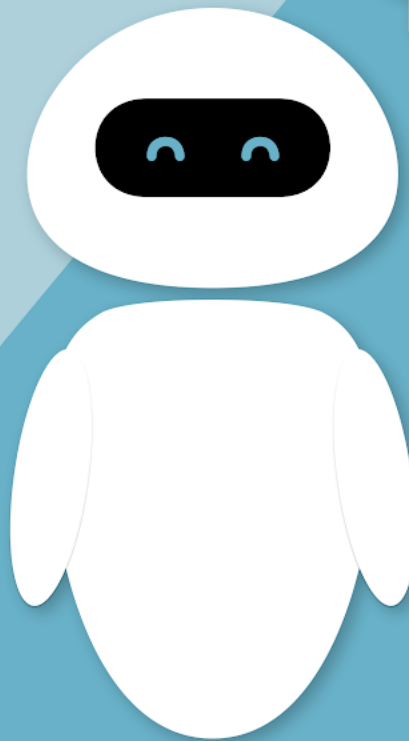


CHATBOT

Alvarez Luka, Calunsag Sabina



Hi ! What's your
favorite movie ?



September 2020, V 1.0



TABLE DES MATIÈRES

INTRODUCTION	3
Description	3
Pourquoi avoir choisi ce sujet ?	3
Quel logiciel de développement ?	3
Pourquoi python ?	3
CAHIER DES CHARGES	4
But	4
Fonctionnalités à réaliser	4
Outils	4
ANALYSE FONCTIONNELLE	5
Interface	5
RAPPORT DE TESTS	6
Début du projet	6
Intégration d'une autre librairie	6
INSTALLATION DE PYTORCH	7
Pour Linux	7
Pour Windows	9
Vérification	13
DÉPLOIEMENT DU PROJET	13
Installation	14
Configuration	14
Entraînement	15
Tester	15
ANNEXES	16
Plannings	16
Références	17
Autre	17

INTRODUCTION

Description

Nous allons faire une IA qui analyse les messages envoyés et répond aux questions des utilisateurs en rapport avec le cinéma qui utilise un dataset contenant des données sur des films.

Dans un prochain temps, réussir à changer / ajouter la dataset du bot pour qu'il apprenne nos goûts sur le cinéma.

Pourquoi avoir choisi ce sujet ?

Étant des informaticiens, nous sommes au courant de l'importance de l'IA dans le monde actuel. Aujourd'hui il y a des IA qui sont utilisées partout. C'est pour cela que nous avons choisi ce projet pour mieux comprendre le fonctionnement des IA de manière générale.

Quel logiciel de développement ?

Pour programmer ce chatbot en python nous avons décidé d'utiliser visual studio code. Il permet de facilement intégrer git directement dans l'éditeur. En plus vscode est facilement modulable grâce à ses extensions.

Pourquoi python ?

Python est un langage de programmation facile à apprendre et très performant...

CAHIER DES CHARGES

But

Le but de ce projet est d'élargir notre connaissance sur le fonctionnement d'un chatbot et de réussir à lui apprendre des choses à partir d'un modèle de données. À la fin du projet, on va pouvoir interagir avec le chatbot en utilisant une interface sur web.

Fonctionnalités à réaliser

- Setup l'environnement
- Tester le projet
- Reproduire le projet
- Changer de modèle de donnée
- Entraîner le chatbot
- Faire une interface graphique

Outils

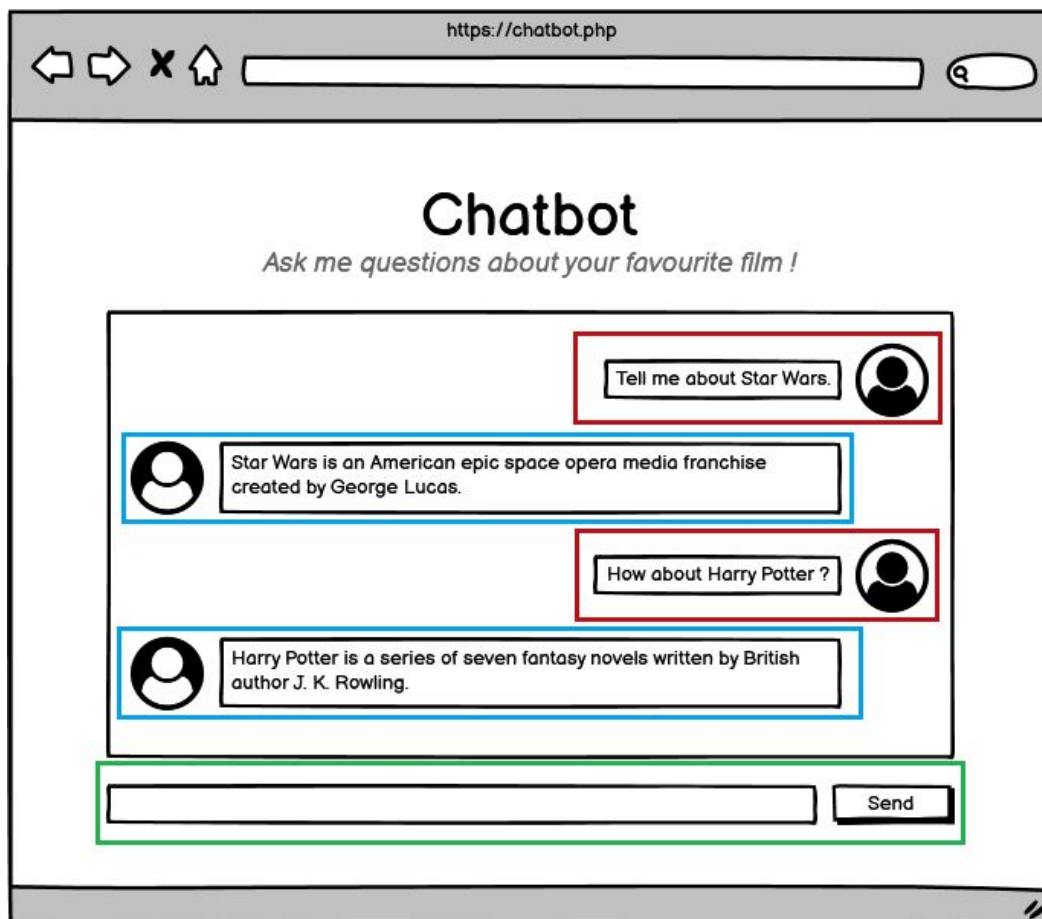
- PyTorch
- Python
- Librairies
- Balsamiq
- MonoDevelop

ANALYSE FONCTIONNELLE

Interface

Pour la partie visuelle, nous avons décidé de produire une interface avec MonoDevelop avec l'utilisation de GTK. On peut voir sur l'image ci-dessous, une maquette de la page principale qui permet à l'utilisateur de communiquer avec le chatbot à l'aide d'un message box.

- **Rouge** : Les messages envoyés par l'utilisateur avec un icône à droite.
- **Bleu** : Les messages envoyés par le chatbot(IA) avec un icône à gauche.
- **Vert** : Un textbox pour insérer un message puis un bouton à droite pour l'envoyer.



RAPPORT DE TESTS

Début du projet

Nous avons utilisé le langage python et le framework Pytorch pour la réalisation de notre projet chatbot. Nous, qui connaissons très peu le fonctionnement et le développement d'un chatbot, nous avons décidé de partir sur un tutoriel puis par la suite intégrer d'autres librairies dans le projet pour lui apprendre.

Pour mettre en place cela, nous avons d'abord installé Pytorch et en suivant le tutoriel nous avons rencontré quelques erreurs que nous avons réussi à résoudre après quelque temps.

Intégration d'une autre librairie

Par la suite, nous allons essayer d'intégrer une librairie différente de celle du tutoriel.

INSTALLATION DE PYTORCH

Pour Linux

Premièrement il faut installer Python 3 sur la machine les commandes peuvent être différentes en fonction des différentes distributions.

sudo apt install python

sudo pacman -S python

```
+ pytorch-chatbot gll(master) sudo pacman -S python
warning: python-3.9.1-1 is up to date -- reinstalling
resolving dependencies...
looking for conflicting packages...

Packages (1) python-3.9.1-1

Total Installed Size: 79.38 MiB
Net Upgrade Size: 0.00 MiB

:: Proceed with installation? [Y/n] y
(1/1) checking keys in keyring [-----] 100%
(1/1) checking package integrity [-----] 100%
(1/1) loading package files [-----] 100%
(1/1) checking for file conflicts [-----] 100%
:: Processing package changes...
(1/1) reinstalling python [-----] 100%
:: Running post-transaction hooks...
(1/1) Arming ConditionNeedsUpdate...
+ pytorch-chatbot gll(master)
```

Ensuite il faut installer l'outil python "pip"

sudo apt install python3-pip

sudo pacman -S python-pip

```
→ ~ sudo pacman -S python-pip

[sudo] password for luka:
warning: python-pip-20.2.4-1 is up to date -- reinstalling
resolving dependencies...
looking for conflicting packages...

Packages (1) python-pip-20.2.4-1

Total Installed Size: 1.56 MiB
Net Upgrade Size: 0.00 MiB

:: Proceed with installation? [Y/n] y
(1/1) checking keys in keyring [-----] 100%
(1/1) checking package integrity [-----] 100%
(1/1) loading package files [-----] 100%
(1/1) checking for file conflicts [-----] 100%
:: Processing package changes...
(1/1) reinstalling python-pip [-----] 100%
:: Running post-transaction hooks...
(1/1) Arming ConditionNeedsUpdate...
→ ~
```

Pour finir il suffit d'exécuter la command en fonction de la version voulu que l'on peut trouver ici : <https://pytorch.org/get-started/locally/>

PyTorch Build	Stable (1.7.1)		Preview (Nightly)		
Your OS	Linux	Mac	Windows		
Package	Conda	Pip	LibTorch	Source	
Language	Python		C++ / Java		
CUDA	9.2	10.1	10.2	11.0	None
Run this Command:	<pre>pip install torch==1.7.1+cpu torchvision==0.8.2+cpu torchaudio==0.7.2 -f https://download.pytorch.org/whl/torch_stable.html</pre>				

L'option cuda sert a choisir si les test sont effectués avec la puissance gpu ou du cpu en cas du choix none c est le cpu qui calculera.

On peut aussi choisir entre la version stable ou bêta de pytorch ici nous allons utiliser la version stable

Il existe aussi d'autres méthodes d'installation comme avec l'outil conda ou alors plus compliqué avec libtorch ou directement depuis le code source en le compilant.

Pour Windows

Pour la réalisation de ce projet, nous avons utilisé un logiciel open source qui s'appelle **PyTorch**. Pour installer cela, nous avons suivi le tutoriel du site officiel de PyTorch : <https://pytorch.org/get-started/locally/>

Sur l'image ci-dessous, nous pouvons personnaliser les préférences et pour ce projet, nous avons choisi d'utiliser Conda qui contient aussi Pip lors de l'installation.

PyTorch Build	Stable (1.7.1)		Preview (Nightly)	
Your OS	Linux	Mac	Windows	
Package	Conda	Pip	LibTorch	Source
Language	Python		C++ / Java	
CUDA	9.2	10.1	10.2	11.0
				None
Run this Command:	NOTE: Python 3.9 users will need to add '-c=conda-forge' for installation conda install pytorch torchvision torchaudio cpuonly -c pytorch			

Pour la réalisation du projet nous avons codé du python. Pour le télécharger, il faut aller sur le site : <https://www.python.org/downloads/windows/>

Une fois sur le site, il faut cliquer sur le lien pour **Python 3.9.1**, montré sur l'image ci-dessous. Comme la version 2.x.x ne marche pas pour PyTorch, il faut forcément une version 3.x.x.

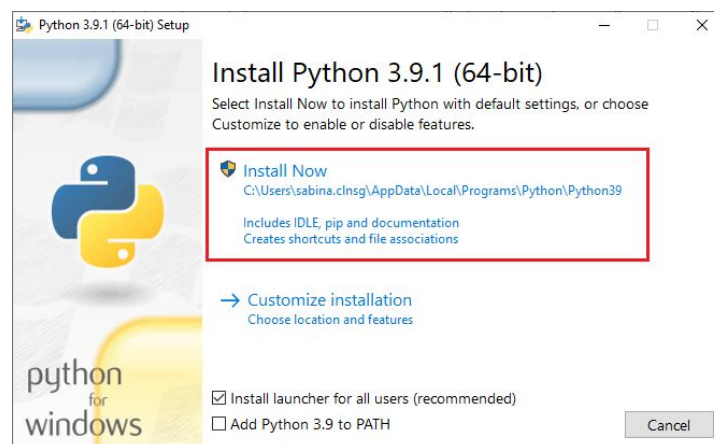
Python Releases for Windows

- [Latest Python 3 Release - Python 3.9.1](#)
- [Latest Python 2 Release - Python 2.7.18](#)

Sur l'image ci-dessous, on peut voir plusieurs versions de Python par rapport au système d'exploitation. Comme nous sommes sur Windows, nous allons forcément prendre une version Windows. Pour notre projet, nous avons téléchargé la version recommandée en 64-bit.

Files					
Version	Operating System	Description	MD5 Sum	File Size	GPG
Gzipped source tarball	Source release		429ae95d24227f8fa1560684fad6fca7	25372998	SIG
XZ compressed source tarball	Source release		61981498e75ac8f00adcb908281fad6	18897104	SIG
macOS 64-bit Intel Installer	Mac OS X	for macOS 10.9 and later	74f5cc5b5783ce8fb2ca55f11f3f0699	29795899	SIG
macOS 64-bit universal2 installer	Mac OS X	for macOS 10.9 and later, including macOS 11 Big Sur on Apple Silicon (experimental)	8b19748473609241e60aa3618bbaf3ed	37451735	SIG
Windows embeddable package (32-bit)	Windows		96c6fa81fe8b650e68c3dd41258ae317	7571141	SIG
Windows embeddable package (64-bit)	Windows		e70e5c22432d8f57a497cde5ec2e5ce2	8402333	SIG
Windows help file	Windows		c49d9b6ef88c0831ed0e2d39bc42b316	8787443	SIG
Windows installer (32-bit)	Windows		dde210ea04a31c27488605a9e7cd297a	27126136	SIG
Windows installer (64-bit)	Windows	Recommended	b3fce2ed8bc315ad2bc49eae48a94487	28204528	SIG

Une fois téléchargé, il faut démarrer l'installation en cliquant sur "Install Now" jusqu'à que l'installation soit terminée.



Ensuite, il faut installer **Anaconda**, un package manager qui permet d'installer les binaires PyTorch : <https://www.anaconda.com/products/individual#windows>

Anaconda Installers

Windows 🖥️

Python 3.8

64-Bit Graphical Installer (457 MB)

32-Bit Graphical Installer (403 MB)

MacOS 🍏

Python 3.8

64-Bit Graphical Installer (435 MB)

64-Bit Command Line Installer (428 MB)

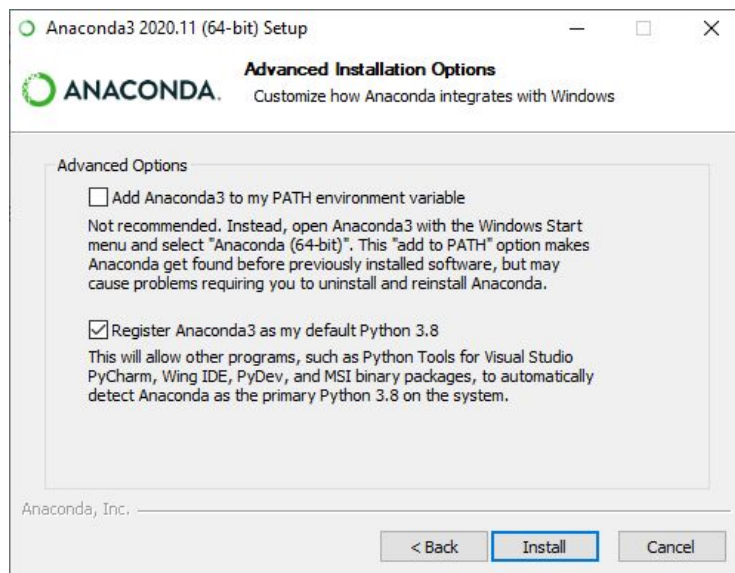
Linux 🐧

Python 3.8

64-Bit (x86) Installer (529 MB)

64-Bit (Power8 and Power9) Installer (279 MB)

Il faut installer la version Windows 64 ou 32-bit. Une fois téléchargé, lancez l'exé puis appuyez sur "Suivant" jusqu'à l'apparition du bouton "Installer".



Maintenant que nous avons installé **Anaconda**, pour passer à l'étape suivante, il faut d'abord ouvrir **Anaconda Prompt (anaconda3)**. Ensuite, pour installer Pytorch, il faut mettre cette ligne.

Run this Command:

```
NOTE: Python 3.9 users will need to add '-c=conda-forge' for installation  
conda install pytorch torchvision torchaudio cpuonly -c pytorch
```

Comme c'est écrit ci-dessus, il ne faut pas oublier de rajouter **-c=conda-forge** si le Python installé est une version 3.9.

conda install pytorch torchvision torchaudio cpuonly -c pytorch -c=conda-forge

La fenêtre devra ressembler à l'image ci-dessous si PyTorch est bien en train de s'installer. Si l'installation est terminée, ça devra être écrit "done".

```

Anaconda Prompt (anaconda3) - conda install pytorch torchvision torchaudio cpuonly -c pytorch -c=conda-forge
CondaSystemExit: Exiting.

(base) C:\Users\sabina.clnsg>conda install pytorch torchvision torchaudio cpuonly -c pytorch -c=conda-forge
Collecting package metadata (current_repodata.json): done
Solving environment: done

## Package Plan ##

  environment location: C:\Users\sabina.clnsg\anaconda3

  added / updated specs:
    - cpuonly
    - pytorch
    - torchaudio
    - torchvision

The following packages will be downloaded:

package | build | size | channel
-----|-----|-----|-----
conda-4.9.2 | py38haa244fe_0 | 3.1 MB | conda-forge
cpuonly-1.0 | 0 | 2 KB | pytorch
libuv-1.40.0 | h8ffe710_0 | 330 KB | conda-forge
ninja-1.10.2 | h5362a0b_0 | 273 KB | conda-forge
python_abi-3.8 | 1_cp38 | 4 KB | conda-forge
pytorch-1.7.1 | py3.8_cpu_0 | 156.8 MB | pytorch
torchaudio-0.7.2 | py38 | 2.7 MB | pytorch
torchvision-0.8.2 | py38_cpu | 6.6 MB | pytorch
-----|-----|-----|-----
Total: | | 169.8 MB |

The following NEW packages will be INSTALLED:

cpuonly | pytorch/noarch::cpuonly-1.0-0
libuv | conda-forge/win-64::libuv-1.40.0-h8ffe710_0
ninja | conda-forge/win-64::ninja-1.10.2-h5362a0b_0
python_abi | conda-forge/win-64::python_abi-3.8-1_cp38
pytorch | pytorch/win-64::pytorch-1.7.1-py3.8_cpu_0
torchaudio | pytorch/win-64::torchaudio-0.7.2-py38
torchvision | pytorch/win-64::torchvision-0.8.2-py38_cpu

The following packages will be SUPERSEDED by a higher-priority channel:

conda | pkgs/main::conda-4.9.2-py38haa95532_0 --> conda-forge::conda-4.9.2-py38haa244fe_0

Proceed ([y]/n)? y

Downloading and Extracting Packages
libuv-1.40.0 | 330 KB | ##### | 100%
python_abi-3.8 | 4 KB | ##### | 100%
ninja-1.10.2 | 273 KB | ##### | 100%
torchvision-0.8.2 | 6.6 MB | ##### | 100%
conda-4.9.2 | 3.1 MB | #####
pytorch-1.7.1 | 156.8 MB | #####

```

Vérification

Pour vérifier si pytorch a bien été installé il faut créer une file python comme ceci :

```
import torch
x = torch.rand(5, 3)
print(x)
```

Maintenant on peut exécuter le scripts python et l'output devrait ressembler à cela :

```
tensor([[0.3380, 0.3845, 0.3217],
        [0.8337, 0.9050, 0.2650],
        [0.2979, 0.7141, 0.9069],
        [0.1449, 0.1132, 0.1375],
        [0.4675, 0.3947, 0.1426]])
```

DÉPLOIEMENT DU PROJET

Installation

La première étape est de clone depuis github le projet de base

git clone <https://github.com/ywk991112/pytorch-chatbot>

```
→ ~ git clone https://github.com/ywk991112/pytorch-chatbot
Cloning into 'pytorch-chatbot'...
remote: Enumerating objects: 318, done.
remote: Total 318 (delta 0), reused 0 (delta 0), pack-reused 318
Receiving objects: 100% (318/318), 63.75 KiB | 453.00 KiB/s, done.
Resolving deltas: 100% (199/199), done.
→ ~
```

Configuration

Ensuite il faut créer un corpus file qui va nous permettre de poser les questions réponses que l'on veut faire dire à notre IA. On peut utiliser l'exemple de base ou faire le notre.

location => /data/<corpus file name>

Le créateur du projet met à disposition un modèle déjà entraîné pour tester mais nous on veut entraîner le robot nous même donc on a fait cette commande pour l'entraîner. On lui a aussi donné notre propre corpus .

Pour utiliser un modèle existant il faut faire ça :

créer le répertoire => mkdir -p save/model/movie_subtitles/1-1_512

Puis déplacer le model dans la four :

mv 50000_backup_bidir_model.tar save/model/movie_subtitles/1-1_512

Entraînement

Pour commencer à entraîner l'IA il faut donner la file corpus et exécuter cette commande :

`python main.py -tr <CORPUS_FILE_PATH> -la 1 -hi 512 -lr 0.0001 -it 50000 -b 64 -p 500 -s 1000`

```
+ pytorch-chatbot [111:master] python main.py -tr ./data/movie_subtitles.txt -la 1 -hi 512 -lr 0.0001 -it 50000 -b 64 -p 500 -s 1000
Start loading training data ...
Building encoder and decoder ...
Building optimizers ...
Initializing ...
0%|                                     | 2/50000 [00:10<75:41:26, 5.45s/it]
```

Si un modèle est déjà existant et que l'on veut continuer à l'entraîner on peut faire ceci et ajoutant juste le liens vers le modèle déjà existant :

`python main.py -tr <CORPUS_FILE_PATH> -l <MODEL_FILE_PATH> -lr 0.0001 -it 50000 -b 64 -p 500 -s 1000`

































Tester

































Pour tester le résultat de l'IA il faut exécuter cette commande et préciser le lien vers le model et le corpus file.


`python main.py -te <MODEL_FILE_PATH> -c <CORPUS_FILE_PATH>`

ANNEXES

Plannings

	Planning Prévisionnel 1								
	31.09.2020	07.09.2020	14.09.2020	21.09.2020	28.09.2020	05.10.2020	12.10.2020	19.10.2020	26.10.2020
Conception	 	 							
Poster			 						
Cahier des charges				 					
Ajout des fonctionnalités supplémentaires (Ajouter une autre librairie)					 	 	 	 	 
Interface									
Documentation			 	 	 	 	 	 	 

	Planning Prévisionnel 2								
	02.11.2020	09.11.2020	16.11.2020	23.11.2020	30.11.2020	07.12.2020	14.12.2020	21.12.2020	
Conception									
Poster									
Cahier des charges									
Ajout des fonctionnalités supplémentaires (Ajouter une autre librairie)	 	 	 	 	 	 			
Interface							 	 	
Documentation	 	 	 	 	 	 	 	 	

Légende :	Luka		Sabina	
-----------	------	---	--------	---

Références

<https://code.visualstudio.com/docs/python/python-tutorial>

<https://github.com/ywk991112/pytorch-chatbot/tree/master>

Autre

Error :

FileNotFoundError: [Errno 2] No such file or directory:

'./save/training_data/corpus/voc.tar' During handling of the above exception, another exception occurred:

=> create directory : save/training_data/corpus/voc.tar

RuntimeError: Attempting to deserialize object on a CUDA device but torch.cuda.is_available() is False. If you are running on a CPU-only machine, please use torch.load with map_location=torch.device('cpu') to map your storages to the CPU.

line 488 => map_location=None => map_location='cpu'

file => /home/luka/.local/lib/python3.7/site-packages/torch/serialization.py

Après avoir réglé ces différents problèmes, nous sommes passés par le training, c'est-à-dire envoyer des données au chatbot pour qu'il apprenne au fur à mesure.