BLONDEL Hector
BUI Hugo
BOHARD Charly
CERVERA Romain
ESPAÑA Tomas
ROBY Edouard

CW - YoutubeCleaner

https://gitlab-ovh-02.cloud.centralesupelec.fr/edouard.roby/insultedetector_s2_YouTubeCleaner

CentraleSupélec

- Fonctionnalités :
  - Récupérer les commentaires et réponses d'une vidéo
  - Détecter les insultes dans un commentaire
  - Classer les vidéos selon le ratio d'insultes qu'elles contiennent
  - Afficher des diagrammes circulaires renseignant sur le taux d'insultes des commentaires d'une vidéo

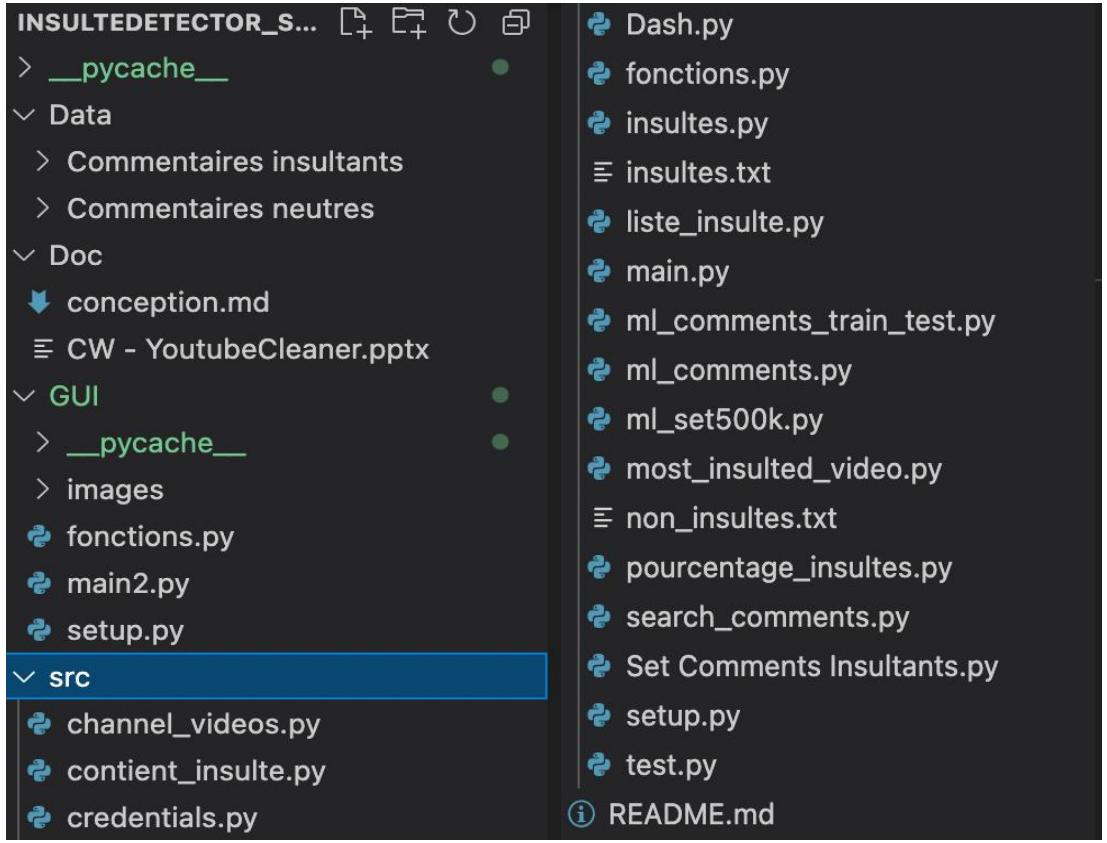
- Ce projet s'adresse au grand public comme aux professionnels, par exemple :
  - Pour de simples utilisateurs de Youtube, à des fins d'information
  - Pour des Youtubeurs, afin de mieux connaître leur communauté
  - Pour les entreprises désireuses de réaliser un partenariat avec un Youtubeur

Le projet

YoutubeCleaner est un projet créé par des étudiants de CentraleSupélec dans le cadre des Coding Weeks 2021 – 2022

La qualité du code

Structure, modules, packages, tests, documentation



Qualité du code

Cette diapositive détaille la structure du code et les modules nécessaires au bon fonctionnement du programme.

Structure du code

- Data : données relatives au projet
- Doc : documentation liée au projet
- src : programmes Python
  - MVP : programmes permettant l'exécution du MVP
  - ML : programmes permettant l'amélioration du MVP via le machine learning
- Readme : informations à lire avant d'exécuter le projet

Modules et packages

- click
- google-auth-oauthlib
- google-auth-httplib2
- google-api-python-client
- bs4
- requests
- pyperclip
- dash
- nltk
- sklearn
- tkinter
- json
- os
- csv
- codecs
- pickle
- time

Qualité du code

Cette diapositive détaille les différents tests effectués et la documentation associée au projet.

Tests

- Toutes les fonctionnalités ont été testées individuellement.
- Certains *print* de test ont été laissés en commentaire sous les fonctions.
- La cohérence des fonctionnalités entre elles a été testée, notamment via le fichier main.py.
- La qualité du code est assurée notamment par les commentaires, l'indentation, le nommage des variables et fonctions, …

Documentation

- L'ensemble de la documentation du code est inclus dans le fichier readme : objectifs du projet, membres de l'équipe, structure du code, indications d'utilisation, usage, etc.

Développement du projet

Répartition du travail, sprints et fonctionnalités

Développement du projet - Répartition des tâches

BLONDEL Hector

- API Youtube
- Optimisation
- distance d'édition

BUI Hugo

- Méthode pour identifier une insulte
- Optimisation
- Machine Learning

BOHARD Charly

- Interface Dash
- Machine Learning

CERVERA Romain

- Méthode pour identifier une insulte
- Documentations, architecture du projet, etc
- Etablissement de la liaison avec l'API

ESPAÑA Tomas

- Machine Learning
- API Youtube

ROBY Edouard

- API Youtube
- Mise en place du GUI

Développement du projet - Découpage en sprints

Ce projet suit la *méthode agile*, qui consiste en un découpage en sprints et fonctionnalités.

Phase 1 : Préparation du MVP

- Sprint 0 : Analyse des besoins et conception
  - Production d'un readme et d'un fichier de conception
- Sprint 1 : Collecte des commentaires YouTube
  - Connexion à l'API YouTube et configuration
  - Recherche des commentaires selon divers critères (nom de chaîne, réponses à un commentaire, mots clés…)
  - Outil de statistiques sur les commentaires
- Sprint 2 : Traitement et analyse des commentaires
  - Création d'une liste d'insultes
  - Détection d'insultes dans un commentaire
  - Recherche de la vidéo avec le plus d'insultes d'une chaîne

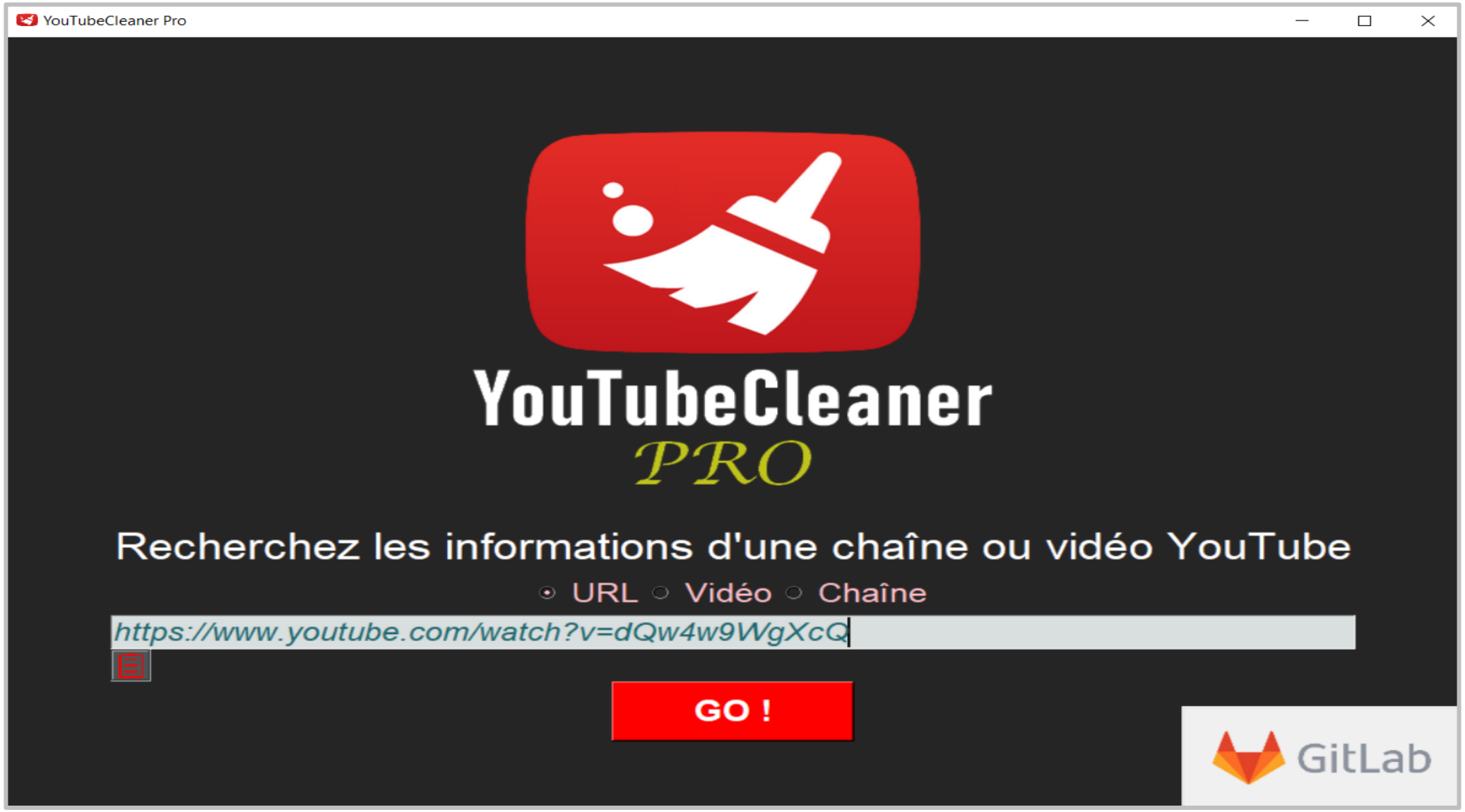
Phase 2 : Réalisation et amélioration du MVP

- Sprint 3 : Visualisation des résultats
  - Interface Dash
  - Interface graphique pour le projet
- Sprint 4 : Amélioration du MVP par machine learning
  - Entraînement d'un modèle de machine learning à partir de sets existants
  - Optimisation des paramètres du modèle
  - implémentation du code dans le MVP

Démonstration et résultats

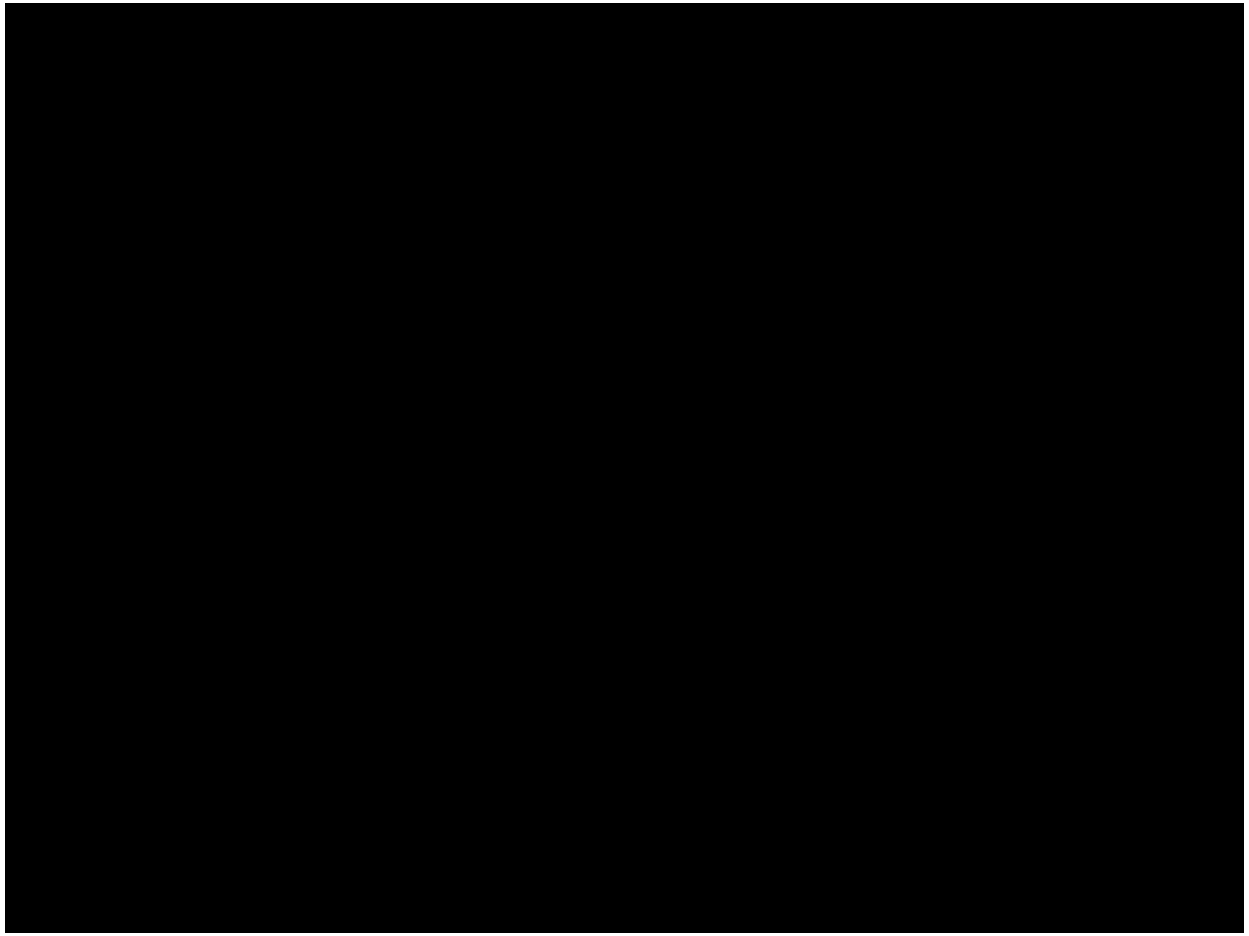
Vidéo de démonstration et graphes associés à divers tests

Interface graphique - Fenêtre principale



Interface graphique - Fenêtre Dash

Démonstration vidéo



Pour aller plus loin…

Ce qu'on aurait fait avec plus de temps :

On aurait pu utiliser le code du modèle de machine learning :

Nous avons entraîné un modèle à partir de 160k commentaires insultants et neutres

→ Modèle utilisé : RandomForestClassifier = arbres décisionnels (modèle conseillé pendant la première semaine des CW)

Problème : Temps d'entraînement du modèle très long (jusqu'à une centaine de minutes pour certaines valeurs de paramètres)

→ Optimisation des paramètres très longue

Essai 1

	precision	recall
Neutre	0.98	0.99
Insulte	0.81	0.52

Essai 2

	precision	recall
Neutre	0.96	0.99
Insulte	0.86	0.62

Notre petite fierté

```
vectorizer = CountVectorizer(
    max_features=2000, min_df=0, max_df=0.7, stop_words=stopwords.words('english'))
X = vectorizer.fit_transform(documents).toarray()

tfidfconverter = TfidfTransformer()
X = tfidfconverter.fit_transform(X).toarray()

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.1)
classifier = RandomForestClassifier(n_estimators=100)
classifier.fit(X_train, y_train)

##############################################################################

commentaires_dico = get_all(video_id)
nb_commentaires = len(commentaires_dico)
liste_panda = [commentaires_dico[elt] for elt in commentaires_dico]
X = pd.Series(liste_panda)
```

```
vectorizer = CountVectorizer(
    max_features=2000, min_df=0, max_df=0.7, stop_words=stopwords.words('english'))
X = vectorizer.fit_transform(documents).toarray()

tfidfconverter = TfidfTransformer()
X = tfidfconverter.fit_transform(X).toarray()

liste = classifier.predict(X)

nb_insultes = 0
nb_neutre = 0
for i in range(len(liste)):
    if liste[i] == 0:
        nb_neutre += 1
    if liste[i] == 1:
        nb_insultes += 1

tf = time.time()

print("pourcentage d'insultes identifiées:",
      (nb_insultes/nb_commentaires)*100)
print("pourcentage de messages neutres identifiés:",
      (nb_neutre/nb_commentaires)*100)
print("temps d'exécution en secondes:", tf-ti)
```