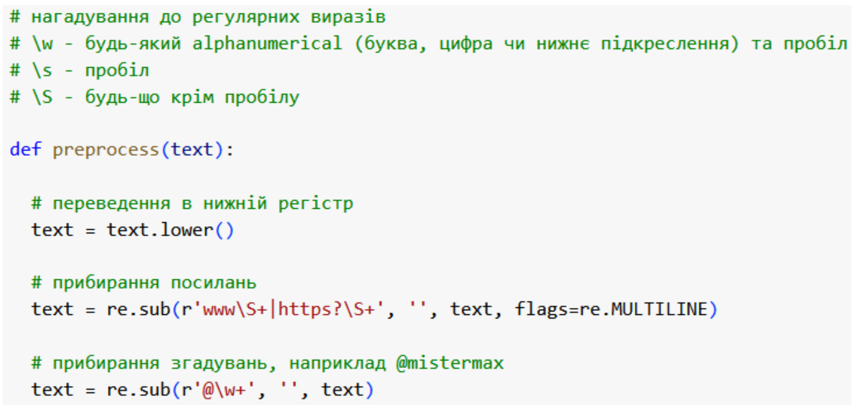
To continue working with the text, it is necessary to remove everything unnecessary for this task (noise). This means uppercase letters, punctuation marks, emoticons, and so on.

To do this, we will write one large function, into which you can pass text and receive processed text that has passed all the planned stages. The following methods will be used in the function: converting to lowercase, cleaning up redundant character structures like links, mentioning people in messages, and everything else, with the help of the "re" regular expression library. Also, with the help of the "nltk" library, we will remove stop words from the texts (for example, the, are, or...) and conduct word stemming so that words in different forms are counted as one (for example, imported -> import, replace ->replac...).

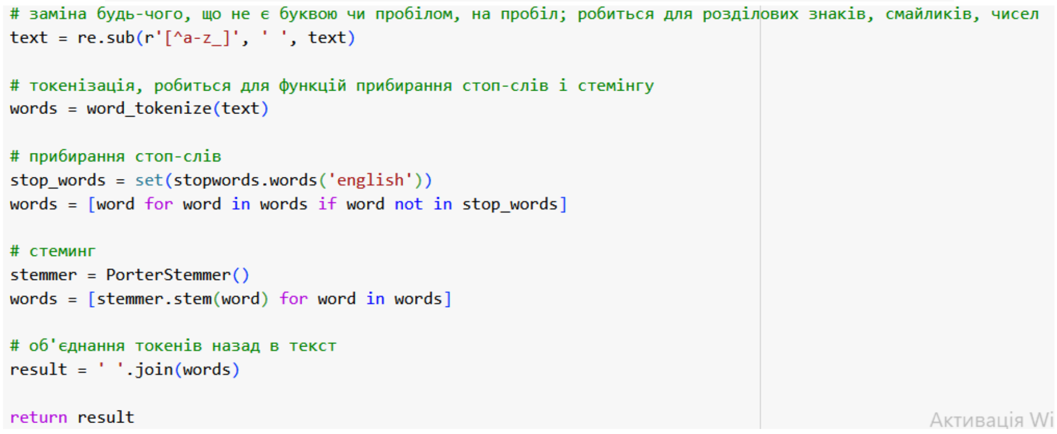


```
import re
import nltk
nltk.download('punkt')
nltk.download('stopwords')
from nltk.tokenize import word_tokenize
from nltk.corpus import stopwords
from nltk.stem import PorterStemmer
```

```
[nltk_data] Downloading package punkt to /root/nltk_data...
[nltk_data]   Unzipping tokenizers/punkt.zip.
[nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data]   Unzipping corpora/stopwords.zip.
```

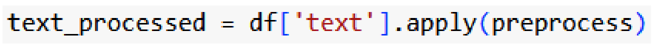


```
# нагадування до регулярних виразів
# \w - будь-який alphanumerical (буква, цифра чи нижнє підкреслення) та пробіл
# \s - пробіл
# \S - будь-що крім пробілу

def preprocess(text):

  # переведення в нижній регістр
  text = text.lower()

  # прибирання посилань
  text = re.sub(r'www\S+|https?\S+', '', text, flags=re.MULTILINE)

  # прибирання згадувань, наприклад @mistermax
  text = re.sub(r'@\w+', '', text)
```

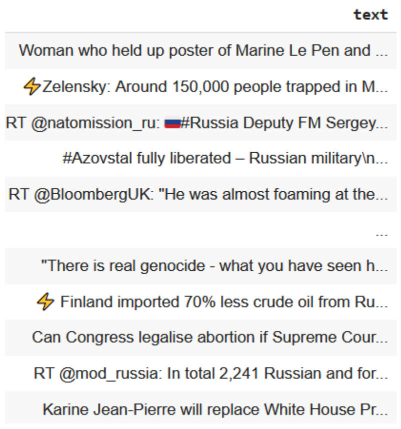


```
# заміна будь-чого, що не є буквою чи пробілом, на пробіл; робиться для розділових знаків, смайликів, чисел
text = re.sub(r'[^a-z_]', ' ', text)

# токенізація, робиться для функцій прибирання стоп-слів і стемінгу
words = word_tokenize(text)

# прибирання стоп-слів
stop_words = set(stopwords.words('english'))
words = [word for word in words if word not in stop_words]

# стеминг
stemmer = PorterStemmer()
words = [stemmer.stem(word) for word in words]

# об'єднання токенів назад в текст
result = ' '.join(words)

return result
```

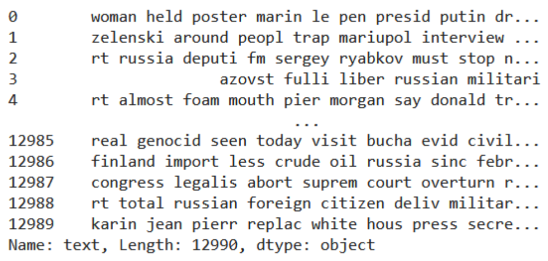
Now we will apply the function on the texts from the dataset, the Before and After view is shown below.



```
text_processed = df['text'].apply(preprocess)
```

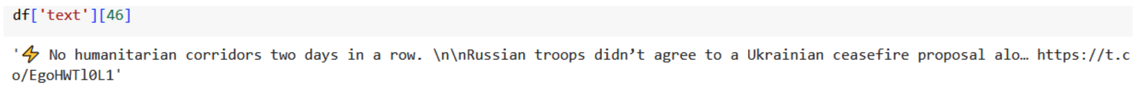


text
Woman who held up poster of Marine Le Pen and ...
⚡Zelensky: Around 150,000 people trapped in M...
RT @natomission_ru: 🇷🇺#Russia Deputy FM Sergey...
#Azovstal fully liberated – Russian military\n...
RT @BloombergUK: "He was almost foaming at the...
...
"There is real genocide - what you have seen h...
⚡ Finland imported 70% less crude oil from Ru...
Can Congress legalise abortion if Supreme Cour...
RT @mod_russia: In total 2,241 Russian and for...
Karine Jean-Pierre will replace White House Pr...



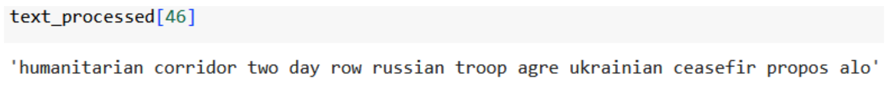
```
0        woman held poster marin le pen presid putin dr...
1        zelenski around peopl trap mariupol interview ...
2        rt russia deputi fm sergey ryabkov must stop n...
3                       azovst fulli liber russian militari
4        rt almost foam mouth pier morgan say donald tr...
                               ...
12985    real genocid seen today visit bucha evid civil...
12986    finland import less crude oil russia sinc febr...
12987    congress legalis abort suprem court overturn r...
12988    rt total russian foreign citizen deliv militar...
12989    karin jean pierr replac white hous press secre...
Name: text, Length: 12990, dtype: object
```

We will also demonstrate the Before and After view on the example of one random message.



```
df['text'][46]
```

```
'⚡ No humanitarian corridors two days in a row. \n\nRussian troops didn’t agree to a Ukrainian ceasefire proposal alo… https://t.c
o/EgoHWTl0L1'
```

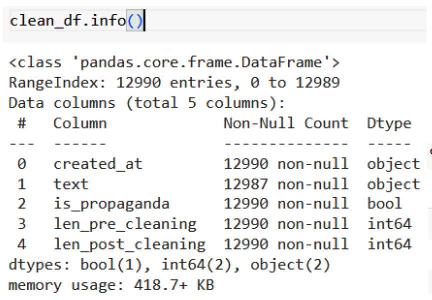


```
text_processed[46]
```

```
'humanitarian corridor two day row russian troop agre ukrainian ceasefir propos alo'
```

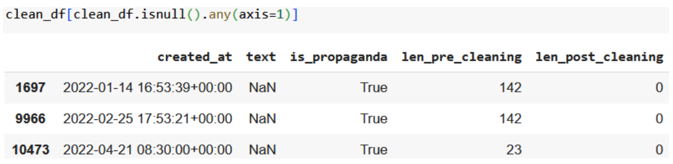
Add the Message length column after processing to the dataframe, and save this dataframe as a clean_df file for convenience.

After processing the messages, it was noticed that the length of some texts became 0.



```
clean_df.info()
```

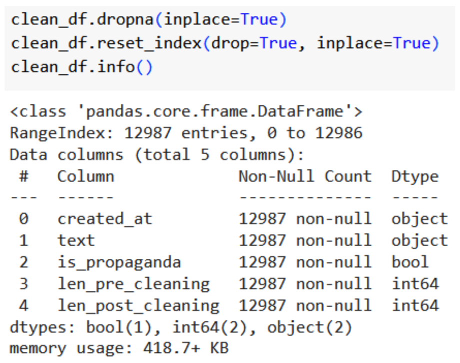
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 12990 entries, 0 to 12989
Data columns (total 5 columns):
 #   Column             Non-Null Count  Dtype
---  ------             --------------  -----
 0   created_at         12990 non-null  object
 1   text               12987 non-null  object
 2   is_propaganda      12990 non-null  bool
 3   len_pre_cleaning   12990 non-null  int64
 4   len_post_cleaning  12990 non-null  int64
dtypes: bool(1), int64(2), object(2)
memory usage: 418.7+ KB
```



```
clean_df[clean_df.isnull().any(axis=1)]
```

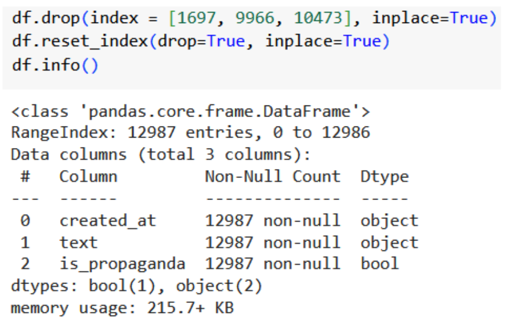
	created_at	text	is_propaganda	len_pre_cleaning	len_post_cleaning
1697	2022-01-14 16:53:39+00:00	NaN	True	142	0
9966	2022-02-25 17:53:21+00:00	NaN	True	142	0
10473	2022-04-21 08:30:00+00:00	NaN	True	23	0

There are only 3 such lines and, most likely, they did not contain useful information, so they became zero. Therefore, we simply discard them from the dataset and update the indices of other rows both in clean_df and in the original df.



```
clean_df.dropna(inplace=True)
clean_df.reset_index(drop=True, inplace=True)
clean_df.info()
```

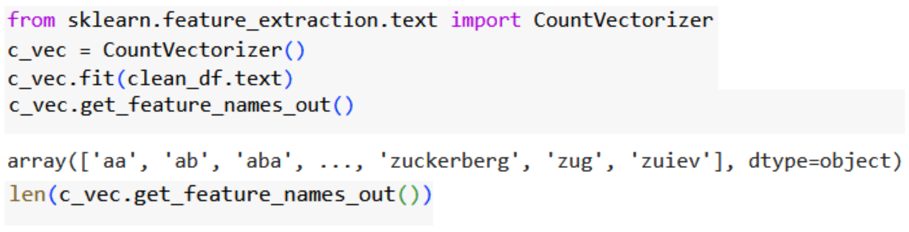
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 12987 entries, 0 to 12986
Data columns (total 5 columns):
 #   Column             Non-Null Count  Dtype
---  ------             --------------  -----
 0   created_at         12987 non-null  object
 1   text               12987 non-null  object
 2   is_propaganda      12987 non-null  bool
 3   len_pre_cleaning   12987 non-null  int64
 4   len_post_cleaning  12987 non-null  int64
dtypes: bool(1), int64(2), object(2)
memory usage: 418.7+ KB
```



```
df.drop(index = [1697, 9966, 10473], inplace=True)
df.reset_index(drop=True, inplace=True)
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 12987 entries, 0 to 12986
Data columns (total 3 columns):
 #   Column         Non-Null Count  Dtype
---  ------         --------------  -----
 0   created_at     12987 non-null  object
 1   text           12987 non-null  object
 2   is_propaganda  12987 non-null  bool
dtypes: bool(1), object(2)
memory usage: 215.7+ KB
```

Next, let's check how many unique words there are in the dataset. To do this, we import CountVectorizer from the sklearn library.



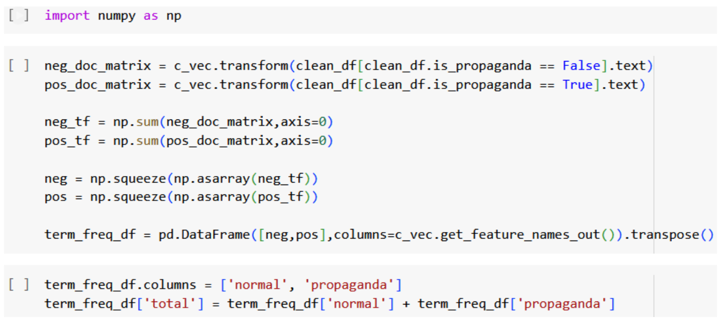
```
from sklearn.feature_extraction.text import CountVectorizer
c_vec = CountVectorizer()
c_vec.fit(clean_df.text)
c_vec.get_feature_names_out()
```

```
array(['aa', 'ab', 'aba', ..., 'zuckerberg', 'zug', 'zuiev'], dtype=object)
```

```
len(c_vec.get_feature_names_out())
```

```
11747
```

We will also compile a word frequency table, which shows which words are used most often in general, in propaganda and non-propaganda. To do this, import the numpy library and use the already created CountVectorizer.

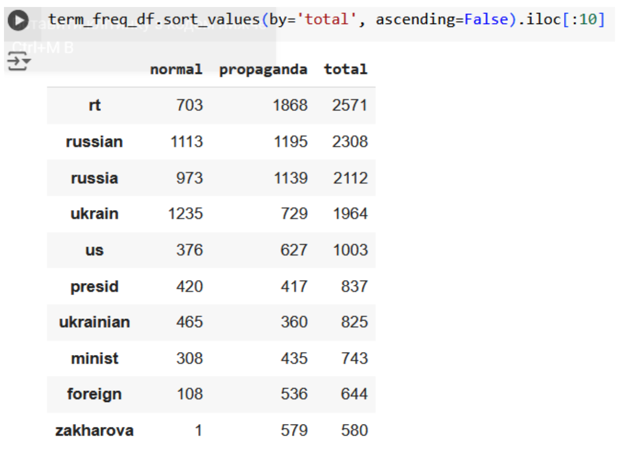


```
import numpy as np
```

```
neg_doc_matrix = c_vec.transform(clean_df[clean_df.is_propaganda == False].text)
pos_doc_matrix = c_vec.transform(clean_df[clean_df.is_propaganda == True].text)

neg_tf = np.sum(neg_doc_matrix,axis=0)
pos_tf = np.sum(pos_doc_matrix,axis=0)

neg = np.squeeze(np.asarray(neg_tf))
pos = np.squeeze(np.asarray(pos_tf))

term_freq_df = pd.DataFrame([neg,pos],columns=c_vec.get_feature_names_out()).transpose()
```

```
term_freq_df.columns = ['normal', 'propaganda']
term_freq_df['total'] = term_freq_df['normal'] + term_freq_df['propaganda']
```

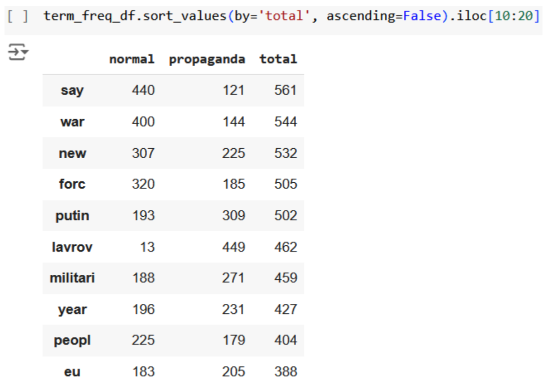
We will display the top 10 words by frequency of appearance from the table.



```
term_freq_df.sort_values(by='total', ascending=False).iloc[:10]
```

	normal	propaganda	total
rt	703	1868	2571
russian	1113	1195	2308
russia	973	1139	2112
ukrain	1235	729	1964
us	376	627	1003
presid	420	417	837
ukrainian	465	360	825
minist	308	435	743
foreign	108	536	644
zakharova	1	579	580

The table shows that the word "rt" comes across most often. This is the name of a propaganda group, and most likely there are many messages from it in the dataset, which is why the word is so common. Given the topic of the dataset, frequent words such as russia, russian, Ukraine and Ukrainian are also expected. The table also shows interesting dependencies. For example, in propaganda, the word foreign is often used, that is, "foreign", probably to set the population against everything foreign. Also, what has already been found, 99% of messages containing the surname "Zakharov" are propaganda. Let's display the next top 10 words.



```
[ ] term_freq_df.sort_values(by='total', ascending=False).iloc[10:20]
```

	normal	propaganda	total
say	440	121	561
war	400	144	544
new	307	225	532
forc	320	185	505
putin	193	309	502
lavrov	13	449	462
militari	188	271	459
year	196	231	427
peopl	225	179	404
eu	183	205	388

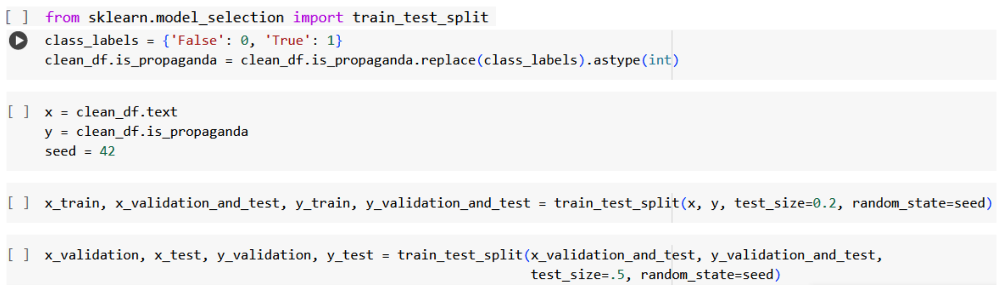
Interesting finds continue. Propaganda rarely uses the word "war". This is probably because they do not like this word there, and more often they use words like "special operation". Also, approximately 99% of messages with the surname "lavrov" are propaganda.

The next step will be to describe methods of text classification, to search for the best parameters.

The first idea for recognizing propaganda was a binary classification (propaganda / not propaganda). There was also the idea of a multi-class classification to understand what specific type of propaganda was used (appeals to authority, cult of personality, labelling...). However, large datasets were not necessary for this, so we settled on binary classification.

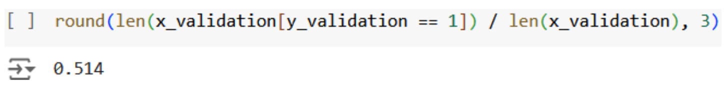
To do this, we will first use simple models, for example, logistic regression, and then we will try to build artificial neural networks. To assess the quality of the model, it is necessary to determine the minimum baseline accuracy. Then you can choose the best parameters, such as the number of n-grams in the text and the number of features, and train the model on such data.

First, let's convert the labels True and False into numbers 1 and 0, and also split the data into training, testing, and validation using the train_test_split function from the sklearn library.



```
[ ] from sklearn.model_selection import train_test_split

    class_labels = {'False': 0, 'True': 1}
    clean_df.is_propaganda = clean_df.is_propaganda.replace(class_labels).astype(int)

[ ] x = clean_df.text
    y = clean_df.is_propaganda
    seed = 42

[ ] x_train, x_validation_and_test, y_train, y_validation_and_test = train_test_split(x, y, test_size=0.2, random_state=seed)

[ ] x_validation, x_test, y_validation, y_test = train_test_split(x_validation_and_test, y_validation_and_test,
                                                                  test_size=.5, random_state=seed)
```

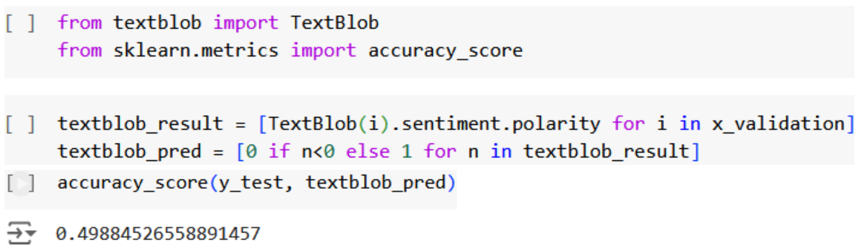
We determine the baseline accuracy by the number of positive labels in the validation set.



```
[ ] round(len(x_validation[y_validation == 1]) / len(x_validation), 3)

    0.514
```

This means that if just any text is classified as propaganda, the accuracy will be 0.514. Therefore, future models should be more accurate than this.

We import and check the finished TextBlob solution.



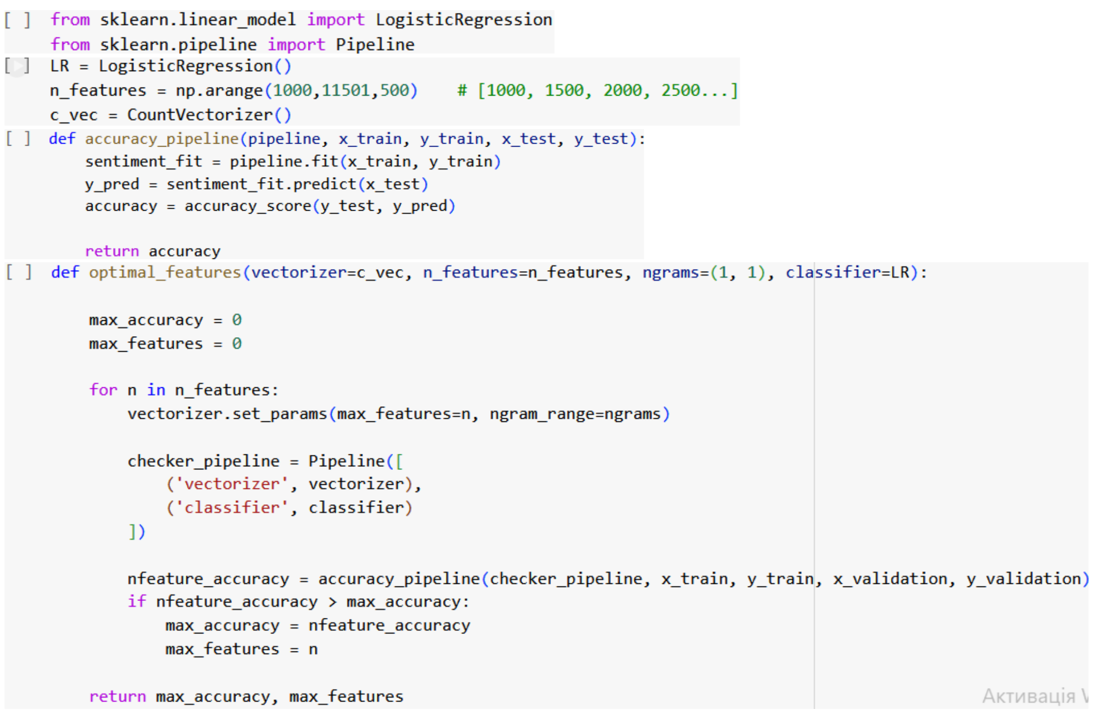
```
[ ] from textblob import TextBlob
    from sklearn.metrics import accuracy_score

[ ] textblob_result = [TextBlob(i).sentiment.polarity for i in x_validation]
    textblob_pred = [0 if n<0 else 1 for n in textblob_result]
[ ] accuracy_score(y_test, textblob_pred)

    0.49884526558891457
```

It turned out to be about the same as the baseline accuracy, even a little worse, so it doesn't fit.

Let's try to use logistic regression for classification, selecting the best parameters in parallel.

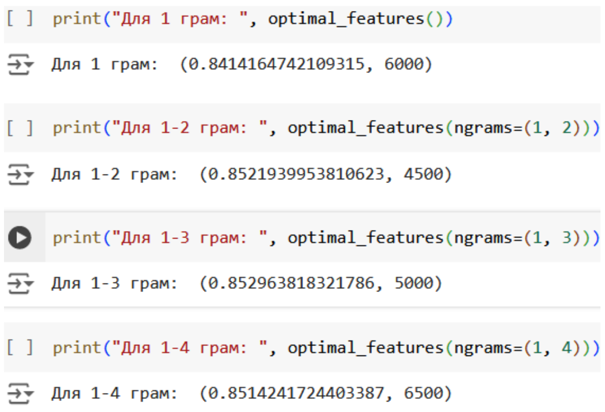
We will check: which vectors are better, Count or TF-IDF, the number of n-grams and the number of features.

To check all this, we will write the function optimal features and the auxiliary function accuracy_pipeline to it.



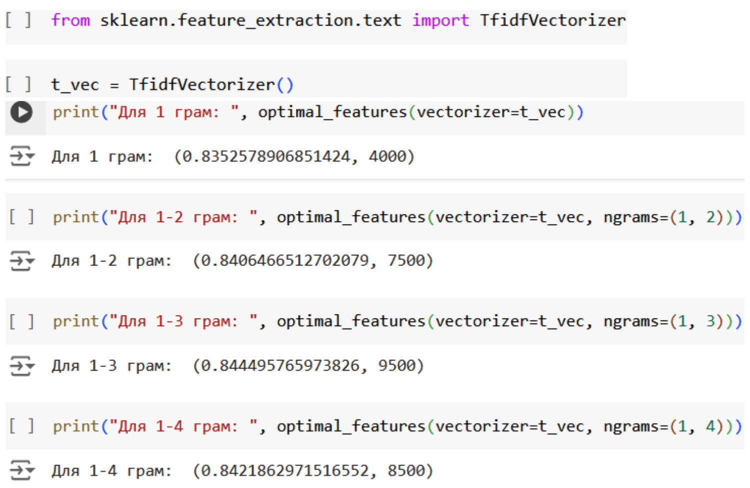
```
[ ] from sklearn.linear_model import LogisticRegression
    from sklearn.pipeline import Pipeline
[ ] LR = LogisticRegression()
    n_features = np.arange(1000,11501,500)    # [1000, 1500, 2000, 2500...]
    c_vec = CountVectorizer()
[ ] def accuracy_pipeline(pipeline, x_train, y_train, x_test, y_test):
        sentiment_fit = pipeline.fit(x_train, y_train)
        y_pred = sentiment_fit.predict(x_test)
        accuracy = accuracy_score(y_test, y_pred)

        return accuracy
[ ] def optimal_features(vectorizer=c_vec, n_features=n_features, ngrams=(1, 1), classifier=LR):

        max_accuracy = 0
        max_features = 0

        for n in n_features:
            vectorizer.set_params(max_features=n, ngram_range=ngrams)

            checker_pipeline = Pipeline([
                ('vectorizer', vectorizer),
                ('classifier', classifier)
            ])

            nfeature_accuracy = accuracy_pipeline(checker_pipeline, x_train, y_train, x_validation, y_validation)
            if nfeature_accuracy > max_accuracy:
                max_accuracy = nfeature_accuracy
                max_features = n

        return max_accuracy, max_features
```

First, we check the Count vectors:



```
print("Для 1 грам: ", optimal_features())
```
```
Для 1 грам:  (0.8414164742109315, 6000)
```
```
print("Для 1-2 грам: ", optimal_features(ngrams=(1, 2)))
```
```
Для 1-2 грам:  (0.8521939953810623, 4500)
```
```
print("Для 1-3 грам: ", optimal_features(ngrams=(1, 3)))
```
```
Для 1-3 грам:  (0.852963818321786, 5000)
```
```
print("Для 1-4 грам: ", optimal_features(ngrams=(1, 4)))
```
```
Для 1-4 грам:  (0.8514241724403387, 6500)
```

The best result is an accuracy of 0.8529 on 1-2-3 grams and several features of 5000. We check the same for TF-IDF vectors.



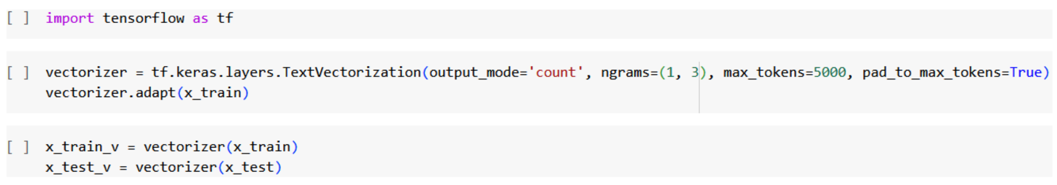
```
from sklearn.feature_extraction.text import TfidfVectorizer
```
```
t_vec = TfidfVectorizer()
print("Для 1 грам: ", optimal_features(vectorizer=t_vec))
```
```
Для 1 грам:  (0.8352578906851424, 4000)
```
```
print("Для 1-2 грам: ", optimal_features(vectorizer=t_vec, ngrams=(1, 2)))
```
```
Для 1-2 грам:  (0.8406466512702079, 7500)
```
```
print("Для 1-3 грам: ", optimal_features(vectorizer=t_vec, ngrams=(1, 3)))
```
```
Для 1-3 грам:  (0.844495765973826, 9500)
```
```
print("Для 1-4 грам: ", optimal_features(vectorizer=t_vec, ngrams=(1, 4)))
```
```
Для 1-4 грам:  (0.8421862971516552, 8500)
```

The best result is 0.8444 on 1-2-3 grams and the number of signs is 9500.

On these Count data, vectors provide greater accuracy and on a smaller number of features, so these vectors and optimal parameters will be used further.

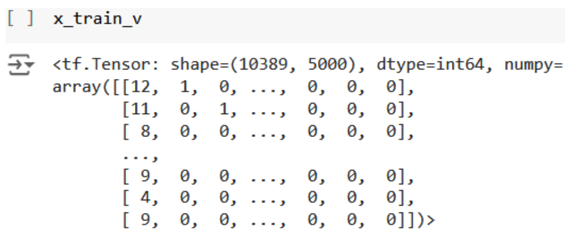
The next step will be to use neural networks for text classification.

Let's use neural networks to classify propaganda. I will start with a simple fully connected neural network, but before that, we need to bring the data into such a format that it can be given to the input of the neural network. We convert texts into vectors using the TextVectorization layer from the TensorFlow library:



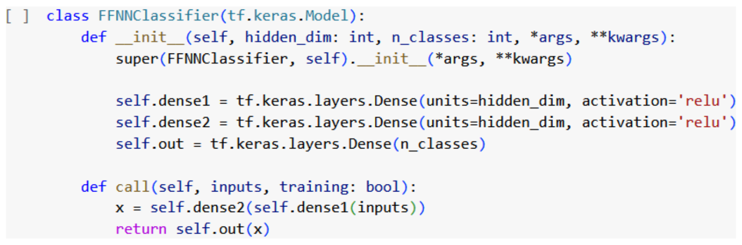
```
import tensorflow as tf
```
```
vectorizer = tf.keras.layers.TextVectorization(output_mode='count', ngrams=(1, 3), max_tokens=5000, pad_to_max_tokens=True)
vectorizer.adapt(x_train)
```
```
x_train_v = vectorizer(x_train)
x_test_v = vectorizer(x_test)
```

Now the texts look like this:



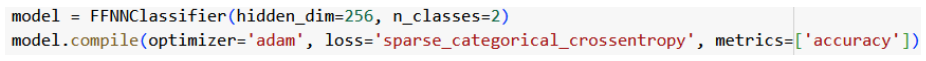
```
[ ]  x_train_v

<tf.Tensor: shape=(10389, 5000), dtype=int64, numpy=
array([[12,  1,  0, ...,  0,  0,  0],
       [11,  0,  1, ...,  0,  0,  0],
       [ 8,  0,  0, ...,  0,  0,  0],
       ...,
       [ 9,  0,  0, ...,  0,  0,  0],
       [ 4,  0,  0, ...,  0,  0,  0],
       [ 9,  0,  0, ...,  0,  0,  0]])>
```

Neural network structure: Input is fed to a Dense layer of 256 neurons, from it to the next Dense layer of 256 neurons, and then to the output Dense layer of 2 neurons, which classifies the input as propaganda or non-propaganda.



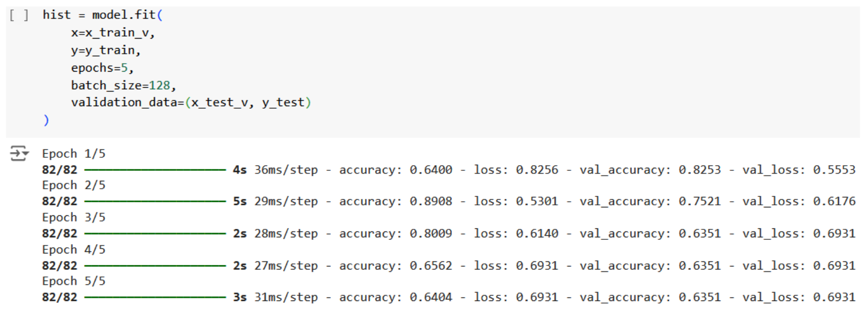
```
[ ]  class FFNNClassifier(tf.keras.Model):
         def __init__(self, hidden_dim: int, n_classes: int, *args, **kwargs):
             super(FFNNClassifier, self).__init__(*args, **kwargs)

             self.dense1 = tf.keras.layers.Dense(units=hidden_dim, activation='relu')
             self.dense2 = tf.keras.layers.Dense(units=hidden_dim, activation='relu')
             self.out = tf.keras.layers.Dense(n_classes)

         def call(self, inputs, training: bool):
             x = self.dense2(self.dense1(inputs))
             return self.out(x)
```

Neural network parameters: optimizer – Adam, loss function – sparse categorical cross-entropy, metrics – accuracy.



```
model = FFNNClassifier(hidden_dim=256, n_classes=2)
model.compile(optimizer='adam', loss='sparse_categorical_crossentropy', metrics=['accuracy'])
```

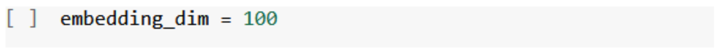
Training the network for 5 epochs:



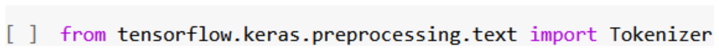
```
[ ]  hist = model.fit(
         x=x_train_v,
         y=y_train,
         epochs=5,
         batch_size=128,
         validation_data=(x_test_v, y_test)
     )

     Epoch 1/5
     82/82 ━━━━━━━━━━━━━━━━━━━━ 4s 36ms/step - accuracy: 0.6400 - loss: 0.8256 - val_accuracy: 0.8253 - val_loss: 0.5553
     Epoch 2/5
     82/82 ━━━━━━━━━━━━━━━━━━━━ 5s 29ms/step - accuracy: 0.8908 - loss: 0.5301 - val_accuracy: 0.7521 - val_loss: 0.6176
     Epoch 3/5
     82/82 ━━━━━━━━━━━━━━━━━━━━ 2s 28ms/step - accuracy: 0.8009 - loss: 0.6140 - val_accuracy: 0.6351 - val_loss: 0.6931
     Epoch 4/5
     82/82 ━━━━━━━━━━━━━━━━━━━━ 2s 27ms/step - accuracy: 0.6562 - loss: 0.6931 - val_accuracy: 0.6351 - val_loss: 0.6931
     Epoch 5/5
     82/82 ━━━━━━━━━━━━━━━━━━━━ 3s 31ms/step - accuracy: 0.6404 - loss: 0.6931 - val_accuracy: 0.6351 - val_loss: 0.6931
```

The best result was on the first epoch, where the validation accuracy val_accuracy = 0.8253. The result is a couple of percent worse than it was in logistic regression.

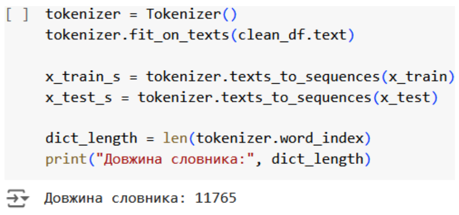
We will try to improve the result by using embeddings instead of ordinary vectors to represent texts. To do this, we divide the texts into tokens, then each token is represented as a vector. I will also slightly improve the structure of the neural network.



```
[ ]  embedding_dim = 100
```



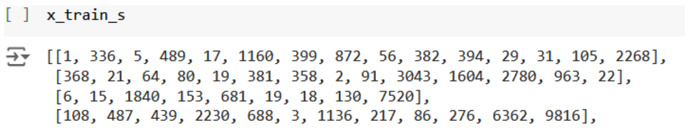
```
[ ]  from tensorflow.keras.preprocessing.text import Tokenizer
```



```
tokenizer = Tokenizer()
tokenizer.fit_on_texts(clean_df.text)

x_train_s = tokenizer.texts_to_sequences(x_train)
x_test_s = tokenizer.texts_to_sequences(x_test)

dict_length = len(tokenizer.word_index)
print("Довжина словника:", dict_length)
```

```
Довжина словника: 11765
```

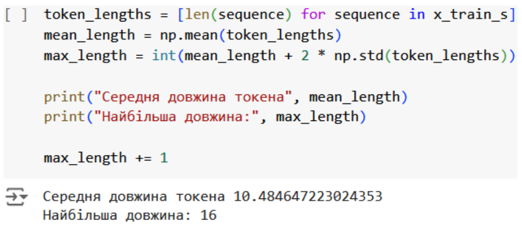
Now the input looks like this:



```
x_train_s
```

```
[[1, 336, 5, 489, 17, 1160, 399, 872, 56, 382, 394, 29, 31, 105, 2268],
 [368, 21, 64, 80, 19, 381, 358, 2, 91, 3043, 1604, 2780, 963, 22],
 [6, 15, 1840, 153, 681, 19, 18, 130, 7520],
 [108, 487, 439, 2230, 688, 3, 1136, 217, 86, 276, 6362, 9816],
```

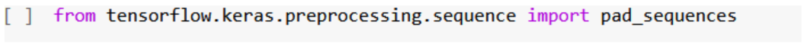
We determine the average and maximum length of tokens:



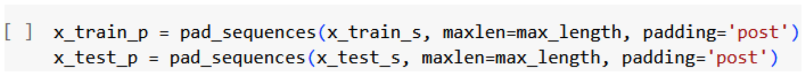
```
token_lengths = [len(sequence) for sequence in x_train_s]
mean_length = np.mean(token_lengths)
max_length = int(mean_length + 2 * np.std(token_lengths))

print("Середня довжина токена", mean_length)
print("Найбільша довжина:", max_length)

max_length += 1
```

```
Середня довжина токена 10.484647223024353
Найбільша довжина: 16
```

We add padding to the tokens so that they are all the same size:

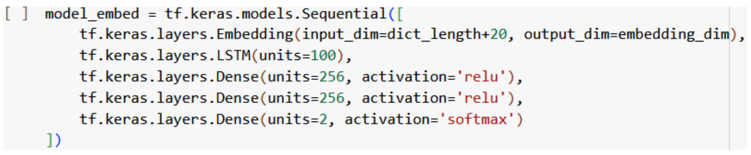


```
from tensorflow.keras.preprocessing.sequence import pad_sequences
```



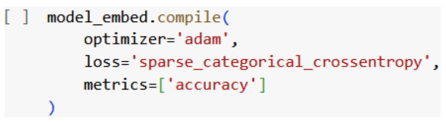
```
x_train_p = pad_sequences(x_train_s, maxlen=max_length, padding='post')
x_test_p = pad_sequences(x_test_s, maxlen=max_length, padding='post')
```

New neural network structure: embedding layer, LSTM (Long short-term memory) layer, followed by the same Dense layers with 2 output classes.

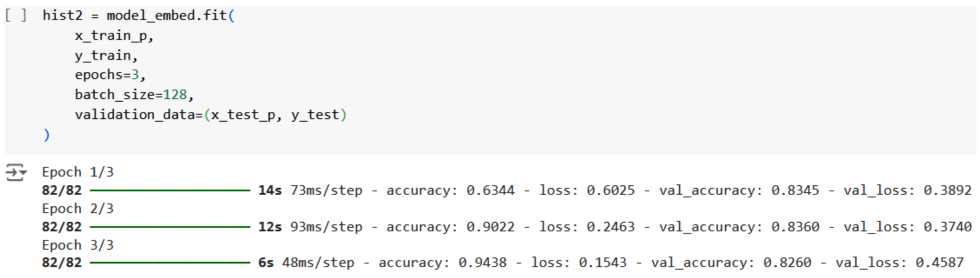


```
model_embed = tf.keras.models.Sequential([
    tf.keras.layers.Embedding(input_dim=dict_length+20, output_dim=embedding_dim),
    tf.keras.layers.LSTM(units=100),
    tf.keras.layers.Dense(units=256, activation='relu'),
    tf.keras.layers.Dense(units=256, activation='relu'),
    tf.keras.layers.Dense(units=2, activation='softmax')
])
```

The network training parameters remain the same.



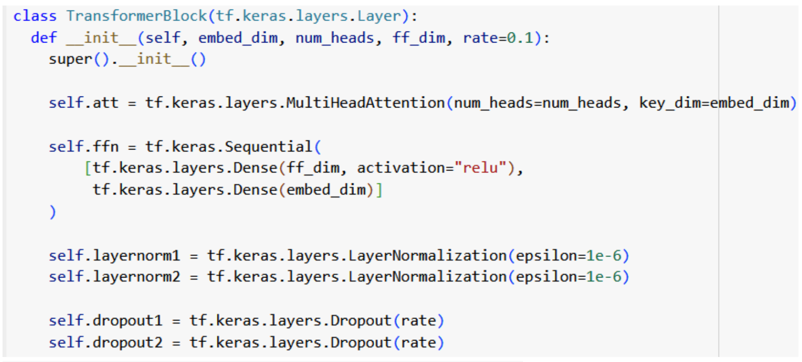
```
model_embed.compile(
    optimizer='adam',
    loss='sparse_categorical_crossentropy',
    metrics=['accuracy']
)
```



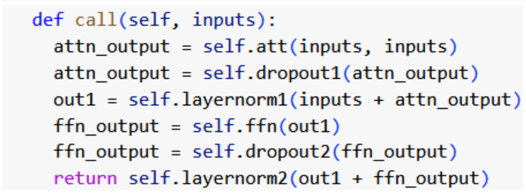
```
[ ] hist2 = model_embed.fit(
        x_train_p,
        y_train,
        epochs=3,
        batch_size=128,
        validation_data=(x_test_p, y_test)
    )

    Epoch 1/3
    82/82 ━━━━━━━━━━━━━━━━━━━━ 14s 73ms/step - accuracy: 0.6344 - loss: 0.6025 - val_accuracy: 0.8345 - val_loss: 0.3892
    Epoch 2/3
    82/82 ━━━━━━━━━━━━━━━━━━━━ 12s 93ms/step - accuracy: 0.9022 - loss: 0.2463 - val_accuracy: 0.8360 - val_loss: 0.3740
    Epoch 3/3
    82/82 ━━━━━━━━━━━━━━━━━━━━ 6s 48ms/step - accuracy: 0.9438 - loss: 0.1543 - val_accuracy: 0.8260 - val_loss: 0.4587
```

Of the three epochs, the second with val_accuracy = 0.8360 had the highest accuracy. Better than the previous neural network, but still not better than logistic regression.

Let's try to use a transformer. To do this, we will build a transformer block.

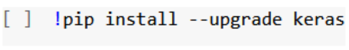


```
class TransformerBlock(tf.keras.layers.Layer):
  def __init__(self, embed_dim, num_heads, ff_dim, rate=0.1):
    super().__init__()

    self.att = tf.keras.layers.MultiHeadAttention(num_heads=num_heads, key_dim=embed_dim)

    self.ffn = tf.keras.Sequential(
        [tf.keras.layers.Dense(ff_dim, activation="relu"),
         tf.keras.layers.Dense(embed_dim)]
    )

    self.layernorm1 = tf.keras.layers.LayerNormalization(epsilon=1e-6)
    self.layernorm2 = tf.keras.layers.LayerNormalization(epsilon=1e-6)

    self.dropout1 = tf.keras.layers.Dropout(rate)
    self.dropout2 = tf.keras.layers.Dropout(rate)
```

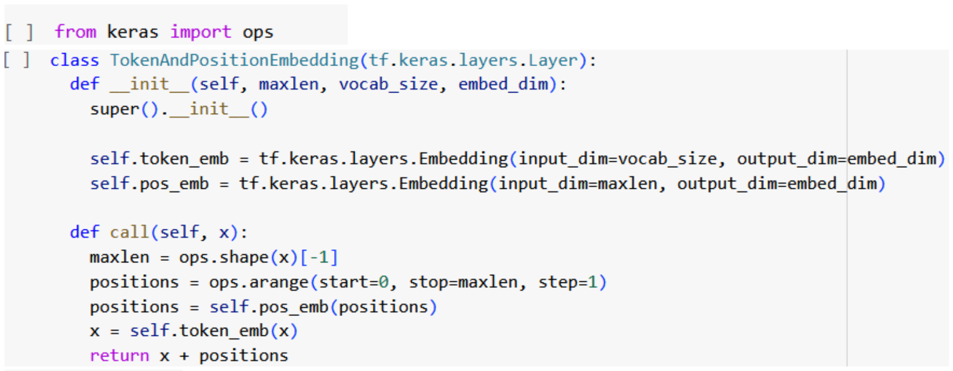


```
  def call(self, inputs):
    attn_output = self.att(inputs, inputs)
    attn_output = self.dropout1(attn_output)
    out1 = self.layernorm1(inputs + attn_output)
    ffn_output = self.ffn(out1)
    ffn_output = self.dropout2(ffn_output)
    return self.layernorm2(out1 + ffn_output)
```

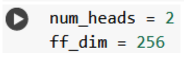
We will also use embeddings here, and for this, we will build an embedding layer.



```
[ ] !pip install --upgrade keras
```

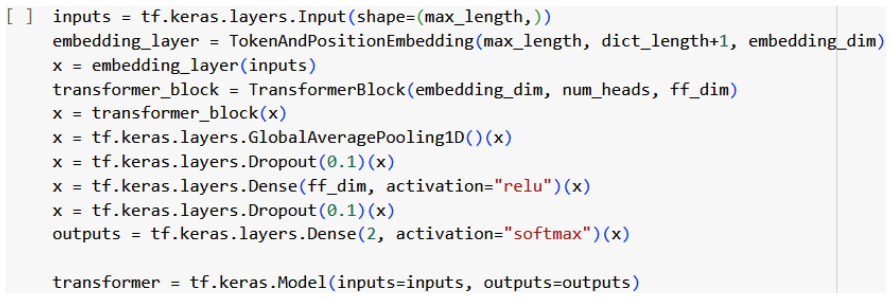


```
[ ] from keras import ops
[ ] class TokenAndPositionEmbedding(tf.keras.layers.Layer):
      def __init__(self, maxlen, vocab_size, embed_dim):
        super().__init__()

        self.token_emb = tf.keras.layers.Embedding(input_dim=vocab_size, output_dim=embed_dim)
        self.pos_emb = tf.keras.layers.Embedding(input_dim=maxlen, output_dim=embed_dim)

      def call(self, x):
        maxlen = ops.shape(x)[-1]
        positions = ops.arange(start=0, stop=maxlen, step=1)
        positions = self.pos_emb(positions)
        x = self.token_emb(x)
        return x + positions
```



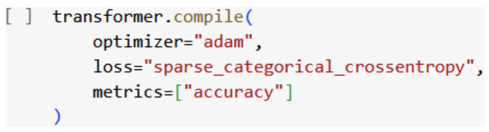
```
num_heads = 2
ff_dim = 256
```

The structure of the model itself: input layer, embedding layer, transformer block, pooling layer, dropout layer, fully connected layer, dropout again, output fully connected layer for 2 neurons that classifies.



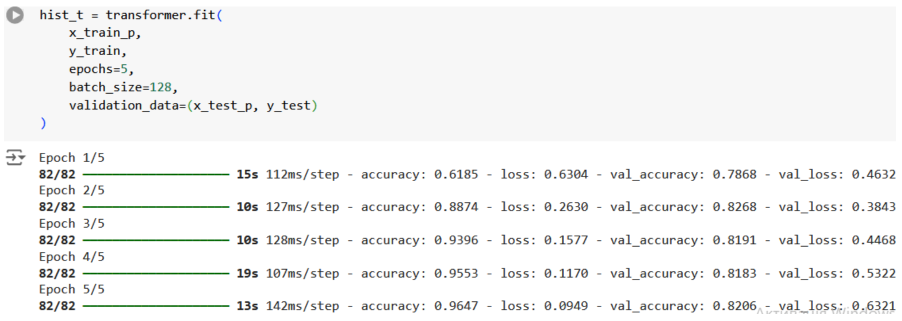
```
[ ]  inputs = tf.keras.layers.Input(shape=(max_length,))
     embedding_layer = TokenAndPositionEmbedding(max_length, dict_length+1, embedding_dim)
     x = embedding_layer(inputs)
     transformer_block = TransformerBlock(embedding_dim, num_heads, ff_dim)
     x = transformer_block(x)
     x = tf.keras.layers.GlobalAveragePooling1D()(x)
     x = tf.keras.layers.Dropout(0.1)(x)
     x = tf.keras.layers.Dense(ff_dim, activation="relu")(x)
     x = tf.keras.layers.Dropout(0.1)(x)
     outputs = tf.keras.layers.Dense(2, activation="softmax")(x)

     transformer = tf.keras.Model(inputs=inputs, outputs=outputs)
```

Learning parameters are unchanged:



```
[ ]  transformer.compile(
         optimizer="adam",
         loss="sparse_categorical_crossentropy",
         metrics=["accuracy"]
     )
```

Model training on 5 epochs:



```
hist_t = transformer.fit(
    x_train_p,
    y_train,
    epochs=5,
    batch_size=128,
    validation_data=(x_test_p, y_test)
)

Epoch 1/5
82/82 ━━━━━━━━━━━━━━━━━━━━ 15s 112ms/step - accuracy: 0.6185 - loss: 0.6304 - val_accuracy: 0.7868 - val_loss: 0.4632
Epoch 2/5
82/82 ━━━━━━━━━━━━━━━━━━━━ 10s 127ms/step - accuracy: 0.8874 - loss: 0.2630 - val_accuracy: 0.8268 - val_loss: 0.3843
Epoch 3/5
82/82 ━━━━━━━━━━━━━━━━━━━━ 10s 128ms/step - accuracy: 0.9396 - loss: 0.1577 - val_accuracy: 0.8191 - val_loss: 0.4468
Epoch 4/5
82/82 ━━━━━━━━━━━━━━━━━━━━ 19s 107ms/step - accuracy: 0.9553 - loss: 0.1170 - val_accuracy: 0.8183 - val_loss: 0.5322
Epoch 5/5
82/82 ━━━━━━━━━━━━━━━━━━━━ 13s 142ms/step - accuracy: 0.9647 - loss: 0.0949 - val_accuracy: 0.8206 - val_loss: 0.6321
```

The best result was on the second epoch with val_accuracy = 0.8268.

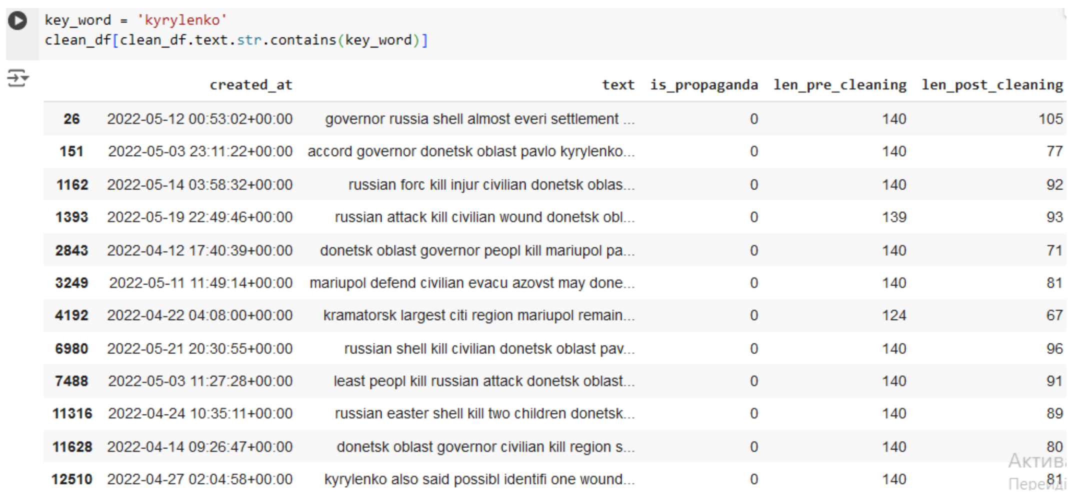
For some reason, neural networks cannot outperform logistic regression. In our opinion, it depends on the data, and on such a small dataset of 13 thousand records, the neural network cannot obtain sufficient generalizing properties.

The next step is to find the most suitable way to measure the distance between texts.

The task is to find chains of propagation of propaganda between groups. To do this, we will search for similar texts and, using metadata such as the date and time of the message, we will investigate how these similar messages were distributed.

But before that, it is necessary to choose a method of measuring the similarity of messages. We plan to test lexical and semantic methods.

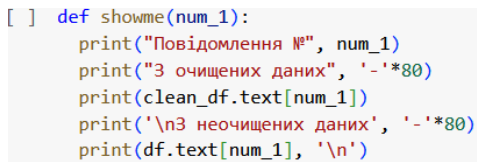
First, we manually find several messages of varying degrees of similarity to compare the methods. To do this, I will display all messages containing the keyword, the surname "kyrilenko".



```
key_word = 'kyrylenko'
clean_df[clean_df.text.str.contains(key_word)]
```

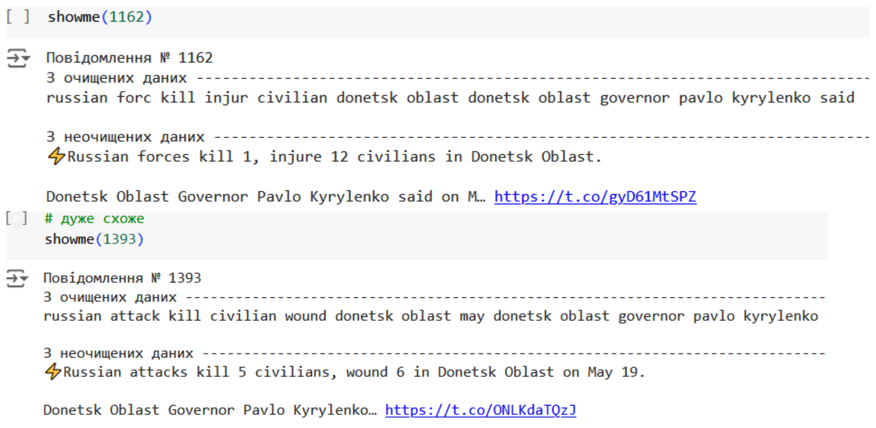
	created_at	text	is_propaganda	len_pre_cleaning	len_post_cleaning
26	2022-05-12 00:53:02+00:00	governor russia shell almost everi settlement ...	0	140	105
151	2022-05-03 23:11:22+00:00	accord governor donetsk oblast pavlo kyrylenko...	0	140	77
1162	2022-05-14 03:58:32+00:00	russian forc kill injur civilian donetsk oblas...	0	140	92
1393	2022-05-19 22:49:46+00:00	russian attack kill civilian wound donetsk obl...	0	139	93
2843	2022-04-12 17:40:39+00:00	donetsk oblast governor peopl kill mariupol pa...	0	140	71
3249	2022-05-11 11:49:14+00:00	mariupol defend civilian evacu azovst may done...	0	140	81
4192	2022-04-22 04:08:00+00:00	kramatorsk largest citi region mariupol remain...	0	124	67
6980	2022-05-21 20:30:55+00:00	russian shell kill civilian donetsk oblast pav...	0	140	96
7488	2022-05-03 11:27:28+00:00	least peopl kill russian attack donetsk oblast...	0	140	91
11316	2022-04-24 10:35:11+00:00	russian easter shell kill two children donetsk...	0	140	89
11628	2022-04-14 09:26:47+00:00	donetsk oblast governor civilian kill region s...	0	140	80
12510	2022-04-27 02:04:58+00:00	kyrylenko also said possibl identifi one wound...	0	140	81

To see what specific messages looked like before and after processing the text, we will write the showme function, in which we will enter the indexes of the messages we are interested in.



```
def showme(num_1):
    print("Повідомлення №", num_1)
    print("З очищених даних", '-'*80)
    print(clean_df.text[num_1])
    print('\nЗ неочищених даних', '-'*80)
    print(df.text[num_1], '\n')
```

Several messages were selected using this function:
1162 - it will be the main one, with which I will compare the others
1393 - a message that is very similar to the main one
11628 is slightly smaller but still similar to the main one
12510 – contains several common words with the main one
6666 - not at all similar to the main one (as far as the nature of the dataset allows)

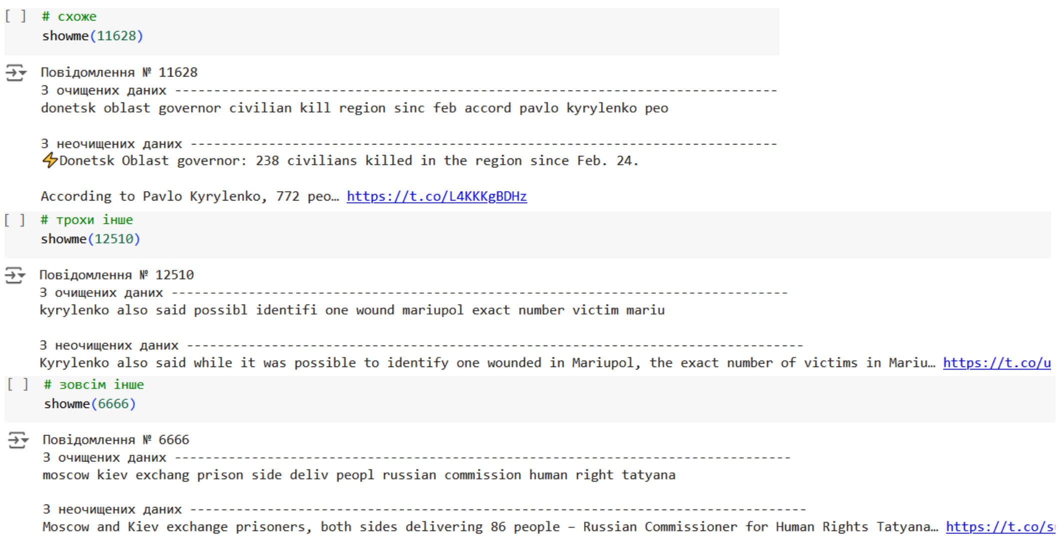


```
showme(1162)
```

```
Повідомлення № 1162
З очищених даних --------------------------------------------------------------------------------
russian forc kill injur civilian donetsk oblast donetsk oblast governor pavlo kyrylenko said

З неочищених даних --------------------------------------------------------------------------------
⚡Russian forces kill 1, injure 12 civilians in Donetsk Oblast.

Donetsk Oblast Governor Pavlo Kyrylenko said on M… https://t.co/gyD61MtSPZ
```

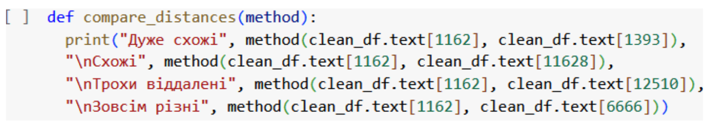
```
# дуже схоже
showme(1393)
```

```
Повідомлення № 1393
З очищених даних --------------------------------------------------------------------------------
russian attack kill civilian wound donetsk oblast may donetsk oblast governor pavlo kyrylenko

З неочищених даних --------------------------------------------------------------------------------
⚡Russian attacks kill 5 civilians, wound 6 in Donetsk Oblast on May 19.

Donetsk Oblast Governor Pavlo Kyrylenko… https://t.co/ONLKdaTQzJ
```



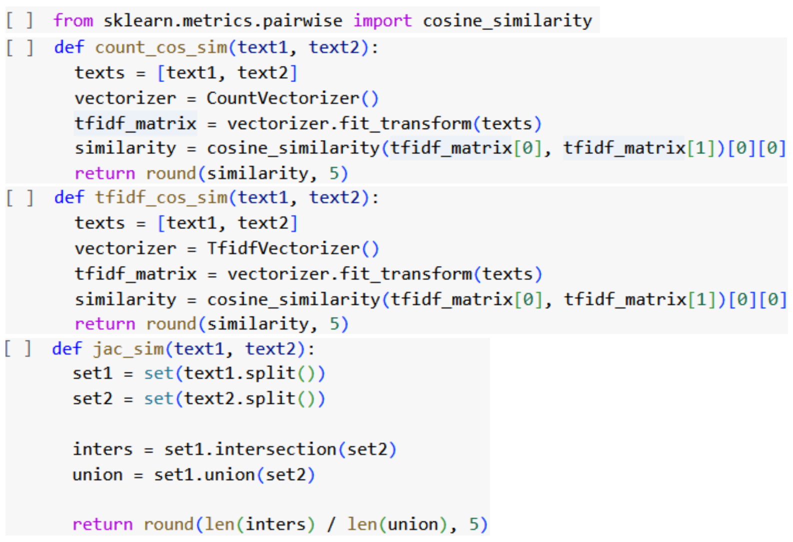
```
[ ] # схоже
    showme(11628)

    Повідомлення № 11628
    З очищених даних --------------------------------------------------------------------------------
    donetsk oblast governor civilian kill region sinc feb accord pavlo kyrylenko peo

    З неочищених даних ------------------------------------------------------------------------------
    ⚡Donetsk Oblast governor: 238 civilians killed in the region since Feb. 24.

    According to Pavlo Kyrylenko, 772 peo… https://t.co/L4KKKgBDHz
[ ] # трохи інше
    showme(12510)

    Повідомлення № 12510
    З очищених даних --------------------------------------------------------------------------------
    kyrylenko also said possibl identifi one wound mariupol exact number victim mariu

    З неочищених даних ------------------------------------------------------------------------------
    Kyrylenko also said while it was possible to identify one wounded in Mariupol, the exact number of victims in Mariu… https://t.co/u
[ ] # зовсім інше
    showme(6666)

    Повідомлення № 6666
    З очищених даних --------------------------------------------------------------------------------
    moscow kiev exchang prison side deliv peopl russian commission human right tatyana

    З неочищених даних ------------------------------------------------------------------------------
    Moscow and Kiev exchange prisoners, both sides delivering 86 people – Russian Commissioner for Human Rights Tatyana… https://t.co/s
```

Now let's write the compare_distances function for a quick comparison of text similarity measurement methods.

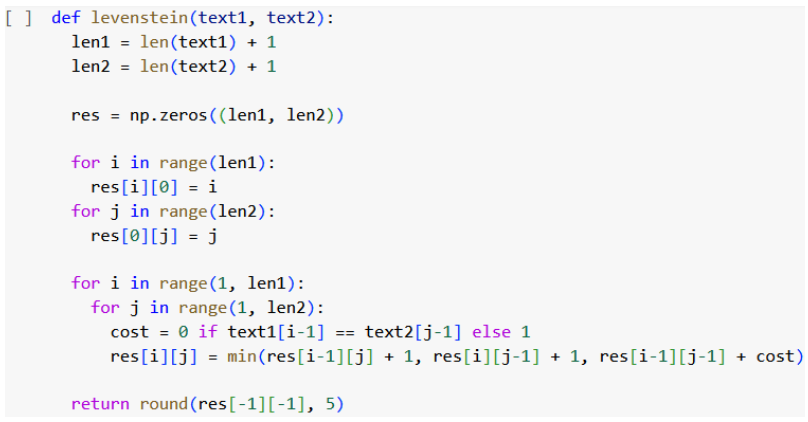


```
[ ] def compare_distances(method):
      print("Дуже схожі", method(clean_df.text[1162], clean_df.text[1393]),
      "\nСхожі", method(clean_df.text[1162], clean_df.text[11628]),
      "\nТрохи віддалені", method(clean_df.text[1162], clean_df.text[12510]),
      "\nЗовсім різні", method(clean_df.text[1162], clean_df.text[6666]))
```

Next, we will write functions for measuring cosine similarity (on Count and TF-IDF vectors), Jaccard similarity and Levenshtein distance. After that, we will compare their results.

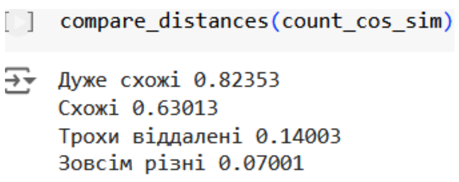


```
[ ] from sklearn.metrics.pairwise import cosine_similarity
[ ] def count_cos_sim(text1, text2):
      texts = [text1, text2]
      vectorizer = CountVectorizer()
      tfidf_matrix = vectorizer.fit_transform(texts)
      similarity = cosine_similarity(tfidf_matrix[0], tfidf_matrix[1])[0][0]
      return round(similarity, 5)
[ ] def tfidf_cos_sim(text1, text2):
      texts = [text1, text2]
      vectorizer = TfidfVectorizer()
      tfidf_matrix = vectorizer.fit_transform(texts)
      similarity = cosine_similarity(tfidf_matrix[0], tfidf_matrix[1])[0][0]
      return round(similarity, 5)
[ ] def jac_sim(text1, text2):
      set1 = set(text1.split())
      set2 = set(text2.split())

      inters = set1.intersection(set2)
      union = set1.union(set2)

      return round(len(inters) / len(union), 5)
```

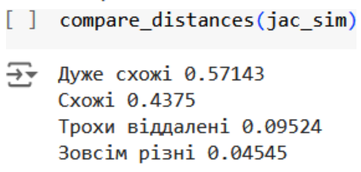


```
[ ] def levenstein(text1, text2):
      len1 = len(text1) + 1
      len2 = len(text2) + 1

      res = np.zeros((len1, len2))

      for i in range(len1):
        res[i][0] = i
      for j in range(len2):
        res[0][j] = j

      for i in range(1, len1):
        for j in range(1, len2):
          cost = 0 if text1[i-1] == text2[j-1] else 1
          res[i][j] = min(res[i-1][j] + 1, res[i][j-1] + 1, res[i-1][j-1] + cost)

      return round(res[-1][-1], 5)
```

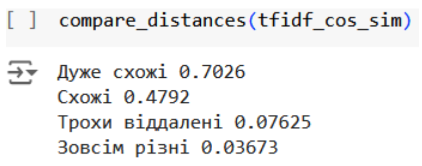
The results:



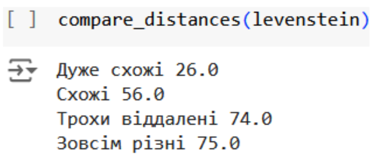
```
[ ] compare_distances(count_cos_sim)

Дуже схожі 0.82353
Схожі 0.63013
Трохи віддалені 0.14003
Зовсім різні 0.07001
```



```
[ ] compare_distances(jac_sim)

Дуже схожі 0.57143
Схожі 0.4375
Трохи віддалені 0.09524
Зовсім різні 0.04545
```



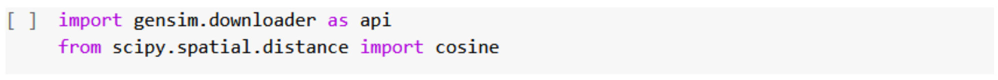
```
[ ] compare_distances(tfidf_cos_sim)

Дуже схожі 0.7026
Схожі 0.4792
Трохи віддалені 0.07625
Зовсім різні 0.03673
```



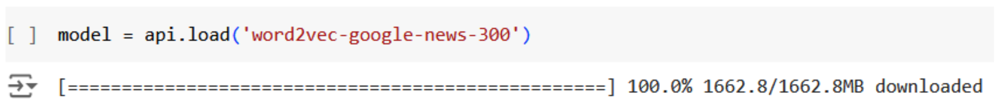
```
[ ] compare_distances(levenstein)

Дуже схожі 26.0
Схожі 56.0
Трохи віддалені 74.0
Зовсім різні 75.0
```

The closest to what we want to get is shown by cosine similarity on count vectors, on TF-IDF vectors the value is closer to 0, in Jaccard similarity the value is closer to 0.5, and as for the Levenshtein distance, we realized that it is not very suitable for this task at all.

Now let's try to take into account the semantic content. To do this, download word2vec embeddings from Google News.

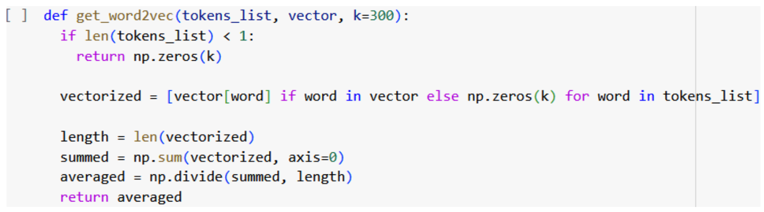


```
[ ] import gensim.downloader as api
    from scipy.spatial.distance import cosine
```

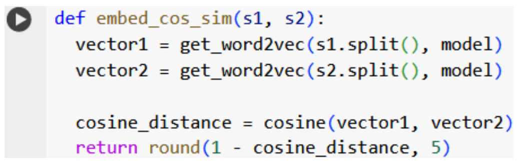


```
[ ] model = api.load('word2vec-google-news-300')

[==================================================] 100.0% 1662.8/1662.8MB downloaded
```

Let's write a function for cosine similarity on embeddings embed_cos_sim and an auxiliary function get_word2vec.

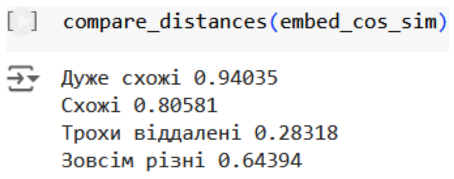


```
[ ] def get_word2vec(tokens_list, vector, k=300):
      if len(tokens_list) < 1:
        return np.zeros(k)

      vectorized = [vector[word] if word in vector else np.zeros(k) for word in tokens_list]

      length = len(vectorized)
      summed = np.sum(vectorized, axis=0)
      averaged = np.divide(summed, length)
      return averaged
```



```python
def embed_cos_sim(s1, s2):
  vector1 = get_word2vec(s1.split(), model)
  vector2 = get_word2vec(s2.split(), model)

  cosine_distance = cosine(vector1, vector2)
  return round(1 - cosine_distance, 5)
```

Result:



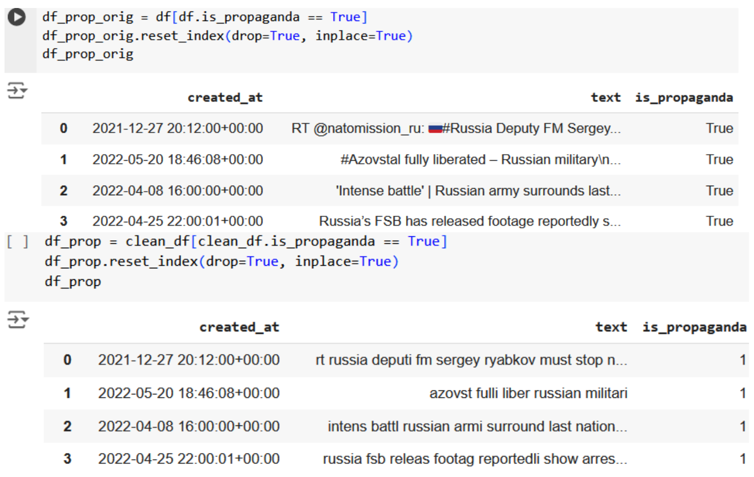
```
[ ] compare_distances(embed_cos_sim)
```

```
Дуже схожі 0.94035
Схожі 0.80581
Трохи віддалені 0.28318
Зовсім різні 0.64394
```

This is exactly what we wanted to get, but for some reason, completely different texts have more similarities than slightly different ones. But it already depends on how these embeddings were trained.

The next step is to search for propaganda networks by examining messages with similar texts.

Using the selected method of measuring the distance between texts (cosine similarity on word2vec-google-news-300 embeddings), we want to find similar messages among the entire dataset. To do this, we create 2 dataframes containing only propaganda messages (since we are investigating the spread of propaganda). The first will contain the texts in their original form, and the second in a cleaned form.



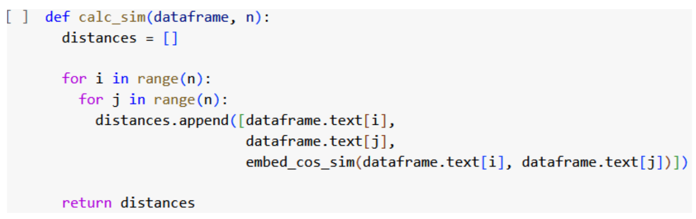
```python
df_prop_orig = df[df.is_propaganda == True]
df_prop_orig.reset_index(drop=True, inplace=True)
df_prop_orig
```

	created_at	text	is_propaganda
0	2021-12-27 20:12:00+00:00	RT @natomission_ru: 🇷🇺#Russia Deputy FM Sergey...	True
1	2022-05-20 18:46:08+00:00	#Azovstal fully liberated – Russian military\n...	True
2	2022-04-08 16:00:00+00:00	'Intense battle' | Russian army surrounds last...	True
3	2022-04-25 22:00:01+00:00	Russia's FSB has released footage reportedly s...	True

```python
[ ] df_prop = clean_df[clean_df.is_propaganda == True]
df_prop.reset_index(drop=True, inplace=True)
df_prop
```

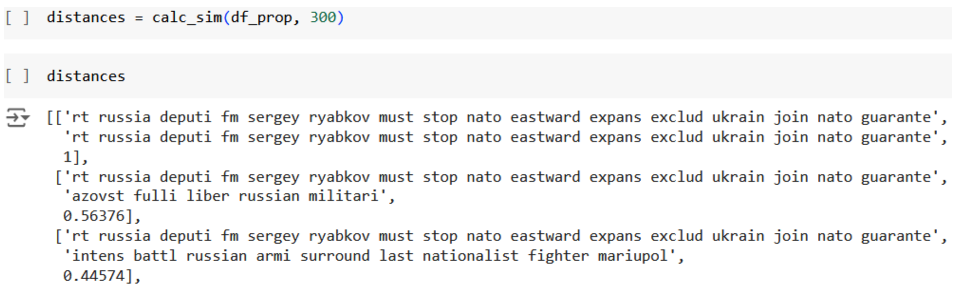
	created_at	text	is_propaganda
0	2021-12-27 20:12:00+00:00	rt russia deputi fm sergey ryabkov must stop n...	1
1	2022-05-20 18:46:08+00:00	azovst fulli liber russian militari	1
2	2022-04-08 16:00:00+00:00	intens battl russian armi surround last nation...	1
3	2022-04-25 22:00:01+00:00	russia fsb releas footag reportedli show arres...	1

Let's write a small function to compare each message with each other.



```python
[ ] def calc_sim(dataframe, n):
      distances = []

      for i in range(n):
        for j in range(n):
          distances.append([dataframe.text[i],
                            dataframe.text[j],
                            embed_cos_sim(dataframe.text[i], dataframe.text[j])])

      return distances
```

Did a test run with 100 x 100 messages, measured the time it took and realized we didn't have the computing power to compare all 6500 x 6500 messages. However, the research is not finished yet, so let's take a small fraction of 300 x 300 messages and work with them.

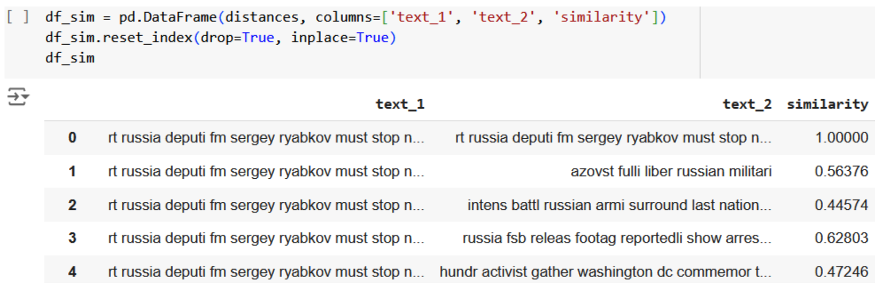


```
[ ] distances = calc_sim(df_prop, 300)
```

```
[ ] distances
```

```
[['rt russia deputi fm sergey ryabkov must stop nato eastward expans exclud ukrain join nato guarante',
  'rt russia deputi fm sergey ryabkov must stop nato eastward expans exclud ukrain join nato guarante',
  1],
 ['rt russia deputi fm sergey ryabkov must stop nato eastward expans exclud ukrain join nato guarante',
  'azovst fulli liber russian militari',
  0.56376],
 ['rt russia deputi fm sergey ryabkov must stop nato eastward expans exclud ukrain join nato guarante',
  'intens battl russian armi surround last nationalist fighter mariupol',
  0.44574],
```

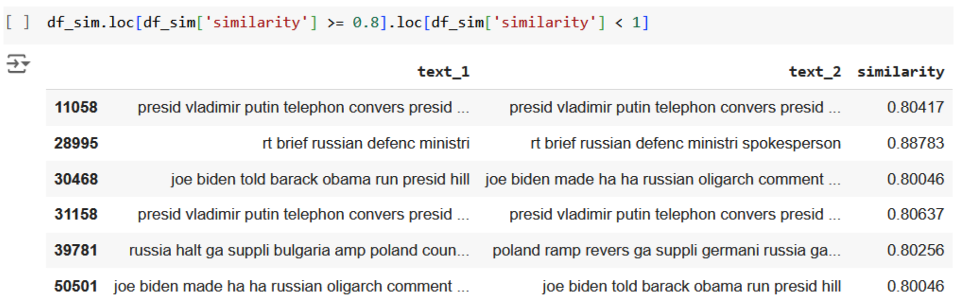
For better clarity, we create a data frame from the result.



```
[ ] df_sim = pd.DataFrame(distances, columns=['text_1', 'text_2', 'similarity'])
    df_sim.reset_index(drop=True, inplace=True)
    df_sim
```

	text_1	text_2	similarity
0	rt russia deputi fm sergey ryabkov must stop n...	rt russia deputi fm sergey ryabkov must stop n...	1.00000
1	rt russia deputi fm sergey ryabkov must stop n...	azovst fulli liber russian militari	0.56376
2	rt russia deputi fm sergey ryabkov must stop n...	intens battl russian armi surround last nation...	0.44574
3	rt russia deputi fm sergey ryabkov must stop n...	russia fsb releas footag reportedli show arres...	0.62803
4	rt russia deputi fm sergey ryabkov must stop n...	hundr activist gather washington dc commemor t...	0.47246

If you represent the result of the comparison as a matrix, then there will always be ones on the main diagonal, since there the message is compared with itself. Therefore, we select from the results only those that are less than one, and at the same time those that are greater than 0.8, that is, very similar messages that are potentially overwritten.

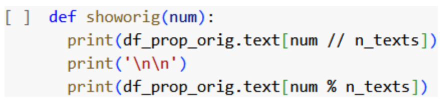


```
[ ] df_sim.loc[df_sim['similarity'] >= 0.8].loc[df_sim['similarity'] < 1]
```

	text_1	text_2	similarity
11058	presid vladimir putin telephon convers presid ...	presid vladimir putin telephon convers presid ...	0.80417
28995	rt brief russian defenc ministri	rt brief russian defenc ministri spokesperson	0.88783
30468	joe biden told barack obama run presid hill	joe biden made ha ha russian oligarch comment ...	0.80046
31158	presid vladimir putin telephon convers presid ...	presid vladimir putin telephon convers presid ...	0.80637
39781	russia halt ga suppli bulgaria amp poland coun...	poland ramp revers ga suppli germani russia ga...	0.80256
50501	joe biden made ha ha russian oligarch comment ...	joe biden told barack obama run presid hill	0.80046

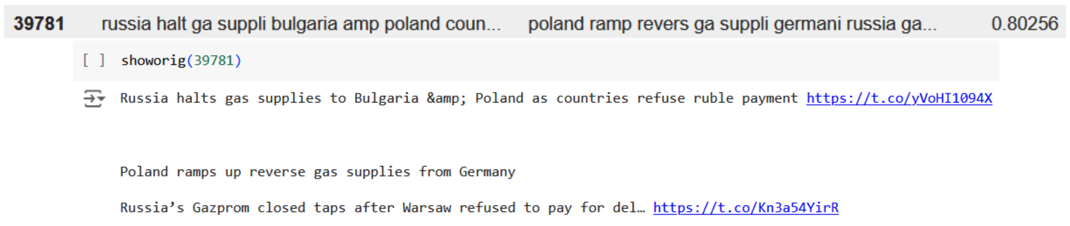
To conclude whether the messages are similar, we want to look at their original appearance. But after comparing 300 x 300 in the resulting dataframe, all the indexes went astray. However, knowing how these indices were formed allows us to return to the original indices. For example, if the result is a message with index 302, it is clear that this is the second message compared to the third, because the first 300 records (0 - 299) are comparisons of the first message with all others, and starting from 300 (300 - 599) there are comparisons second message with all others.

In this way, to get the index of the first message, you need to perform a special division "//" operation on the index from the result, which returns only the whole part (floor division), and for the second index, perform a "%" operation on the remainder of the division (modulus).



```
[ ] def showorig(num):
      print(df_prop_orig.text[num // n_texts])
      print('\n\n')
      print(df_prop_orig.text[num % n_texts])
```

We are interested in this line, so we will generate original views for it.



```
39781   russia halt ga suppli bulgaria amp poland coun...   poland ramp revers ga suppli germani russia ga...   0.80256
```

```
[ ] showorig(39781)
```

```
Russia halts gas supplies to Bulgaria &amp; Poland as countries refuse ruble payment https://t.co/yVoHI1094X


Poland ramps up reverse gas supplies from Germany

Russia’s Gazprom closed taps after Warsaw refused to pay for del… https://t.co/Kn3a54YirR
```

In these messages, some links were followed and made sure that the messages could be called the same.

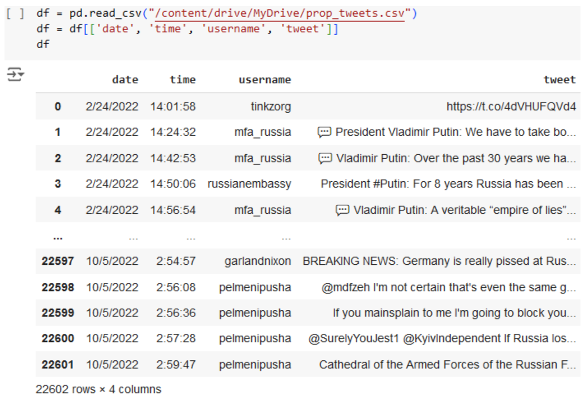
Link:

https://x.com/RT_com/status/1519692929065009155

https://x.com/i/web/status/1519292781226868742

They both say that russia is stopping gas supplies to Poland because it does not want to pay for gas in rubles.

This is a success in terms of finding similar messages, but both of these messages came from the same group, a day apart. We want to find connections between different groups. To do this, I download a second dataset containing message metadata such as date, time, and group, and in the process of downloading, I select only those columns that interest us.

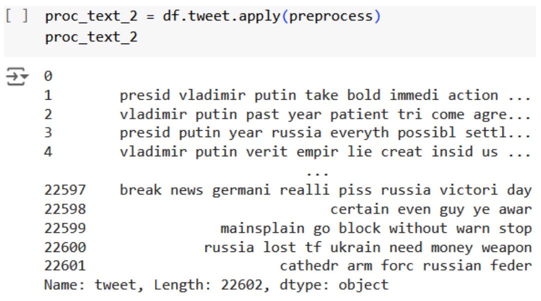


```
[ ] df = pd.read_csv("/content/drive/MyDrive/prop_tweets.csv")
    df = df[['date', 'time', 'username', 'tweet']]
    df
```

	date	time	username	tweet
0	2/24/2022	14:01:58	tinkzorg	https://t.co/4dVHUFQVd4
1	2/24/2022	14:24:32	mfa_russia	💬 President Vladimir Putin: We have to take bo...
2	2/24/2022	14:42:53	mfa_russia	💬 Vladimir Putin: Over the past 30 years we ha...
3	2/24/2022	14:50:06	russianembassy	President #Putin: For 8 years Russia has been ...
4	2/24/2022	14:56:54	mfa_russia	💬 Vladimir Putin: A veritable “empire of lies”...
...	...	...	...	...
22597	10/5/2022	2:54:57	garlandnixon	BREAKING NEWS: Germany is really pissed at Rus...
22598	10/5/2022	2:56:08	pelmenipusha	@mdfzeh I'm not certain that's even the same g...
22599	10/5/2022	2:56:36	pelmenipusha	If you mainsplain to me I'm going to block you...
22600	10/5/2022	2:57:28	pelmenipusha	@SurelyYouJest1 @KyivIndependent If Russia los...
22601	10/5/2022	2:59:47	pelmenipusha	Cathedral of the Armed Forces of the Russian F...

22602 rows × 4 columns

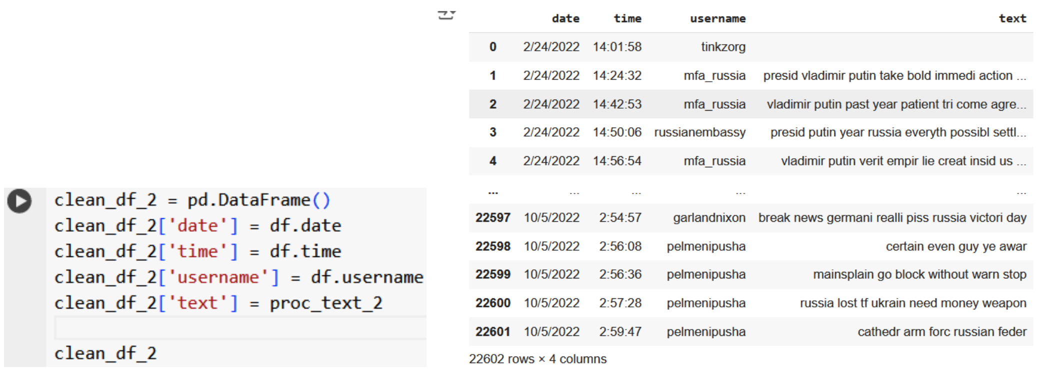
We use the old function for text processing:



```
[ ] proc_text_2 = df.tweet.apply(preprocess)
    proc_text_2
```

```
0
1        presid vladimir putin take bold immedi action ...
2        vladimir putin past year patient tri come agre...
3        presid putin year russia everyth possibl settl...
4        vladimir putin verit empir lie creat insid us ...
                                ...
22597    break news germani realli piss russia victori day
22598                           certain even guy ye awar
22599                  mainsplain go block without warn stop
22600                russia lost tf ukrain need money weapon
22601                          cathedr arm forc russian feder
Name: tweet, Length: 22602, dtype: object
```

We create a dataframe for cleaned texts:

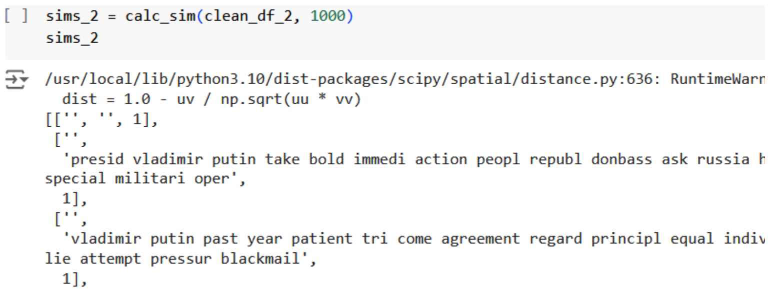


```
clean_df_2 = pd.DataFrame()
clean_df_2['date'] = df.date
clean_df_2['time'] = df.time
clean_df_2['username'] = df.username
clean_df_2['text'] = proc_text_2

clean_df_2
```

	date	time	username	text
0	2/24/2022	14:01:58	tinkzorg	
1	2/24/2022	14:24:32	mfa_russia	presid vladimir putin take bold immedi action ...
2	2/24/2022	14:42:53	mfa_russia	vladimir putin past year patient tri come agre...
3	2/24/2022	14:50:06	russianembassy	presid putin year russia everyth possibl settl...
4	2/24/2022	14:56:54	mfa_russia	vladimir putin verit empir lie creat insid us ...
...	...	...	...	...
22597	10/5/2022	2:54:57	garlandnixon	break news germani realli piss russia victori day
22598	10/5/2022	2:56:08	pelmenipusha	certain even guy ye awar
22599	10/5/2022	2:56:36	pelmenipusha	mainsplain go block without warn stop
22600	10/5/2022	2:57:28	pelmenipusha	russia lost tf ukrain need money weapon
22601	10/5/2022	2:59:47	pelmenipusha	cathedr arm forc russian feder

22602 rows × 4 columns

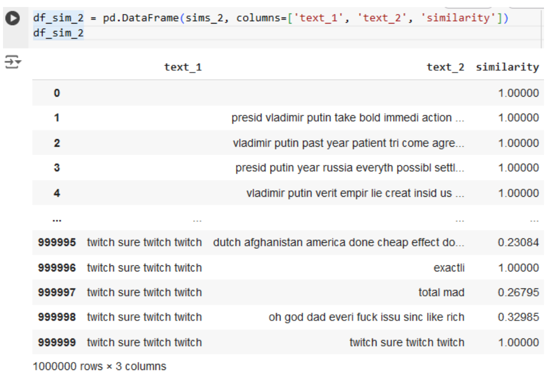
We use the old function to compare messages, this time 1000 x 1000:



```
[ ] sims_2 = calc_sim(clean_df_2, 1000)
    sims_2
```

```
/usr/local/lib/python3.10/dist-packages/scipy/spatial/distance.py:636: RuntimeWarr
  dist = 1.0 - uv / np.sqrt(uu * vv)
[['', '', 1],
 ['',
  'presid vladimir putin take bold immedi action peopl republ donbass ask russia h
special militari oper',
  1],
 ['',
  'vladimir putin past year patient tri come agreement regard principl equal indiv
lie attempt pressur blackmail',
  1],
```

We transfer the result to the dataframe:

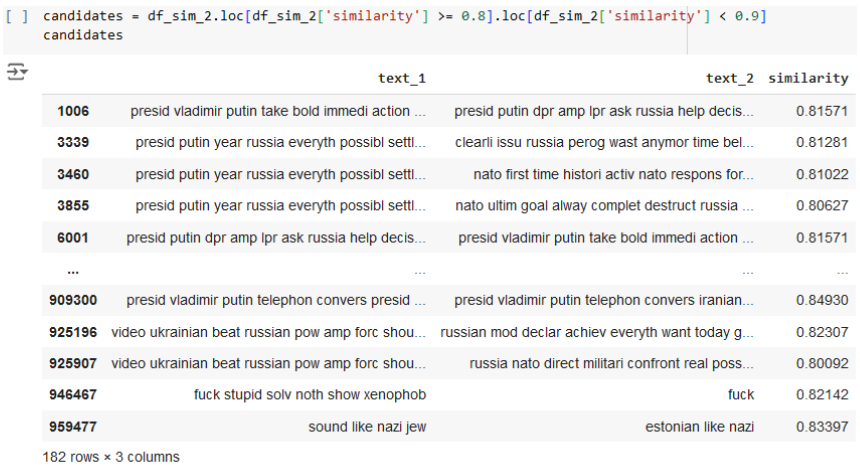


```
df_sim_2 = pd.DataFrame(sims_2, columns=['text_1', 'text_2', 'similarity'])
df_sim_2
```

	text_1	text_2	similarity
0			1.00000
1		presid vladimir putin take bold immedi action ...	1.00000
2		vladimir putin past year patient tri come agre...	1.00000
3		presid putin year russia everyth possibl settl...	1.00000
4		vladimir putin verit empir lie creat insid us ...	1.00000
...	...	...	...
999995	twitch sure twitch twitch	dutch afghanistan america done cheap effect do...	0.23084
999996	twitch sure twitch twitch	exactli	1.00000
999997	twitch sure twitch twitch	total mad	0.26795
999998	twitch sure twitch twitch	oh god dad everi fuck issu sinc like rich	0.32985
999999	twitch sure twitch twitch	twitch sure twitch twitch	1.00000

1000000 rows × 3 columns

It seems that after processing, some texts became empty, but they will still be rejected when selecting messages with a similarity of less than 1.

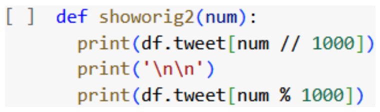


```
[ ] candidates = df_sim_2.loc[df_sim_2['similarity'] >= 0.8].loc[df_sim_2['similarity'] < 0.9]
    candidates
```

	text_1	text_2	similarity
1006	presid vladimir putin take bold immedi action ...	presid putin dpr amp lpr ask russia help decis...	0.81571
3339	presid putin year russia everyth possibl settl...	clearli issu russia perog wast anymor time bel...	0.81281
3460	presid putin year russia everyth possibl settl...	nato first time histori activ nato respons for...	0.81022
3855	presid putin year russia everyth possibl settl...	nato ultim goal alway complet destruct russia ...	0.80627
6001	presid putin dpr amp lpr ask russia help decis...	presid vladimir putin take bold immedi action ...	0.81571
...	...	...	...
909300	presid vladimir putin telephon convers presid ...	presid vladimir putin telephon convers iranian...	0.84930
925196	video ukrainian beat russian pow amp forc shou...	russian mod declar achiev everyth want today g...	0.82307
925907	video ukrainian beat russian pow amp forc shou...	russia nato direct militari confront real poss...	0.80092
946467	fuck stupid solv noth show xenophob	fuck	0.82142
959477	sound like nazi jew	estonian like nazi	0.83397

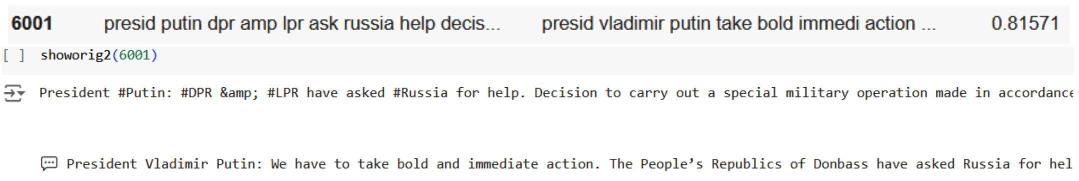
182 rows × 3 columns

We create a function similar to the previous one for viewing the original appearance of messages.



```
[ ] def showorig2(num):
      print(df.tweet[num // 1000])
      print('\n\n')
      print(df.tweet[num % 1000])
```

We check this line:

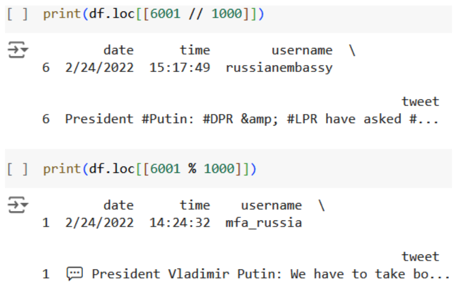


6001	presid putin dpr amp lpr ask russia help decis...	presid vladimir putin take bold immedi action ...	0.81571

```
[ ] showorig2(6001)

President #Putin: #DPR &amp; #LPR have asked #Russia for help. Decision to carry out a special military operation made in accordance


 💬 President Vladimir Putin: We have to take bold and immediate action. The People’s Republics of Donbass have asked Russia for hel
```

Both messages are about the beginning of a full-scale invasion, and they both say that "the people of the DNR and LNR are asking russia for help, and according to Article 51 of the United Nations Charter, russia is launching a 'special operation' in those lands".

The texts are similar, now it's worth checking where they were taken from.



```
[ ] print(df.loc[[6001 // 1000]])

        date      time         username  \
6  2/24/2022  15:17:49  russianembassy

                                               tweet
6  President #Putin: #DPR &amp; #LPR have asked #...

[ ] print(df.loc[[6001 % 1000]])

        date      time     username  \
1  2/24/2022  14:24:32  mfa_russia

                                               tweet
1  💬 President Vladimir Putin: We have to take bo...
```

The messages were posted on the same day, with an hour difference, and from different groups. It should be taken into account that it was such a large-scale event that it is possible that all the groups wrote about it. However, the text of the messages itself is so similar that we believe it is evidence of a propaganda network. In the mfa_russia group, the message was posted at 14:24, and in the russianembassy group - at 15:17, the chain looks like this: mfa_russia → russianembassy.

Let's try to write a function to automate the search for such chains.

```
def find_nets(dataframe):
  result = []

  for row in candidates.index:
    if df.username[row // 1000] != df.username[row % 1000]:
      result.append([df.username[row // 1000] + '___' + df.username[row % 1000]])

  return result
```

```
links = find_nets(candidates)
links = list(np.squeeze(links))
links
```

```
['mfa_russia___russianembassy',
 'russianembassy___thesiriusreport',
 'russianembassy___thesiriusreport',
 'russianembassy___russ_warrior',
 'russianembassy___mfa_russia',
 'alexxa1721___thesiriusreport',
 'russianembassy___alexxa1721',
 'russianembassy___russ_warrior',
 'russianembassy___mfa_russia',
 'colonelhomsi___alexxa1721',
 'mfa_russia___russianembassy',
 'russ_warrior___thesiriusreport',
 'russ_warrior___geromanat',
 'alexxa1721___thesiriusreport',
 'alexxa1721___russ_warrior',
 'russ_warrior___geromanat',
```

We compile a dictionary from the obtained results using the collections library:

```
from collections import Counter
```

```
Counter(links)
```

```
Counter({'mfa_russia___russianembassy': 2,
         'russianembassy___thesiriusreport': 2,
         'russianembassy___russ_warrior': 2,
         'russianembassy___mfa_russia': 2,
         'alexxa1721___thesiriusreport': 4,
         'russianembassy___alexxa1721': 1,
         'colonelhomsi___alexxa1721': 1,
         'russ_warrior___thesiriusreport': 2,
         'russ_warrior___geromanat': 3,
         'alexxa1721___russ_warrior': 3,
         'russ_warrior___alexxa1721': 3,
         'jacksonhinklle___colonelhomsi': 1,
         'colonelhomsi___mfa_russia': 1,
         'thesiriusreport___rwapodcast': 1,
         'gbazov___angieskys': 2,
         'gbazov___geromanat': 2,
         'angieskys___geromanat': 4,
         'angieskys___alexxa1721': 5,
         'rwapodcast___thesiriusreport': 1,
         'thesiriusreport___alexxa1721': 4,
```

Next, you can combine those cases when there are symmetrical chains, that is, from the first group to the second and from the second to the first. Then you can look at the number, how many cases of reinterpretation there are between groups, and based on this, conclude the presence of a propaganda network. You can also build longer chains of 3-4 or more groups. Potentially, you can further write functions that will themselves look at numbers from the dictionary and conclude the existence of the network.

However, this is where our research ends, as we do not have sufficient computing power to cover all available data. We could not process 6500 x 6500 messages, and in this dataset, there are as many as 22500 x 22500. Therefore, here we end our review and, accordingly, research.

7. Conclusions

At the beginning of the research, the topic of the project is formulated in the article, and the purpose and tasks of the research are described:

1. Recognition of propaganda messages.
2. Analysis of the spread of propaganda messages between groups, thereby finding propaganda networks.

The relevance, object and subject of the research are described. The means of influencing public opinion are studied, in particular propaganda, in particular russian propaganda. During the war with russia, the methods of combating information attacks of the enemy are extremely important. A search and review of data for the study was conducted. Several datasets of different structures were selected for further research. The quality of tagged data, the balance of tags and the presence of empty values are checked. Several regularities of specific datasets are deduced (regarding the length of messages, and the use of certain emoticons and keywords).

Processing of text data was carried out using a combination of different methods (conversion to lowercase, extraction of characters through regular expressions, extraction of stop words, stemming). The unique words present in the texts of the dataset were reviewed. In particular, a table of word frequencies was compiled, which demonstrates some regularities for certain words (for example, frequent use of the word "foreign" in propaganda). The best parameters for the binary classification of propaganda on machine learning models were selected, such as logistic regression (1-2-3-grams, 5000 features). A classification accuracy of ~0.85 was achieved.

Binary classification was tested using artificial neural networks, namely: a simple fully connected neural network with count vectors at the input, a neural network with embeddings at the input, and a transformer. Unusually, neural networks were unable to outperform logistic regression. Different methods of measuring the similarity (distance) between texts were tested: cosine similarity with different types of input vectors, Jaccard similarity, Leventschein distance, and cosine similarity on downloaded word2vec-google-news-300 embeddings. The last method proved to be the best, so it was used further. On several datasets, part of the messages were compared with each other, using the selected method of distance measurement. Similar messages were found, which most likely were reinterpreted, both within the same group and between different groups.

A method of calculating the number of reinterpretations between groups has been created, which allows the recognition of propaganda networks.

The results of the project:

1. For the first time, the fundamentals and main principles of the synthesized information technology for the automatic detection of sources of disinformation and inauthentic behaviour of chat users were developed, which will allow timely detection of destructive and suspicious communities in various social networks, identify their leaders and curators, identify information threats in user messages, prevent the spread of fake and harmful information.
2. For the first time, a method of stylistic analysis and linguistic processing of disinformation was developed to form an information portrait of the author/text content generation bot as part of the search parameters for both similar author content and distribution channels.
3. Developed criteria and parameters of inauthentic behaviour of chat users for the formation of information portraits of potential disinformation disseminators and detection of distribution routes and mechanisms, frequency of fake generation, topics and keywords characteristic of the relevant group.

NLP of the process of identifying content as fake/not fake is a complex process, as it depends not only on the speed/quality of pre-collected/integrated and processed content (blocked/unblocked in a certain region, content topics) but also on an effectively selected machine learning model on training sessions datasets. Usually, the fake is not blocked. The purpose of its creation is to spread it as quickly as possible both throughout the world and in those regions where usually true information (not fake) can potentially be blocked (not guaranteed). If non-fake information is blocked in a certain territory, and the opposite (fake) information is distributed from this territory, then the chance of identifying a fake increases. If the non-fake is not blocked and the fake is freely distributed in parallel, NLP methods will not help here. They can only label two sets with opposite explanations for an event/phenomenon. And only with additional statistical research, it is possible to identify which set is fake and which is not. The difficulty lies in the language of the content itself, in particular Ukrainian. In comparison with English-language content, Ukrainian/russian languages are quite difficult to automatically process, especially when analyzing semantics and building an ontology. Standard and traditional methods used for processing English-language tests are not suitable for processing Slavic languages, including for identifying disinformation and stylistic features of authors who generate fakes and propaganda. Similarly, in addition to the fact that the inauthentic behaviour of chat users, both people and bots, is different, people with different motivations (belief in propaganda, work for money, just one of the types of vandalism and, so to speak, leisure), nationality, education, gender, mentality, level of knowledge of the language of the text, degree of faith, intelligence, etc. All this significantly affects the process of determining the criteria for the behaviour of different communities and within the same community, which in turn significantly affects the formation of an informational portrait of the inauthentic behaviour of users of various chats (what is typical for a Muslim propagandist is significantly different for a representative of a rashka as russia or, respectively, lpr/dpr).

Justification of the practical value of the planned results of the project for the economy and society:

- Reducing the amount of disinformation, fakes and propaganda and the frequency/regularity of publication by tracking stylistically similar content and distribution routes.
- Reducing the negative impact of disinformation on public sentiment and reducing the degree of control of public opinion through the spread of propaganda in information warfare. For example, suppressing the psyche of young people (including mental disorders, leading to lethal consequences), encouraging them to engage in antisocial behaviour, forming groups of civil disobedience or aggressive behaviour on fabricated pretexts, analyzing the social consequences of cyberattacks, etc.
- Reducing the cost of finding, identifying and blocking disinformation, authors/targeted distributors and sources.

The development of the above-described methods is aimed at identifying threats, third-party intervention (attacks) in the early stages, classifying threats by type, and further countering each type of threat. Description of the ways and methods of further use of the results of the project in social practice.

1. Implementation of information security in the form of response to cyber threats related to the spread of fakes and propaganda in information warfare is an important modern practice in the USA and in many countries of Asia and the European Union.
2. The results of this project can be used in the information security organization of Ukraine to identify not only disinformation but also targeted groups with primary sources of dissemination. This synthesized information technology will make it possible to significantly reduce the overall negative impact on public sentiment and opinion, eliminate deceptive elements of public opinion management in society, and also reduce the cost of automatically finding, processing and blocking disinformation/distribution sources.
3. In the future, the results of the project may have a long-term social impact in the information sphere, in particular for the implementation of PSYOP among such directed groups in favour of Ukraine to ensure the cyber security of the state and inflict damage on the enemy in the information war.

The following risks may affect the implementation of the project:

- the risk associated with the war in Ukraine, the conduct of hostilities on the territory of the country, periodic mass shelling and the possibility of a blackout;
- the risk associated with the instability of economic legislation and the current economic situation;
- incompleteness and inaccuracy of information on the dynamics of technical and economic indicators, parameters of new equipment and technology.

Organizational, production and financial risks are not assumed. Possible technical risks are that the level of automation of the stages may not fully satisfy the initial requirements. It is possible to involve elements of the manual implementation of some tasks with the adjustment of the corresponding methods; a possible replacement in the use of basic mathematical methods of the project for the development of new models and methods. Then there will likely be a transition to traditional approaches or new ones (if new approaches and information technologies are developed in the world in the next two years); the development of individual methods and tools may go beyond the established deadlines. Then, during the implementation of the demo version of the system, traditional approaches will be applied to ensure the minimal functionality of the methods.

References

[1] I. Afanasieva, N. Golian, V. Golian, A. Khovrat, K. Onyshchenko, Application of Neural Networks to Identify of Fake News, CEUR Workshop Proceedings 3396 (2023) 346-358.

[2] A. Shupta, O. Barmak, A. Wierzbicki, T. Skrypnyk, An Adaptive Approach to Detecting Fake News Based on Generalized Text Features, CEUR Workshop Proceedings 3387 (2023) 300-310.

[3] V.-A. Oliinyk, V. Vysotska, Y. Burov, K. Mykich, V. Basto-Fernandes, Propaganda Detection in Text Data Based on NLP and Machine Learning, CEUR workshop proceedings 2631 (2020) 132-144.

[4] R. A. Dar, Dr. R. Hashmy, A Survey on COVID-19 related Fake News Detection using Machine Learning Models, CEUR Workshop Proceedings 3426 (2023) 36-46.

[5] V. Vysotska, S. Mazepa, L. Chyrun, O. Brodyak, I. Shakleina, V. Schuchmann, NLP Tool for Extracting Relevant Information from Criminal Reports or Fakes/Propaganda Content, in Proceedings of IEEE 17th International Conference on Computer Sciences and Information Technologies (CSIT), 2022, pp. 93-98, doi: 10.1109/CSIT56902.2022.10000563.

[6] A. Mykytiuk, V. Vysotska, O. Markiv, L. Chyrun, Y. Pelekh, Technology of Fake News Recognition Based on Machine Learning Methods, CEUR Workshop Proceedings 3387 (2023) 311-330.

[7] Y. Zhao, J. Da, J. Yan, Detecting health misinformation in online health communities: Incorporating behavioral features into machine learning based approaches, Information Processing & Management 58(1) (2021) 102390.

[8] M. Hartmann, Y. Golovchenko, I. Augenstein, Mapping (dis-)information flow about the MH17 plane crash, arXiv:1910.01363, 2019.

[9] S. Ahmed, Classification of Censored Tweets in Chinese Language using XLNet, in Proceedings of the Fourth Workshop on NLP for Internet Freedom: Censorship, Disinformation, and Propaganda, 2021, pp. 136-139.

[10] V. Vysotska Modern state and prospects of information technologies development for natural language content processing, CEUR Workshop Proceedings 3668 (2024) 198–234.

[11] I. Zamaruieva, S. Lienkov, O. Babich, A. Shevchenko, Y. Khlaponin, N. Bernaz, Analytical Approaches to News Content Processing during the War in Ukraine in Opposing Geopolitical Alliances Mass Media, CEUR Workshop Proceedings 3403 (2023) 618-631.

[12] V. Vysotska, Computer Linguistic Systems Design and Development Features for Ukrainian Language Content Processing, CEUR Workshop Proceedings 3688 (2024) 229–271. URL: https://ceur-ws.org/Vol-3688/paper18.pdf.

[13] S. Albota, Creating a Model of War and Pandemic Apprehension: Textual Semantic Analysis, CEUR Workshop Proceedings 3396 (2023) 228-243.

[14] N. Khairova, Y. Holyk, D. Sytnikov, Y. Mishcheriakov, N. Shanidze, Topic Modelling of Ukraine War-Related News Using Latent Dirichlet Allocation with Collapsed Gibbs Sampling, CEUR Workshop Proceedings 3688 (2024) 1-15.

[15] S. Mainych, A. Bulhakova, V. Vysotska, Cluster Analysis of Discussions Change Dynamics on Twitter about War in Ukraine, CEUR Workshop Proceedings 3396 (2023) 490-530.

[16] R. Nazarchuk, S. Albota, Tweets about Ukraine during the russian-Ukrainian War: Quantitative Characteristics and Sentiment Analysis, CEUR Workshop Proceedings 3426 (2023) 551-560.

[17] N. Khairova, A. Kolesnyk, O. Mamyrbayev, K. Mukhsina, The Aligned Kazakh-Russian Parallel Corpus Focused on the Criminal Theme, CEUR Workshop Proceedings 2362 (2019) 116-125.

[18] S. Voloshyn, V. Vysotska, O. Markiv, I. Dyyak, I. Budz, V. Schuchmann, Sentiment Analysis Technology of English Newspapers Quotes Based on Neural Network as Public Opinion Influences Identification Tool, in Proceedings of 2022 IEEE 17th International Conference on Computer Sciences and Information Technologies (CSIT), 2022, pp. 83-88, doi: 10.1109/CSIT56902.2022.10000627.

[19] N. Khairova, A. Shapovalova, O. Mamyrbayev, N. Sharonova, K. Mukhsina, Using BERT model to Identify Sentences Paraphrase in the News Corpus, CEUR Workshop Proceedings 3171 (2022) 38-48.

[20] N. Bondarchuk, I. Bekhta, O. Melnychuk, O. Matviienkiv, Keyword-based Study of Thematic Vocabulary in British Weather News, CEUR Workshop Proceedings 3171 (2022) 451-460.

[21] S. Voloshyn, O. Markiv, V. Vysotska, I. Dyyak, L. Chyrun, V. Panasyuk, Emotion Recognition System Project of English Newspapers to Regional E-Business Adaptation, Proceedings of IEEE 17th International Conference on Computer Sciences and Information Technologies (CSIT), 2022, pp. 392-397, doi: 10.1109/CSIT56902.2022.10000527.

[22] N. Antonyuk, L. Chyrun, V. Andrunyk, A. Vasevych, S. Chyrun, A. Gozhyj, I. Kalinina, Y. Borzov, Medical news aggregation and ranking of taking into account the user needs, CEUR Workshop Proceedingsnn248 (2019) 369–382.

[23] V. Andrunyk, A. Vasevych, L. Chyrun, N. Chernovol, N. Antonyuk, A. Gozhyj, V. Gozhyj, I. Kalinina, M. Korobchynskyi, Development of information system for aggregation and ranking of news taking into account the user needs, CEUR Workshop Proceedings 2604 (2020) 1127–1171.

[24] V. Vysotska, S. Voloshyn, O. Markiv, O. Brodyak, N. Sokulska, V. Panasyuk, Tone Analysis of Regional Articles in English-Language Newspapers Based on Recurrent Neural Network Bi-LSTM, in Proceedings of the 5th International Conference on Advanced Information and Communication Technologies (AICT), 2023, pp. 158-163.

[25] S. Albota, Linguistic and Psychological Features of the Reddit News Post, in Proceedings of the IEEE 15th International Scientific and Technical Conference on Computer Sciences and Information Technologies, CSIT, 2020, 1, pp. 295–299.

[26] N. Shakhovska, M. Medykovskyj, L. Bychkovska, Building a smart news annotation system for further evaluation of news validity and reliability of their sources, Przeglad Elektrotechniczny 91(7) (2015) 43-44.

[27] V. Vysotska, R. Holoshchuk, S. Goloshchuk, O. Voloshynskyi, M. Shevchenko, V. Panasyuk, Predicting the Effects of News on the Financial Market Based on Machine Learning Technology, in Proceedings of the 5th International Conference on Advanced Information and Communication Technologies (AICT), 2023, pp. 152-157.