The project hase been developped during an Internship of 5 months, from the 3d April 2023 to the 12th September 2023, realized in Paris associations.

The objectives:

- Professional Experience: Engage in full-time professional practice within a Parisian association
- Skill Development: Cultivate skills as a development engineer, focusing on real-world applications
- Career Exploration: Investigate potential career trajectories, especially in computer sciences applied to societal challenges
- Hands-on Learning: Immerse in new techniques, methodologies, tools, and their practical applications
- Association Ethics: Grasp the ethical considerations and operational nuances of associations
- Professional Adaptability: Evaluate one's ability to integrate and adapt in a professional environment
- Job Market Transition: Equip oneself with the skills and experience to seamlessly enter the job market

Context and motivation

The technologies has been always influenced greatly the social relations. In particular, the present-day computer science is an exceptional tool for cooperation and co-reflection.

This project is aimed at automatical detecting of propagandistic information in Telegram [1] channels. That is, the information, which may not be objective and may be selectively presenting facts to encourage a particular perception, or using loaded language to produce an emotional rather than a rational response to the information. [2]

The methodology

The metodogy applicated was close to the agile practices which include requirements discovery and solutions improvement through self-organizing and cross-functional teams with the users, except that, the projet being a litle one, both the developpers team and the users were represented by myself only.

The approach to development followed the Scrum principles like:

- to break work into goals to be completed within time-boxed iterations, called sprints, where the sprints were dedicated to the contexte, the large language models, the architecture, python and its interactions with the concerned API study, the databasbases and elasticsearch, frontend);
- bringing decision-making authority to an operational level;
- continuous feedback and flexibility;
- changing of the requirements as the project evolves.

Aquired competences

A lof of experience, completely new for me, like:

- the conception of a whole application with several servers, several API, several external services
- programming languages languanges (python, node.js, javascript)
- frameworks (Vue, express, ...)

What the application does

1. Real time verification of Telegram messages veracity by two methods:

- looking for the marks of the propaganda [3] in every separate message via OpenAI
- comparison, via OpenAI, of the recent information diffused by several channels, in order to detect similar channels

2. Constant improvement of the results by the mean of a Learning service, which consists in attaching some previous OpenAI responses, corrected by a user, to OpenAI requests

Simplified diagram of the application (except Learning service)

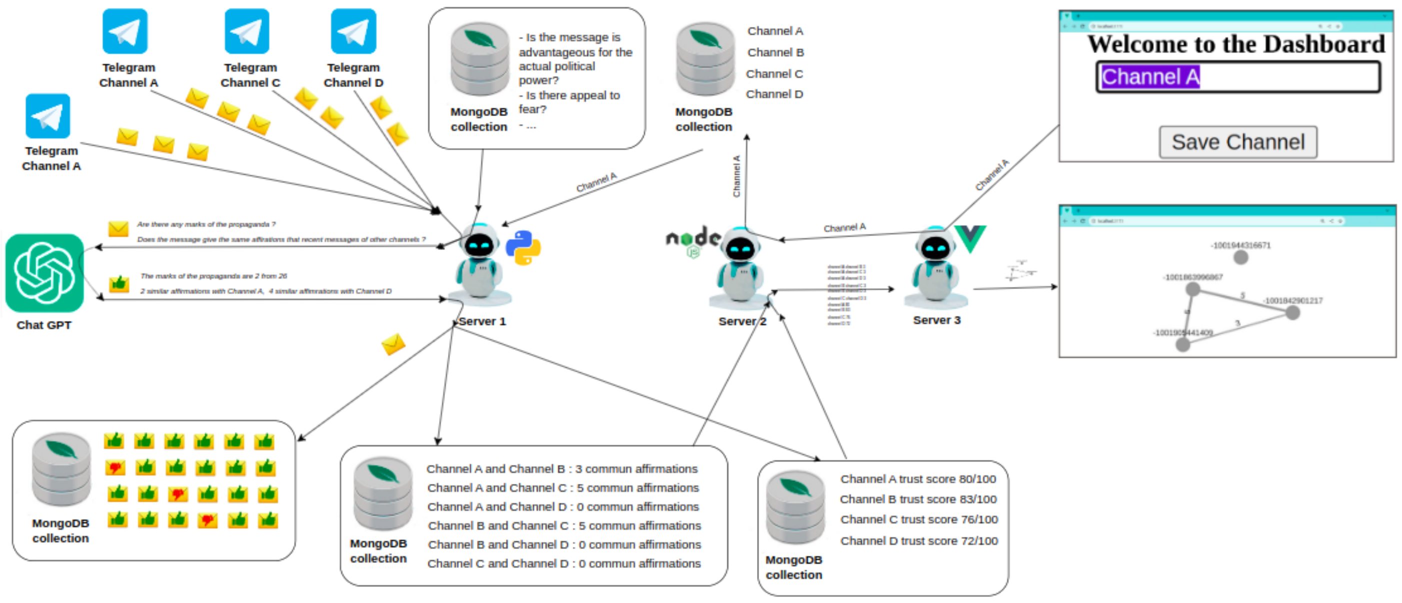
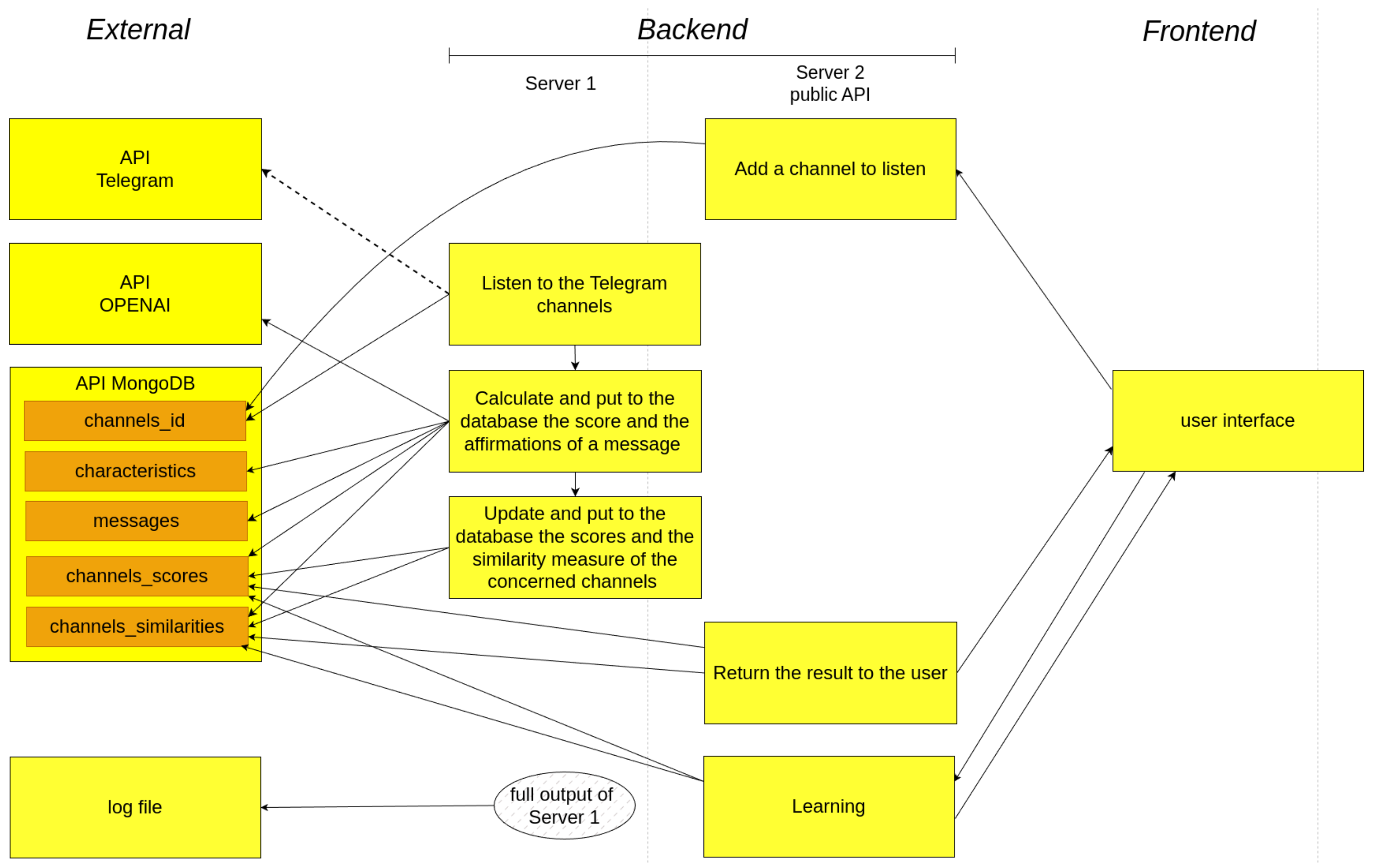

Telegram Channel A
Telegram Channel C
Telegram Channel D
Telegram Channel A
MongoDB collection
- Is the message is advantageous for the actual political power?
- Is there appeal to fear?
- ...
MongoDB collection
Channel A
Channel B
Channel C
Channel D
Welcome to the Dashboard
Channel A
Save Channel
Channel A
Channel A
Channel A
Channel A
Are there any marks of the propaganda ?
Does the message give the same affirations that recent messages of other channels ?
The marks of the propaganda are 2 from 26
2 similar affirmations with Channel A, 4 similar affimrations with Channel D
Chat GPT
Server 1
Server 2
Server 3
MongoDB collection
MongoDB collection
Channel A and Channel B : 3 commun affirmations
Channel A and Channel C : 5 commun affirmations
Channel A and Channel D : 0 commun affirmations
Channel B and Channel C : 5 commun affirmations
Channel B and Channel D : 0 commun affirmations
Channel C and Channel D : 0 commun affirmations
MongoDB collection
Channel A trust score 80/100
Channel B trust score 83/100
Channel C trust score 76/100
Channel D trust score 72/100

Diagram of the application in the programmer style


External
Backend
Frontend
Server 1
Server 2
public API
API
Telegram
API
OPENAI
API MongoDB
channels_id
characteristics
messages
channels_scores
channels_similarities
log file
Listen to the Telegram channels
Calculate and put to the database the score and the affirmations of a message
Update and put to the database the scores and the similarity measure of the concerned channels
full output of Server 1
Add a channel to listen
Return the result to the user
Learning
user interface

Description of the application in English

1. *Server 3* gets from the web browser a name of Telegram channel to examinate, via Server 3 API
2. *Server 3* transmet the name of the channel to *Server 2*, via Server 2 API
3. *Server 2* put puts the name of the channel to MongoDB Atlas database, via MongoDB API
4. *Server 1* permanently gets the names of the channels from MongoDB Atlas, via MongoDB API
5. *Server 1* listens permanently to the chosen channels, via Telegram API
6. *Server 1* treats every new message, that is:

- estimates the marks of the propaganda of the message, via OpenAI API
- based on these marks of the propaganda it calculates the trust coefficient of the message
- extracts the principal information of the new message, in the form of several affirmations, via OpenAI API
- compares these affirmations to the recent affirmations of the other followed channels
- stocks all obtained information in MongoDB Atlas database (the message itself, the result if its analisys, updates the trust coefficients of the channels, updates the measure of similarity of the channels), via MongoDB API

7. *Server 2* consults permanently the results of the computations in MongoDB Atlas, via MongoDB API
8. *Server 2* returns permanently the current results of the computations to the *Server 3*, via Server 2 API
9. *Server 3* passes the the results to the web browser in the form of a graph of the channels, where every summit contains the id of the channel and its trust coefficient, and every edge is the measure of similarity of two concerned channels, via Server 3 API
10. The web browser displays the graph to the user

Simultaneously, the *Learning service* is working:

1. *Server 3* proposes to the user to correct OpenAI's previous responses in the web browser, via the Server 3 API
2. As soon as the user provides the corrected examples, *Server 3* passes them to *Server 2*
3. *Server 3* puts the corrected examples to the database, via MongoDB API
4. *Server 1* attaches (a limited number of) these corrected examples to every new OpenAI request

How to use the application

The user should have a web browser compatible with ECMAScript 5 (for example, IE8 and its previous versions are not compatible with ECMAScript 5)

MongoDB Atlas configuration

MondoDB Atlas in a database MongoDB the in cloud.

Create a MongoDB account

In your account, create a database by the name 'telegram'

Import the collection 'characteristics' from this file to your database 'telegram': 🟥

```
sudo mongoimport --db telegram --collection characteristics --file collection_characte
```

Go to the MongoDB interface - Database Deployments - `to add your current ip address`

Go to the MongoDB interface - Database - Connect - Drives - `to get you MongoDB connection string`

In the line 13, here, put the same MongoDB connection token (connection string) as in `server1/.env` :
.env 🟥

```
const mongoUrl = '...';
```

Be careful not to publish your MongoDB connection token on the internet and not to transmit it to unfamiliar people

OpenAI configuration

Get your OpenAI connection token

Your account should have access to gpt-4 (a paying option)

Be very very careful not to publish your OpenAI connection token on the internet and not to transmit it to unfamiliar people

Telegram configuration

Get Telegram credentials api_id and api_hash

During the first launching of the application, enter the phone number of your Telegram account and then enter the confirmation code

The application will create a session file `anon.session` in the folder `server1` in order to you can to login without re-sending the code.

Be very very careful not to publish the Telegram credentials on the internet and not to transmit them to unfamiliar people

`.env` file configuration

Put MONGO = your MongoDB connection token, OPENAI = OpenAI connection token, API_ID and API_HASH = Telegram credentials in the file `server1/.env.example` :

```
API_ID=...
API_HASH=...
OPENAI=...
MONGO=...
```

Rename `server1/.env.example` to `server1/.env`

Copy `server1/.env` to `server2/.env`

Be very very careful not to publish this file on the internet and not to transmit it to unfamiliar people

Setup

Install python version >= 3.7.1

Install the python libraries :

```
pip3 install telethon
pip install DateTime
npm i dotenv
pip install requests
pip install pymongo
pip install --upgrade openai
```

Install nmp package manager

```
cd server3
npm install
```

PS `npm install` should be executed in the same folder where `package.json` file is

Compile and run

In the first terminal launch *Server 1*

```
python server1/server1.py
```

In the second terminal launch *Server 2*

```
node server2/server2.js
```

In the third terminal launch *Server 3*

```
cd user_interface
npm run dev
```

Go to the user interface

After having installed and configured all noted above, enjoy the service http://localhost:5173/

Technical details of the deveppement

To unserstand this section, the reader should have basic knowledge of the teminology of computer sciences

Separation of the data treatment provided by *Server 1* and the presentation functoinality provided by *Server 2* and *Server 3* falls into the pattern of Model-View-ViewModel (MVVM).

Server 1 and *Server 2* represent the backend functionality, while *Server 3* ensures the Frontend one.

The parameters of the application

- The characteristics of the propaganda
- The text of the characteristics request
- The text of the affirmations request
- OpenAI model for a characteristics request
- OpenAI model for a affirmations request
- OpenAI temperature, between 0 and 1. Higher values like 0.8 will make the output more random, while lower values like 0.2 will make it more deterministic. If set to 0, the model will use log probability to automatically increase the temperature until certain thresholds are hit. [4]
- OpenAI request maximal lenth (in tokens [5])
- Telegram message maximal length (in characters)
- The time where a message is considered as recent (in hours)

Computation details

Similarity measure of two channels (channel1, channel2) = the numbre of their similar affirmations - the number of their opposite affirmations

The trust coefficient of a channel is a number in the interval [0 … 100]

The application does O(N) OpenAI requests and O(N*K) MongoDB requests, where N is the total numbre of messages, K is the nuber of followed channels

Python details

The *Server 1* is written in Python, because:

- Python is well adapted to data science projects because of its specilised libraries like telethon, DateTime, requests, pymongo, openai
- Python is a rather easy language (partly becauseof its easy syntax)

Presentation functionality (node.js and Vue) details

Node.js, a asynchronous event-driven JavaScript runtime environment and library, runs the application outside of the client's web browser, which is a raisonable choice because no function in Node.js directly performs I/O, so the process never blocks except when the I/O is performed using synchronous methods of Node.js standard library [6].

Vue framework choice as programming model to develop the user interface (regarding to Angular and React as its possible alternatives), is explained by the simple syntax of Vue, its intuitive documentation and its pertinence for smalle projects and novice developers [7] [8].

Express Node web framework, is used by *Server 2* to:

- write handlers for requests
- set the port to use, and the location of templates that are used for rendering the response
- integrate with "view" rendering engines in order to generate responses by inserting data into templates.

Vite, a local development server used by Vue, monitors files as they're being edited. Upon file save the web browser reloads the code being edited through a process called Hot Module Replacement which works by reloading only the specific file being changed. [9]

Asynchrony details

Asynchrony refers to the actions instigated by a program that take place concurrently with program execution, without the program blocking to wait for results. In this application, it concerns the requests to Telegram, OpenAI and MongoDB.

Telegram requests are asynchronous

OpenAI server 1 requests are **not asynchronous**

MongoDB server 1 requests are **not all asynchronous**

Server 2 requests 🟥

Database details

A noSql database usage is explained chiefly by the changing number of the `characteristics` , as well as by changing of the `characteristics` themselves, while adjusting the application (it concerns the collections `characteristics` and `messages`).

MongoDB usage is explained by its SaaS offer and its popularity. The databse operations of this project are simple, so other noSql databases would provide the same functionnalty and at the speed.

Other technical details

Output of server 1 is saved in the logs

The limits and improvements of the application

It is developped only for Linux.

The installation and configuration instructions are complicated for a user, they should be unified in one instruction by using Docker.

The limits related to OpenAI

Gpt-4 treats only about 5 requests per minute. However, this large language model analysis, respesenting the lowest part of the appliation, may be accelerated :

- by involving a great number of powerful machines
- by involving a grand number of OpenAI accounts
- using of other language models can be envisaged, for example

The quality of OpenAI analisys may be improuved :

- by cross-analysis by several language models
- by prompt design
- by learning ()
- by fine-tuning

Alternatives to the paid approach could be to train a self-hosted model (like LLama2) on a corpus proofread by humans, since it has been proven that smaller models can perform way better than larger models. [10]

The length of an examined message is limited (see The parameters of the application), a message is cut off beyond this length

The learning service is limited to 5 examples par a request (but if the message, the examples and the OpenAI response are altogether longer than *The maximal length of OpenAI request* parameter, then the learning is limited to less than 5 examples)

Extraction of affirmations gives acceptable results only with Gpt-4 (not with Gpt-3) and only with some examples included in the request.

The limits related to MongoDB

The installation instructions are provided in this document only for the cloud version MongoDB (MongoDB Atlas), however the user can install MongoDB locally.

The speed of MongoDB requests is not critical in this project, because they are much faster than OpenAI requests. However, once OpenAI operations are accelerated (by the means descibed above or others), the databade requests can be accelerated by installing MongoDn locally et may be by using other databases.

A BSON document in MongoDB cannot excede 16 Mb [11] and a MongoDB database cannot exceed 64 TB in size.

The limits related to Telegram

- The application **can't read some channels**, for example this one

The limits related to Vue

Vue supports web browsers compatible with ECMAScript 5

Other possible improvement

Some othe feautres could be added, like:

- notifications for exceptional analysis results;
- detection of the first channel to spead information;
- comparison of the rusults with other projects.

Conceptual limits

The learning and the choice of the characteristics are founded on a human subjective opinion.

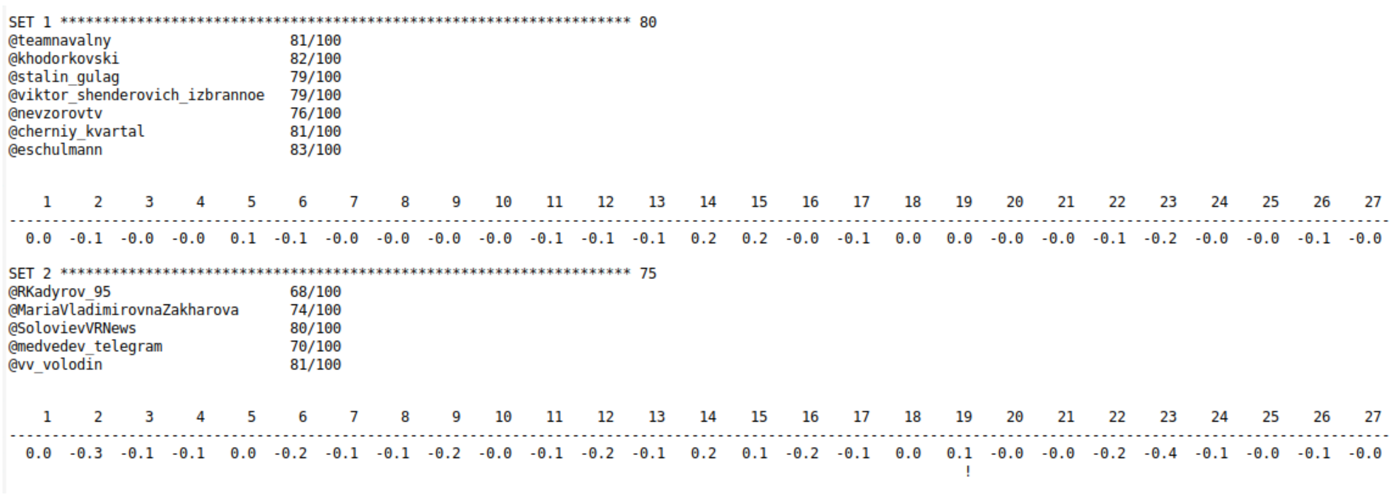
The application may help to the malefactors to adjust propagandistic messages to make them pass unnoticed.

The application doesn't aime at the deep causes of the propaganda.

Experimentations

OpenAI experimentations

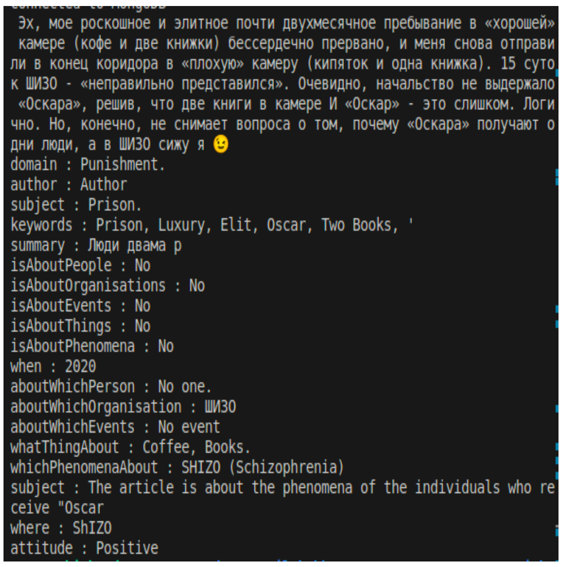
The teste launched on two groups of channels, a propagandistic group and a non-propagandistic one (accordingly to personal intuition), shows the difference of the average trust coefficient of the two groups between 3 and 8 point:



```
SET 1 ******************************************************************* 80
@teamnavalny                      81/100
@khodorkovski                     82/100
@stalin_gulag                     79/100
@viktor_shenderovich_izbrannoe    79/100
@nevzorovtv                       76/100
@cherniy_kvartal                  81/100
@eschulmann                       83/100


    1     2     3     4     5     6     7     8     9    10    11    12    13    14    15    16    17    18    19    20    21    22    23    24    25    26    27
---------------------------------------------------------------------------------------------------------------------------------------------------------------------
  0.0  -0.1  -0.0  -0.0   0.1  -0.1  -0.0  -0.0  -0.0  -0.0  -0.1  -0.1  -0.1   0.2   0.2  -0.0  -0.1   0.0   0.0  -0.0  -0.0  -0.1  -0.2  -0.0  -0.0  -0.1  -0.0

SET 2 ******************************************************************* 75
@RKadyrov_95                      68/100
@MariaVladimirovnaZakharova       74/100
@SolovievVRNews                   80/100
@medvedev_telegram                70/100
@vv_volodin                       81/100


    1     2     3     4     5     6     7     8     9    10    11    12    13    14    15    16    17    18    19    20    21    22    23    24    25    26    27
---------------------------------------------------------------------------------------------------------------------------------------------------------------------
  0.0  -0.3  -0.1  -0.1   0.0  -0.2  -0.1  -0.1  -0.2  -0.0  -0.1  -0.2  -0.1   0.2   0.1  -0.2  -0.1   0.0   0.1  -0.0  -0.0  -0.2  -0.4  -0.1  -0.0  -0.1  -0.0
                                                                                                       !
```

Two tests on the values of the trust coefficient depending in the temperature parameter, every of the tests launched on the two groups of channels, show a tendancy of better distinction between the two groups while the temperature parameter is hier.

Temperature	0.0	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1.0
difference test 1	2	2	2	2	3	4	3	3	5	5	5
difference test 2	1	1	2	3	1	1	3	4	2	4	4



```
 Эх, мое роскошное и элитное почти двухмесячное пребывание в «хорошей»
 камере (кофе и две книжки) бессердечно прервано, и меня снова отправи
ли в конец коридора в «плохую» камеру (кипяток и одна книжка). 15 суто
к ШИЗО - «неправильно представился». Очевидно, начальство не выдержало
 «Оскара», решив, что две книги в камере И «Оскар» - это слишком. Логи
чно. Но, конечно, не снимает вопроса о том, почему «Оскара» получают о
дни люди, а в ШИЗО сижу я 😉
domain : Punishment.
author : Author
subject : Prison.
keywords : Prison, Luxury, Elit, Oscar, Two Books, '
summary : Люди двама p
isAboutPeople : No
isAboutOrganisations : No
isAboutEvents : No
isAboutThings : No
isAboutPhenomena : No
when : 2020
aboutWhichPerson : No one.
aboutWhichOrganisation : ШИЗО
aboutWhichEvents : No event
whatThingAbout : Coffee, Books.
whichPhenomenaAbout : SHIZO (Schizophrenia)
subject : The article is about the phenomena of the individuals who re
ceive "Oscar
where : ShIZO
attitude : Positive
```

Extracting of detailed information (like the main subject, the people it deals with, etc) from a message, that is "undesrstanding" of the message, didn't worked correctly because of the poor quality of the analysis. Sorry for the example in Russian.

The comparaison of paires of messages directly via OpenAI (instead of extracting the principal information in the form of affirmations) demands O(N^2) operations and so is too long (see log example).

The direct question to OpenAI, *Is there any marks of the propagande in this message?*, didn't work correctly.

The classifying of the channes into groups according to their subject didn't prove to be useful in this project.

Other experimentations

Keeping of a part of the data in the application memory, and not in the database : because the application has no acces to the results of the previous executions

Elasticsearch 🟥

Distances euclidienne, jaccard, cos, ... 🟥

дописать 🟥

Some projects of similar orientation

- Detecting of communities with similar ideologies by cross-channel interactions by DRFLab
- The project of Huan Cao [ru], exploring activity and localistation of the users, etc
- Machine learning project Faking Fake News for Real Fake News Detection: Propaganda-loaded Training Data Generation
- Machine learning project Botometer
- Machine learning project by The Institute of Mathematical and Computing Sciences (Brazil)
- Machine learning project Buster.ai
- Workshops Fever
- Auxipresse

Welcome

All the questions are welcome

Get help to install the application

1. Telegram is an application similar to WhatApp, Viber, Signal, etc. Its particularities are that it has a lot of channels (a channel is a one-way broadcast tool) on different subjects, chiefly in Russian, with little censorship. ↩
2. https://www.britannica.com/topic/propaganda ↩
3. https://www.cairn.info/revue-questions-de-communication-2020-2-page-371.htm ↩
4. https://platform.openai.com/docs/api-reference/audio/createTranscription#audio/createTranscription-temperature ↩
5. in English 1 token ≈ 3/4 of a word ↩
6. https://nodejs.org/en/about ↩
7. https://skillbox.ru/media/code/vuejs-chto-takoe-kak-on-ustroen-i-chem-otlichaetsya-ot-react/ ↩
8. https://www.codeinwp.com/blog/angular-vs-vue-vs-react/#gref ↩
9. https://en.wikipedia.org/wiki/Vite_(software) ↩
10. https://deepgram.com/learn/the-underdog-revolution-how-smaller-language-models-outperform-llms ↩
11. https://www.mongodb.com/docs/manual/reference/limits/ ↩