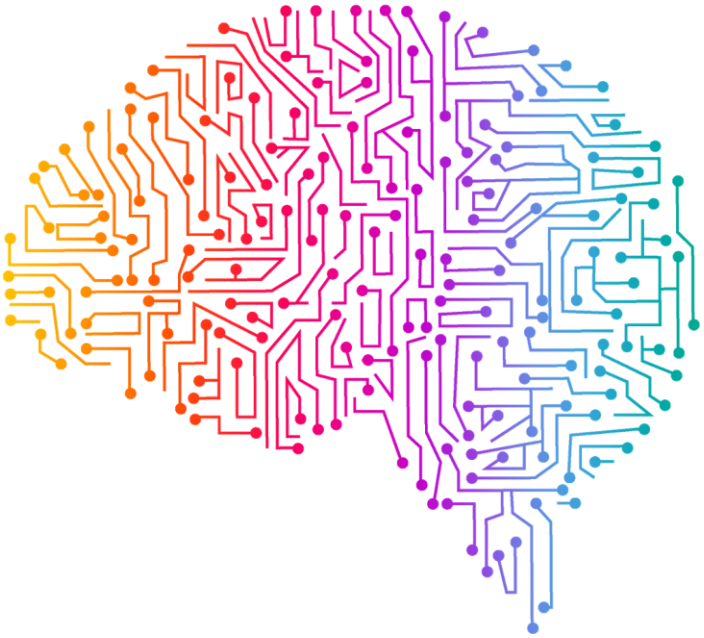
Bordeaux INP
ENSC

*Rapport technique*

# Projet Informatique Individuel : Chatbot AlBidert



Lucie BANNAY

Tuteur : Théodore LETOUZE

*Avril 2021*

# Table des matières

*I – Introduction* ..... 5
1. **Contexte** ..... 5
2. **Motivations et enjeux** ..... 5
3. **Contraintes** ..... 5

*II - Déroulement du projet* ..... 6
1. **Planning de travail** ..... 6
2. **Démarche** ..... 7
Rédaction du cahier des charges ..... 7
Prise en main de Python ..... 7
Développement du premier prototype de chatbot ..... 7
Modélisation du langage naturel ..... 8
Récupération de données ..... 8
Machine Learning ..... 8
Interface graphique ..... 9
3. **Réunions Tuteur** ..... 9

*III - Application* ..... 10
1. **Architecture** ..... 10
2. **Fonctionnement** ..... 11
Fonctionnement global ..... 11
Chatbot ..... 11
a) Programme principal ..... 11
b) Interface graphique ..... 12
Choix technique ..... 12
Présentation du module utilisé ..... 12
Fonctionnement de interface.py ..... 12
*Initialisation* ..... *12*
*Fonction get_response* ..... *13*
c) Algorithme : ELIZA ..... 15
Choix technique ..... 15
Présentation du chatbot utilisé ..... 15
Fonctionnement de eliza.py ..... 15
Algorithme de détection d'émotions ..... 16

a) Base de données utilisée ..... 16
b) Programme principal ..... 17
c) Programme de vectorisation de texte ..... 17
Choix technique ..... 17
Présentation du module utilisé ..... 17
Fonctionnement de csvVect.py ..... 18
Fonctionnement NLP de spaCy ..... 18
*Fonctionnement général ..... 18*
*Pipeline model ..... 19*
*Définition des étapes principales du nlp ..... 19*
d) Algorithme de machine-learning ..... 19
Choix technique ..... 19
Présentation du module utilisé ..... 20
Fonctionnement de fonctionML.py ..... 20
Présentation de MLPClassifier ..... 20
Fonctionnement d'un MLP ..... 20
*Perceptron 1 couche ..... 20*
*Perceptron multicouche ..... 21*
Tests ..... 21
***IV – Pistes d'évolution ..... 22***
**1. Techniques ..... 22**
Chatbot ..... 22
a) ELIZA en français ..... 22
b) Ajouter une IA ..... 22
c) Interface : site-web ou application mobile ..... 22
Détection d'émotion ..... 22
a) Ajouter des bases de données ..... 22
b) Algorithme de machine-learning ..... 23
*Réduction de la taille du vecteur ..... 23*
*Autres algorithmes et paramètres ..... 23*
Tests utilisateurs ..... 23
**2. Fonctionnelles ..... 23**
Activité ..... 23
Résultats automatiques ..... 23
Validation des résultats ..... 23

Choix de la langue ..... 23
Analyse de la personnalité ..... 24
Compte et historique des conversations ..... 24
Explication de la raison du résultat final ..... 24
*V - Conclusion ..... 24*
*VI - Bibliographie ..... 25*

# Table des figures

Figure 1. Exigences fonctionnelles primaires ..... 5
Figure 2. Exigences techniques primaires ..... 5
Figure 3. Planning Janvier-Février ..... 6
Figure 4. Planning Mars-Avril ..... 6
Figure 5. Tâches et dates de début et fin ..... 7
Figure 6. Arborescence simplifiée de l'application ..... 10
Figure 7. Fonctionnement de l'application ..... 11
Figure 8. Programme principal du chatbot : main.py ..... 12
Figure 9. Illustration du fonctionnement de interface.py ..... 13
Figure 10. Illustration du cas 1 : Dialogue avec l'utilisateur ..... 14
Figure 11. Illustration du cas 2 : affichage de l'analyse des émotions ..... 14
Figure 12. Illustration du fonctionnement de ELIZA ..... 16
Figure 13. Extrait de la base de données ISEAR ..... 17
Figure 14. Illustration du fonctionnement de CsvVect.py ..... 18
Figure 15. Schématisation de la fonction nlp de spaCy ..... 18
Figure 16. Schématisation du fonctionnement du pipeline model ..... 19
Figure 17. Schématisation de l'algorithme du perceptron ..... 20
Figure 18. Illustration d'un MLP ..... 21

# I – Introduction

## 1. Contexte

Ce projet a été réalisé dans le cadre du module Projet Informatique Individuel qui s'inscrit dans l'UE Sciences Fondamentales du S8 de l'ENSC. L'objectif de ce projet à sujet libre était de monter en compétences sur une/des technologie(s) de notre choix tout en travaillant sur un domaine inspirant personnellement. Le délai imposé était de 3 mois (début du projet le 21 Janvier 2021).

## 2. Motivations et enjeux

Avec la crise sanitaire, de nombreuses personnes peuvent se sentir isolées et démotivées. C'est pourquoi j'ai décidé de développer un chatbot pour permettre à ces personnes d'avoir "quelqu'un" à qui parler. La particularité de ce chatbot est qu'il détecte les émotions ressenties par la personne, pour l'aider à mieux comprendre son état émotionnel, et donc à aller mieux.

## 3. Contraintes

Les contraintes ont été fixées en début de projet par le biais d'un Cahier Des Charges. Voici un rappel des exigences primaires :

| Code | Description |
|---|---|
| EF_1 | AlBidert fait parler l'utilisateur. |
| EF_2 | AlBidert détecte l'humeur de l'utilisateur. |

*Figure 1. Exigences fonctionnelles primaires*

| Code | Description |
|---|---|
| ET_1 | L'application est codée en langage Python. |
| ET_2 | L'application utilise du machine learning |
| ET_3 | L'ensemble du code source respecte les bonnes pratiques de développement logicielles (pythonCase, KISS, YAGNI, etc.). |
| ET_4 | Le code a un dépôt GitHub. |

*Figure 2. Exigences techniques primaires*

# II - Déroulement du projet

Dans cette partie, je présente le planning du travail effectué, ainsi que la démarche suivie pour mener à bien ce projet.

## 1. Planning de travail

Voici le planning du projet. En rouge sont représentées les dates de rendu des livrables. En vert sont illustrées la date des réunions avec le tuteur.

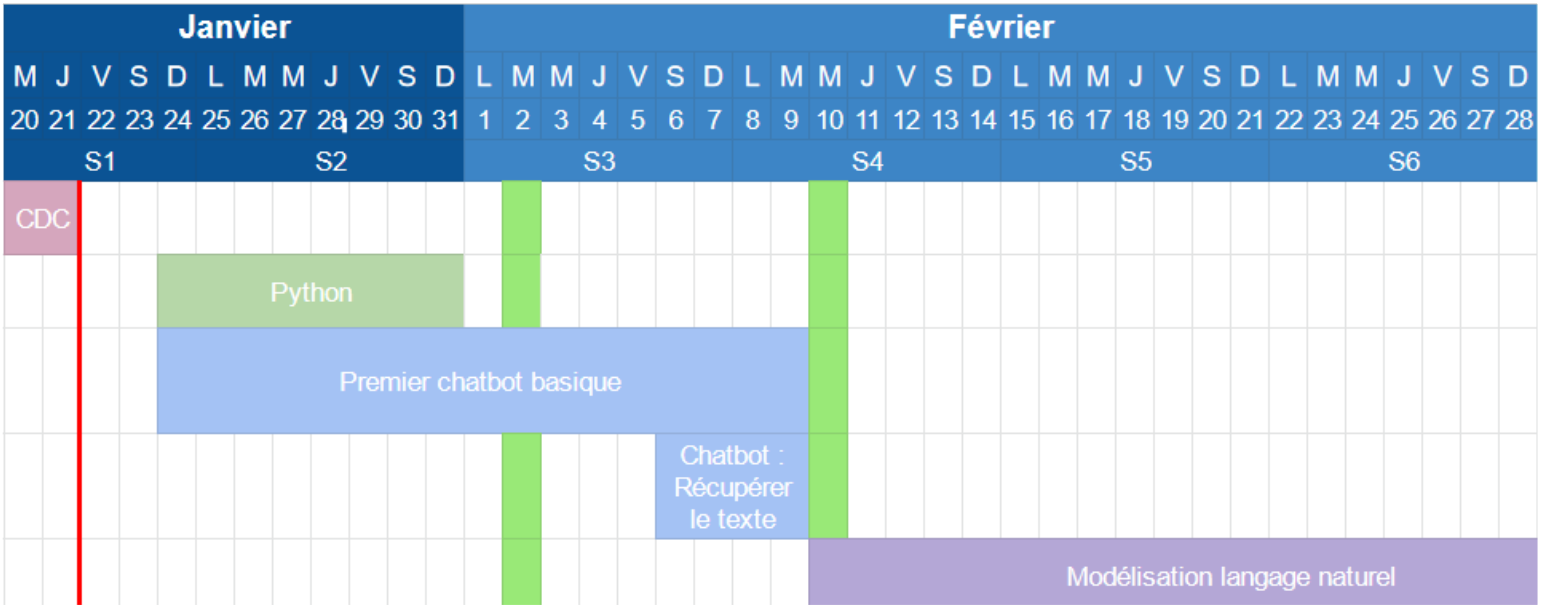



*Figure 3. Planning Janvier-Février*

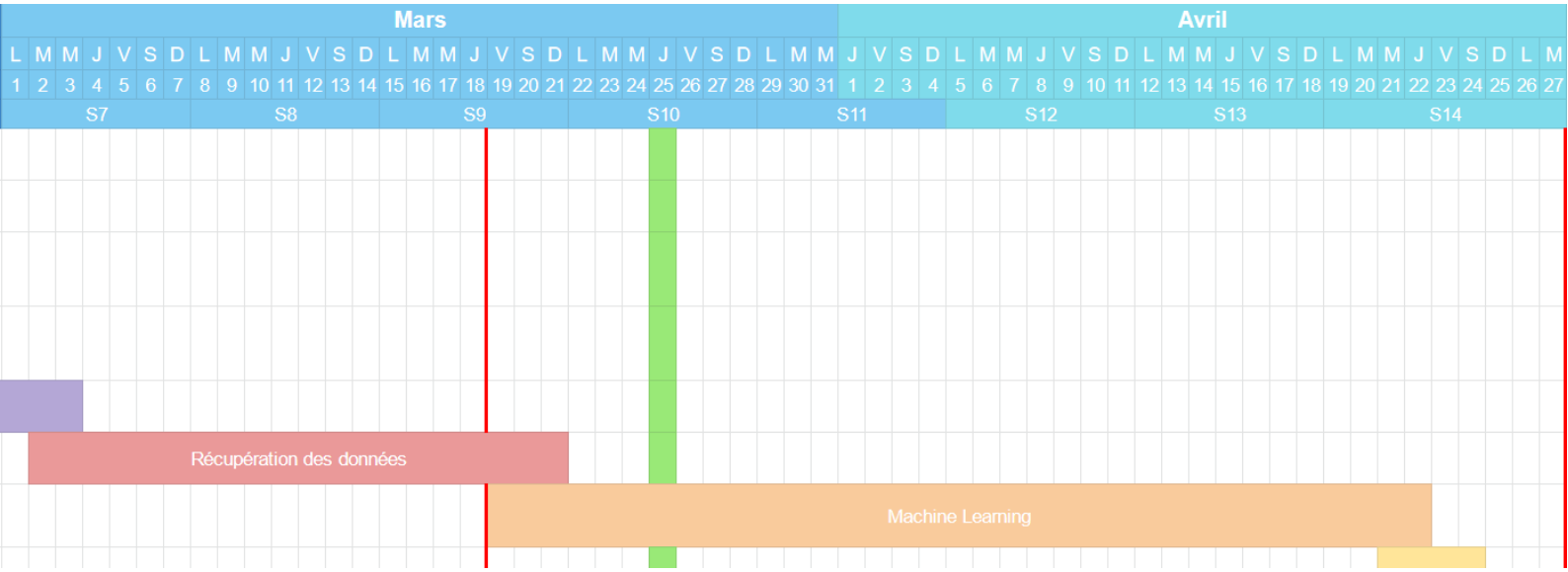



*Figure 4. Planning Mars-Avril*

## 2. Démarche

Comme on peut le voir sur les plannings ci-dessus, le projet a été découpé en 7 taches résumées dans ce tableau :

| N° | Taches | Date de début | Date de fin |
|---|---|---|---|
| 1 | Rédaction cahier des charges | 20/1/21 | 21/1/2021 |
| 2 | Prise en main langage python | 24/1/2021 | 31/1/2021 |
| 3.1 | Développement premier prototype fonctionnel | 24/1/2021 | 9/2/2021 |
| 3.2 | Chatbot : récupère le texte écrit par l'utilisateur | 6/2/2021 | 9/2/2021 |
| 4 | Modélisation du langage naturel | 10/2/2021 | 3/3/2021 |
| 5 | Récupération de données | 1/3/2021 | 21/3/2021 |
| 6 | Machine Learning | 19/3/21 | 22/4/2021 |
| 7 | Interface Graphique | 21/4/21 | 25/4/21 |

*Figure 5. Tâches et dates de début et fin*

### Rédaction du cahier des charges

La tâche de rédaction du cahier des charges a duré 2 jours. Comme son nom l'indique, elle a consisté en la rédaction du cahier des charges afin de définir des objectifs sous forme d'exigences, ainsi que de contraintes.

### Prise en main de Python

Pendant la première semaine de travail, j'ai mis en place l'environnement de travail en installant les modules nécessaires, en créant le dépôt Git du projet et j'ai aussi suivi un tutoriel pour me familiariser avec le langage Python.

### Développement du premier prototype de chatbot

Cette phase du projet a duré un peu plus de 2 semaines. Durant celle-ci et dans un premier temps, j'ai fait des recherches concernant des chatbot existants et j'ai implémenté celui qui me paraissait le plus adapté : ELIZA.
Je l'ai ensuite amélioré et adapté à mes besoins en faisant différents tests de dialogue :

- Utilisation d'un module de traduction pour avoir un dialogue en français

- Rectification de certaines phrases
- Ajout de mots-clés (ex : dead, die Pour qu'ELIZA réponde certaines phrases en fonction de ce mot-clé)

Durant cette phase, j'ai également implémenté une interface graphique basique.
Enfin j'ai fait en sorte d'avoir un chatbot fonctionnel, qui récupère le texte de l'utilisateur dans un fichier txt.

## Modélisation du langage naturel

Pour pouvoir utiliser des algorithmes de machine learning, il faut trouver un moyen de transformer des variables qualitatives en variables quantitatives. Pendant ces trois semaines, je me suis renseignée sur les meilleures manières de vectoriser du texte dans le cadre de la détection des émotions. J'ai finalement utilisé le module spaCy et sa fonction nlp qui prend en entrée une chaine de caractère de n'importe quelle longueur et renvoie un vecteur de longueur 300 ( par le biais d'un objet doc qui contient un attribut vector).

## Récupération de données

Initialement, j'avais prévu de récupérer les données moi-même en faisant passer un questionnaire de 118 questions à des personnes qui auraient parlé avec mon chatbot. Cependant, cette méthode n'était pas réalisable en pratique, car pour faire un algorithme de machine learning efficace, il faut généralement des centaines de données.
Pendant un peu plus de deux semaines, j'ai donc cherché une base de données sur laquelle je pourrais entrainer un algorithme de machine learning pour détecter les émotions. J'ai choisi d'utiliser la base de données ISEAR qui rassemble 7468 phrases avec leur émotion associée, recueillies par des psychologues un peu partout dans le monde. Il y a ici 7 émotions différentes : la colère, le dégout, la peur, la culpabilité, la joie, la tristesse et la honte.
Une fois la base de données récupérée, je l'ai nettoyée et mise au format CSV afin de faciliter l'exploitation de celle-ci dans l'étape d'après.

## Machine Learning

Cette dernière phase la plus importante m'a pris 5 semaines. Dans un premier temps, j'ai fait des recherches sur les différents algorithmes de machine learning existants. J'ai finalement utilisé les cours d'Apprentissage Automatique de ce semestre, et je me suis beaucoup aidé du dernier TP de machine learning concernant l'apprentissage supervisé.
Ma problématique majeure a été le nombre considérable de données à gérer : 300 valeurs pour chacune des 7468 phrases.
J'ai finalement réussi en écrivant 2 algorithmes séparément. Le premier qui transforme toutes les phrases du fichier csv par leur vecteur en utilisant la méthode NLP de spaCy (après avoir normalisé les phrases en enlevant les guillemets de certaines phrases qui en contenait).

Le deuxième qui utilise du machine learning sur le nouveau fichier csv. Il est composé de cinq parties :

- Chargement des données
- Nettoyage et normalisation des données
- Séparation des données (entrainement/tests)
- Implémentation du modèle
- Entrainement du modèle

Une fois cet algorithme écrit, j'ai pu tester différents modèles et différentes proportions pour la séparation données entrainement / données de test.
Ensuite j'ai utilisé le module joblib pour exporter le modèle choisi et l'utiliser dans le programme du chatbot.

### Interface graphique

Pour finir, j'ai complété le code principal en ajoutant la partie d'analyse des émotions de l'utilisateur une fois que le mot « DONE » est entré par ce dernier. J'ai également ajusté l'interface graphique (couleurs, affichage des résultats).

## 3. Réunions Tuteur

Tout au long de ce projet, j'avais la possibilité d'échanger avec Théodore Letouzé en cas de besoin. La première réunion a eu lieu le 02/02/2021 pour valider les objectifs du projet et partir dans la bonne direction.
Le deuxième échange a eu lieu le 10 février et concernait le chatbot ELIZA car j'avais besoin d'une validation de ce dernier.
Enfin la troisième réunion du 25 mars m'a permis de lui faire un retour concernant mon avancement du projet, notamment la modification de la démarche initiale au niveau de la récupération de données. Ensuite, je lui ai montré mon algorithme de vectorisation utilisant le module spaCy pour avoir son avis. Il m'a alors conseillé sur la démarche à suivre pour la construction de l'algorithme de machine learning.

# III - Application

## 1. Architecture

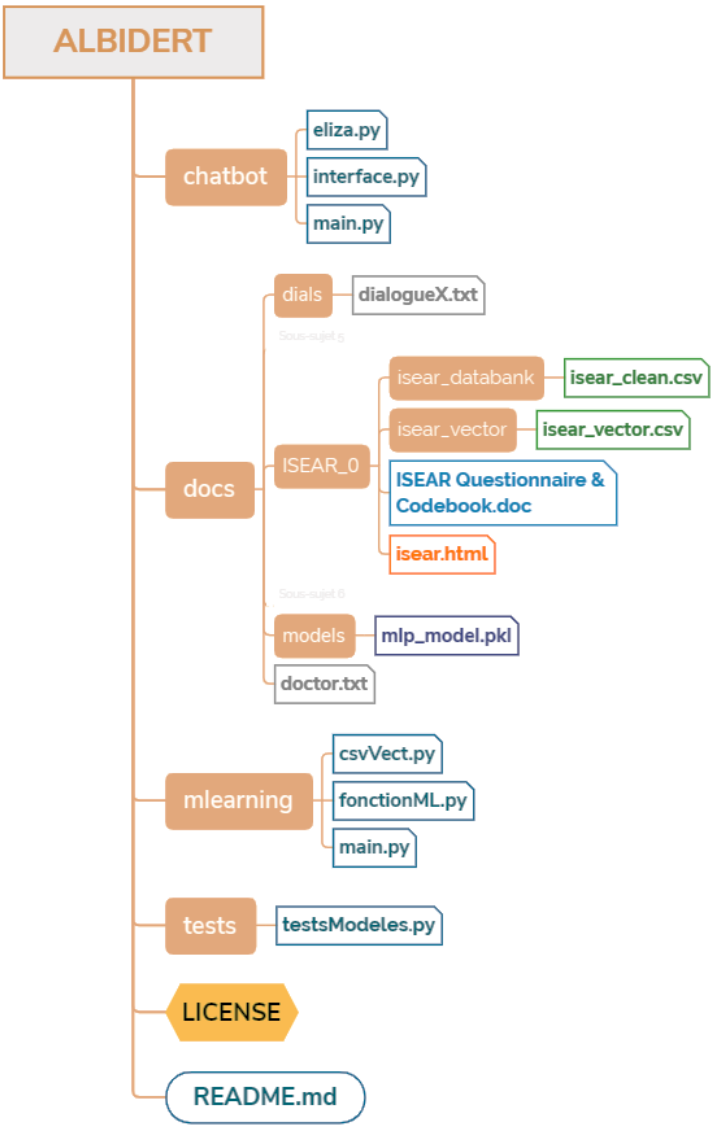



*Figure 6. Arborescence simplifiée de l'application*

L'application est composée de 4 dossiers essentiels.
Tout d'abord le dossier chatbot, où comme son nom l'indique, on y retrouve tous les fichiers en rapports avec le chatbot :

- main.py le programme principal
- interface.py
- eliza.py

Ensuite, le dossier docs regroupe tous les documents utilisés par les différents programmes python :

- dials avec les dialogues utilisateurs
- ISEAR_0 contient les fichiers relatifs à la base de données ISEAR
- models regroupe les modèles de machine learning générés par le programme main.py du dossier mlearning.

Le dossier mlearning justement, s'occupe de la partie machine learning avec :

- main.py le programme principal
- csvVect.py le programme qui transforme un fichier csv de texte en fichier csv de vecteurs
- fonctionML.py qui génère le modèle de machine learning

Le quatrième dossier est le dossier tests qui contient un fichier testsModeles.py qui, comme l'indique son nom, teste quelques paramètres de séparation de données pour l'algorithme d'apprentissage automatique.

## 2. Fonctionnement

Dans cette partie, je détaille le fonctionnement global ainsi que le fonctionnement des composants de l'application. Est inclus dans chaque partie, la justification de mes choix techniques.

### Fonctionnement global

La figure ci-dessous illustre le fonctionnement global de l'application. Pour rappel, le programme main.py de mlearning n'est utile uniquement pour générer ou modifier le modèle mlp_model.pkl. Il faut savoir que l'application est livrée avec un fichier mlp_model.pkl et qu'il n'est donc pas nécessaire d'utiliser le programme mlearning pour le bon fonctionnement de l'application. Le programme principal utilisé par l'utilisateur est bien le programme main.py du dossier chatbot.

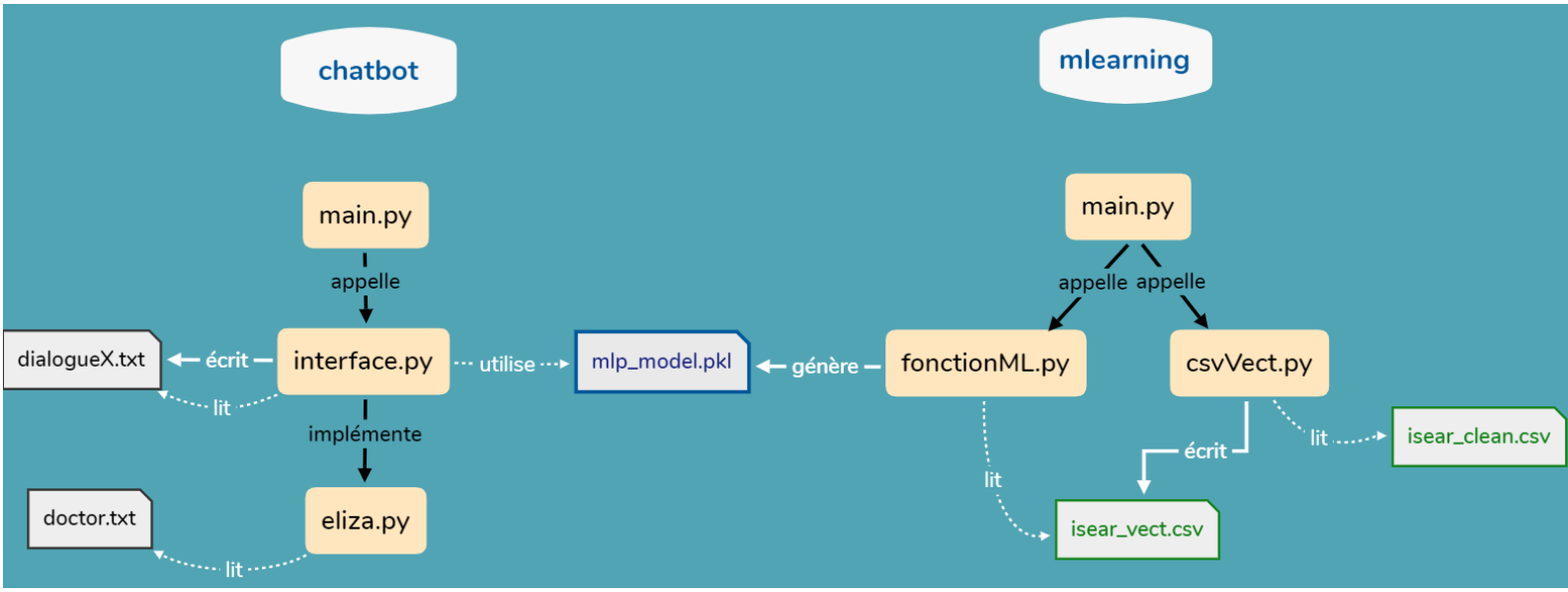



*Figure 7. Fonctionnement de l'application*

### Chatbot

Le programme qui va être utilisé principalement par l'utilisateur est le programme de chatbot.

#### a) Programme principal

Le programme principal contenu dans le fichier main.py se contente d'uniquement créer une interface et d'appeler la boucle principale.

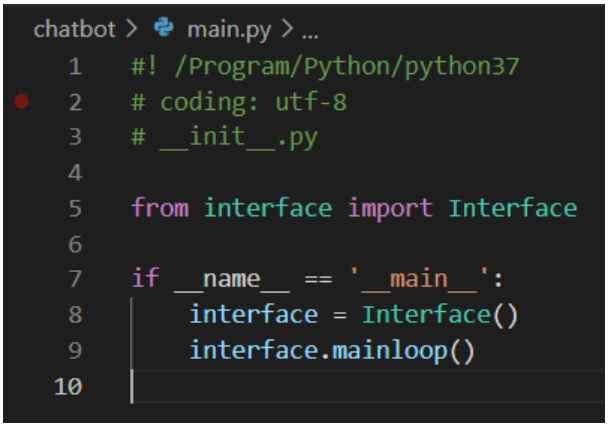



*Figure 8. Programme principal du chatbot : main.py*

### b) Interface graphique

#### Choix technique

J'ai choisi de faire l'interface graphique grâce au module TkInter pour sa facilité de prise en main tout en laissant la possibilité de faire une application ergonomique.

#### Présentation du module utilisé

TkInter (Tool kit interface) permet la création d'interfaces graphiques. C'est la « bibliothèque graphique libre d'origine pour le langage Python » Wikipédia.

#### Fonctionnement de interface.py

##### *Initialisation*

La première partie de ce programme consiste en l'initialisation des composants graphiques. L'interface est composée de :

- Une fenêtre principale (815x650)
- La fenêtre de chat qui va afficher le dialogue ainsi que les résultats de l'analyse des émotions
- Une « scroll-bar » sur le coté
- Un espace où l'utilisateur va taper son message
- Un bouton « Envoyer » pour envoyer son message (Il est également possible de faire cette action avec la touche « Entrée » du clavier)

Quand l'utilisateur envoie son message, le programme fait appel à la fonction get_response.

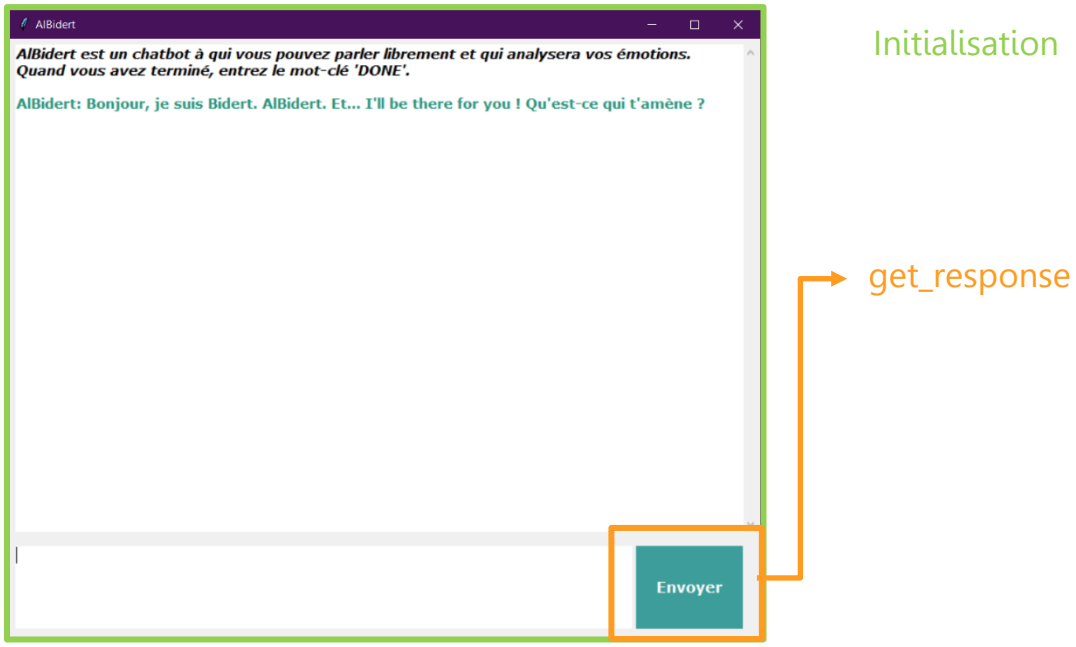



*Figure 9. Illustration du fonctionnement de interface.py*

### *Fonction get_response*

Dans cette fonction, après avoir vérifié que le message n'était pas vide pour éviter toute erreur indésirée, deux cas sont possibles :

**1er cas** : Le message de l'utilisateur est différent de « DONE » (ou « done »). Ici, l'utilisateur converse avec le chatbot. On utilise donc la fonction write_in_file(path,string) afin de recopier le message envoyé par l'utilisateur dans un fichier texte (dialogueX.txt)
Ensuite, on génère la réponse du chatbot par la fonction chatbot.respond(msg) qui va prendre en entrée le message de l'utilisateur, préalablement traduit en anglais par le module Google Traduction car ELIZA est codé en anglais pour de l'anglais. La fonction utilisée est définie dans le fichier eliza.py.
Il ne reste plus qu'à traduire la réponse de ELIZA de l'anglais vers le français pour l'afficher à l'utilisateur et revenir au début de la boucle.

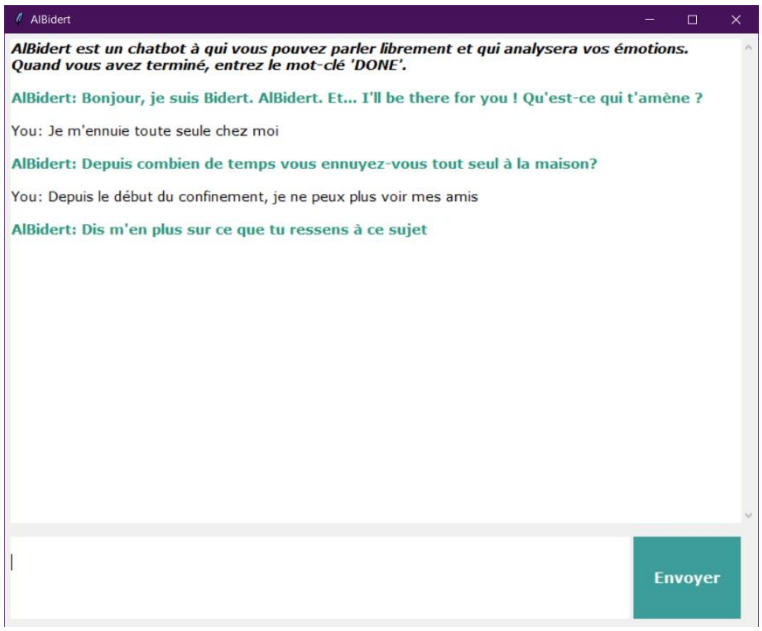



*Figure 10. Illustration du cas 1 : Dialogue avec l'utilisateur*

**2ème cas** : Le message de l'utilisateur est « DONE » (ou « done »). Cette fois-ci, l'utilisateur a fini de parler au chatbot et souhaite une analyse de ses émotions.
On récupère alors le contenu du fichier avec le texte de l'utilisateur par la fonction f.read()
On transforme le texte en vecteur et on importe le modèle de machine learning se trouvant dans le dossier models, grâce à la fonction joblib.
On peut alors faire la prédiction avec model.predict_proba(vector) pour avoir les prédictions du modèles concernant les émotions ressenties par l'utilisateur.
Il n'y a plus qu'à les afficher. Voici un exemple de ce qu'on peut obtenir.

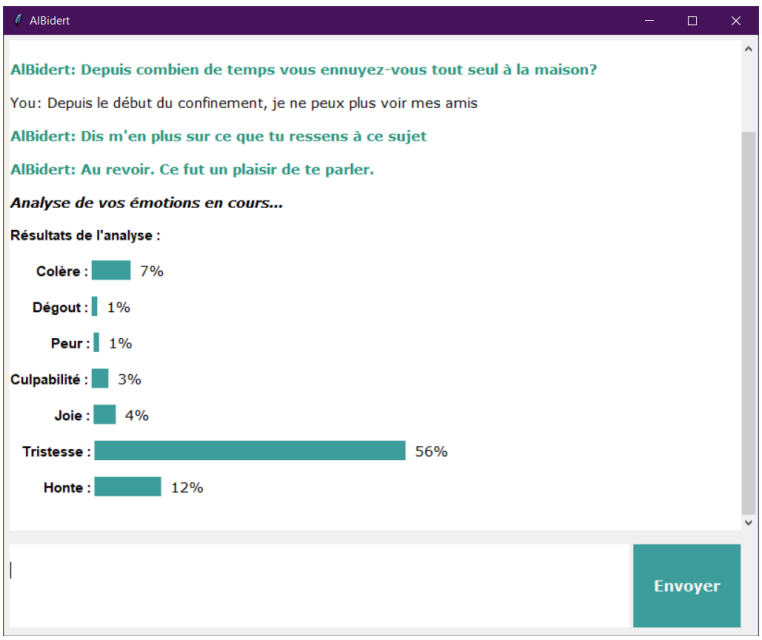



*Figure 11. Illustration du cas 2 : affichage de l'analyse des émotions*

### c) Algorithme : ELIZA

#### Choix technique

Le choix de l'algorithme du chatbot a été la première chose sur laquelle je me suis penchée. En effet, ce choix est essentiel pour l'application et il existe de nombreuses variantes de chatbot. Dans mon cas, le but était d'avoir un chatbot semblable à un psychologue, qui relance l'utilisateur pour le faire parler afin d'avoir un maximum de texte venant de lui pour l'analyser par la suite. De nombreux chatbot utilisent de l'Intelligence Artificielle dans leur algorithme dans le but de les rendre plus « humain » mais cette caractéristique n'était pas importante pour moi, ou du moins n'était pas une caractéristique principale. Je me suis donc tournée vers le chatbot ELIZA.

#### Présentation du chatbot utilisé

ELIZA est un algorithme écrit entre 1964 et 1966 par Joseph Weizenbaum qui reformule la plupart des affirmations de l'utilisateur sous forme de question pour simuler un psychologue rogérien. C'est l'un des premiers chatbot et l'un des premiers à avoir passé le test de Turing. (*ELIZA - Wikipedia*, s. d.)

#### Fonctionnement de eliza.py

ELIZA est un chatbot qui fonctionne par reconnaissance de mots-clés. Après avoir découper le texte en phrases, toutes les phrases sont analysées une par une. Des mots-clés sont présents dans le fichier doctor.txt. Le programme va les parcourir un par un et voir s'il y a une correspondance.
S'il y en a une, le programme va chercher une décomposition qui correspond.
Enfin, il ne reste plus qu'à choisir aléatoirement une réponse dans la liste correspondante.
Dans le cas où aucun mot clés n'est trouvé, ELIZA affiche une réponse par défaut.
Si plusieurs mots clés sont trouvés, c'est celui qui a le poids le plus important qui est pris en compte.

MSG : « You help me »

docs > doctor.txt

```
195   key: you
196     decomp: * you remind me of *
197       reasmb: goto alike
198     decomp: * you are *
199       reasmb: What makes you think I am (2) ?
200       reasmb: Does it please you to believe I am (2) ?
201       reasmb: Do you sometimes wish you were (2) ?
202       reasmb: Perhaps you would like to be (2) .
203     decomp: * you * me *
204       reasmb: Why do you think I (2) you ?
205       reasmb: you like to think I (2) you -- don't you ?
206       reasmb: What makes you think I (2) you ?
207       reasmb: Really, I (2) you ?
208       reasmb: Do you wish to believe I (2) you ?
209       reasmb: Suppose I did (2) you -- what would that mean ?
210       reasmb: Does someone else believe I (2) you ?
211     decomp: * you *
212       reasmb: We were discussing about you -- not me.
```

ELIZA : « What makes you think I help you ? »

*Figure 12. Illustration du fonctionnement de ELIZA*

## Algorithme de détection d'émotions

Le deuxième programme concerne la détection d'émotions et est contenu dans le fichier mlearning. Il a pour but de générer le fichier mlp_model.pkl qui est utilisé pour la prédiction des émotions dans le programme du chatbot. Pour cela, il y a une première étape de vectorisation du texte, puis une deuxième étape de création d'un modèle qu'on entraine sur cette base de données.

### a) Base de données utilisée

La base de données sur laquelle je me base est ISEAR (International Survey on Emotion Antecedents and Reactions). Les données de ISEAR ont été collectées pendant les années 1990 par un grand groupe de psychologues à travers le monde. Ce projet a été dirigé par Klaus R. Scherer et Herald Wallbott. Les étudiants interrogés, qui étaient en étude de psychologie ou non, ont dû décrire des situations pendant lesquelles ils avaient ressenti les 7 émotions majeures (joie, peur, colère, tristesse, dégout, honte et culpabilité). La base de données finale contient des descriptions de situation correspondant à 7 émotions avec chacune environ 3000 personnes interrogées venant de 37 pays différents et des 5 continents. (*Papers with Code - ISEAR Dataset*, s. d.)

docs > ISEAR_0 > isear_databank > isear_clean.csv

```
Emotion,Text
anger,"When I was driving home after  several days of hard work, there was a motorist ahead of me who was driving at 50 km/hour and refused, despite his low speeed to let me overtake."
disgust,"The time I knocked a deer down - the sight of the animal's injuries and helplessness.  The realization that the animal was so badly hurt that it had to be put down, and when the animal screamed at the moment of death."
fear,When I was involved in a traffic accident.
guilt,When I caused problems for somebody because he could not keep the appointed time and this led to various consequences.
joy,"During the period of falling in love, each time that we met and especially when we had not met for a long time."
sadness,When I lost the person who meant the most to me.
```

*Figure 13. Extrait de la base de données ISEAR*

b) Programme principal

Le programme main.py du dossier mlearning est composé de trois parties. La première nettoie certaines phrases de la base de données en enlevant les guillemets de début et fin de phrase. La suivante vectorise le texte des données contenues dans isear_clean.csv et la troisième partie qui créée et exporte le modèle de machine learning.

c) Programme de vectorisation de texte

Le fichier csvVect.py contient la définition de deux fonctions : Une qui enlève simplement les guillemets de début et fin de phrase quand il y en a. L'autre fonction prend en entrée deux paramètres :

- le chemin du fichier csv avec l'émotion et le texte
- le chemin où sera créé le fichier csv avec l'émotion et le vecteur associé à l'ancien texte.

Pour vectoriser le texte, j'utilise le module NLP de spaCy. Le but de cette fonction est de vectoriser le texte de la base de données ISEAR pour pouvoir faire tourner un algorithme d'apprentissage automatique supervisé par la suite.

Choix technique

J'ai choisi le module nlp de spaCy après avoir fait certaines recherches sur le sujet. En effet, c'est un algorithme très complet, qui utilise d'autres algorithmes de réseaux de neurones récurrents (comme wordToVec) et qui prend en compte le sens des mots pour vectoriser un texte.

Présentation du module utilisé

SpaCy est une librairie open-source et gratuite pour le Traitement du Langage Naturel Avancé (TLNA) en Python.(*SpaCy 101*, s. d.)

## Fonctionnement de csvVect.py

La fonction csvVectToCsvText définie dans le fichier csvVect.py commence par charger le modèle nlp de spacy. Après avoir supprimé s'il existait déjà le fichier de sortie, on ouvre le fichier d'entrée en lecture et le fichier de sortie en écriture (cela va donc créer un nouveau fichier). Après avoir recopier mot à mot la première ligne (« Emotion,Text »), on applique ligne par ligne la fonction nlp de spaCy à la partie « Text » de la ligne.

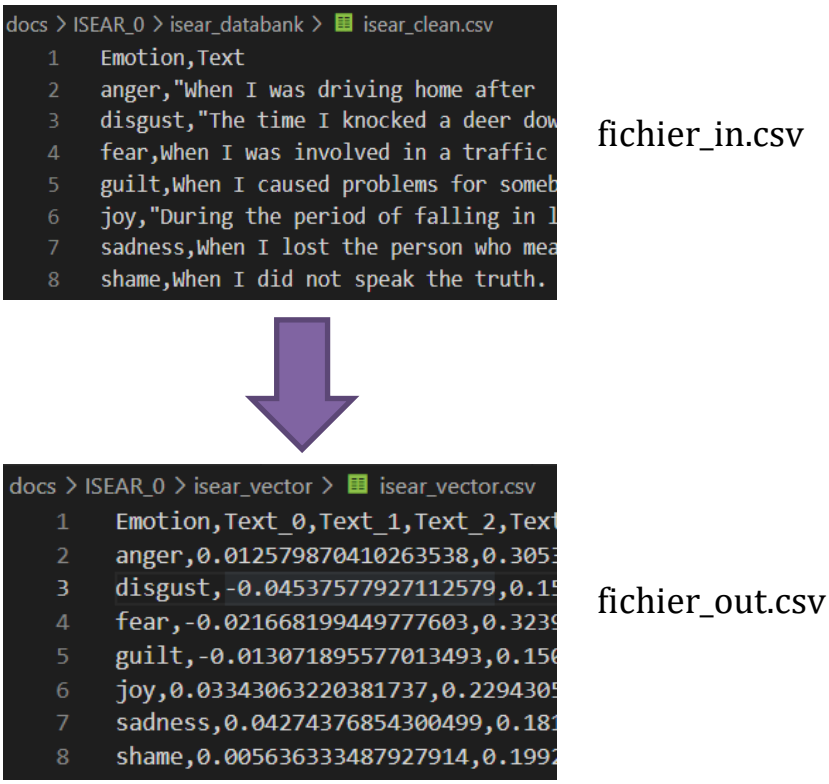



*Figure 14. Illustration du fonctionnement de CsvVect.py*

## Fonctionnement NLP de spaCy

### *Fonctionnement général*

La fonction nlp (Natural Language Processing) de spaCy transforme un texte (chaine de caractères) en un objet de type Doc. Ce dernier possède un attribut vector que j'utilise pour vectoriser les phrases de la base de données. La fonction nlp passe par plusieurs étapes avant pour passer d'un texte à un Doc dont les plus importantes sont celles de tokenisation, étiquetage, analyse de dépendances et reconnaissance d'entités nommées.(*SpaCy 101*, s. d.)

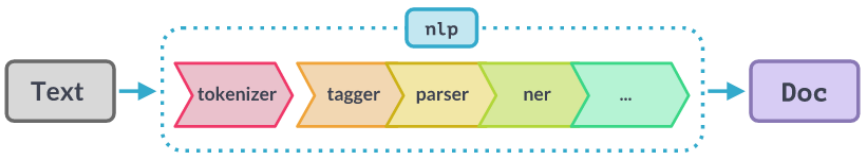



*Figure 15. Schématisation de la fonction nlp de spaCy*

### *Pipeline model*

Toutes les étapes sauf la tokenisation utilisent un modèle entrainé de réseaux de neurones appelé « pipeline ». Ce modèle est entrainé de façon itérative où ses prédictions sont comparées aux annotations référentes pour estimer le gradient de perte. Ce gradient de perte est ensuite utilisé pour calculer les poids, grâce à de la « backpropagation ». Le gradient indique comment la valeur des poids doivent être changés pour que la prédiction du modèle soit plus similaire aux valeurs de références.(*SpaCy 101*, s. d.)

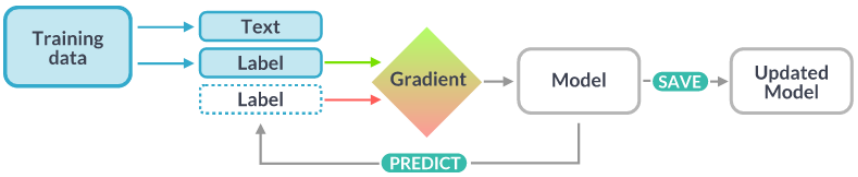



*Figure 16. Schématisation du fonctionnement du pipeline model*

### *Définition des étapes principales du nlp*

Tokenisation : C'est la séparation d'une phrase en mots (ou « tokens »).

Etiquetage : On assigne chaque mot à un type, comme verbe ou nom.

Analyse de dépendances : C'est l'attribution d'étiquettes de dépendance syntaxique qui décrive les relations entre les « tokens » comme sujet ou objet.

Reconnaissance d'entités nommées : C'est la labellisation d'objets du « monde réel » comme personnes, entreprises, lieux.(*SpaCy 101*, s. d.)

### d) Algorithme de machine-learning

La deuxième partie de ce programme concerna la création et l'exportation du modèle de machine learning utilisé pour la prédiction des émotions. Pour cela, l'algorithme à besoin d'apprendre l'association entre un vecteur de longueur 300 (représentant un texte ou une phrase) et une émotion.
La fonction contenue dans le fichier fonctionML.py prend en entrée un fichier csv contenant l'émotion associée à un vecteur de 300 et le chemin vers l'endroit où sera exporté le modèle construit.

#### Choix technique

J'ai décidé d'utiliser l'algorithme MLPClassifier de scikit-learn. En effet, après avoir fait quelques tests avec les algorithmes de Régression Linéaire et d'arbre de décision, l'algorithme MLP à eu le score le plus élevé et me semblait plus cohérent concernant l'analyse des émotions. De plus, c'est un algorithme que j'ai étudié dans le cadre du module Apprentissage Automatique ce semestre.

### Présentation du module utilisé

Scikit-learn (formellement scikits.learn et aussi connu sous le nom de sklearn) est une librairie gratuite et libre de Python destinée à l'apprentissage automatique. Elle offre de nombreux algorithmes de classification, de régression, de regroupent et qui a été conçu pour interagir avec les librairies scientifiques et numériques de Python : NumPy et SciPy.(« Scikit-Learn », 2021)

### Fonctionnement de fonctionML.py

La fonction principale de ce fichier, qui est appelée par le programme main est la fonction exportMlpLearning qui prend en entrée un fichier csv vectorisé ainsi que le chemin menant à l'endroit où le modèle sera exporté.

### Présentation de MLPClassifier

MLPClassifier signifie Multi-Layer Perceptron (MLP) classifier soit en français : Perceptron multicouches classificateur. Ce modèle s'appuie sur un réseau de neurones pour effectuer la tâche de classification. (Nair, 2019)

### Fonctionnement d'un MLP

Le fonctionnement d'un MLP est similaire à celui d'un perceptron 1 couche. C'est pourquoi j'explique ici le fonctionnement d'un perceptron à couche unique.

#### *Perceptron 1 couche*

C'est un classificateur linéaire dont toutes les entrées sont connectées à la sortie qui est unique et booléenne.
Dans un premier temps, toutes les entrées sont multipliées par leurs poids et sont ensuite ajoutées entre-elles pour créer une somme pondérée. Sur cette somme, on applique ensuite la fonction d'activation qui produit la sortie du perceptron. La fonction d'activation agit de la sorte :

- Si la somme dépasse un certain seuil, la sortie vaudra 1
- Sinon la sortie vaudra 0.

(*Perceptron*, 2019)

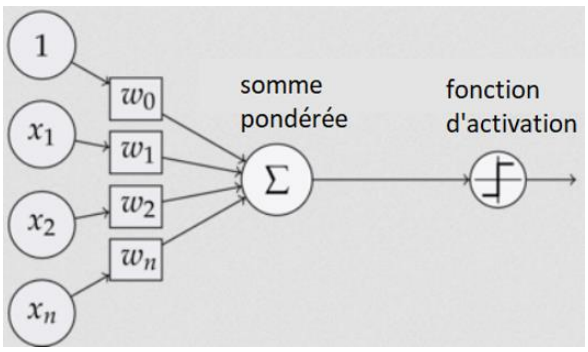



*Figure 17. Schématisation de l'algorithme du perceptron*

### *Perceptron multicouche*

Similairement au perceptron à couche unique, l'information va circuler de la couche d'entrée vers la couche de sortie uniquement et de façon directe (feedforward). Il est composé d'une couche d'entrée, d'une couche de sortie et toutes les couches entre les deux sont dites couches cachées (au moins une). Ici, chaque neurone agit comme un perceptron 1 seule couche et prend comme entrée toutes les sorties des neurones précédents, et par conséquent sa sortie à lui va être donnée en entrée aux neurones de la couche suivante.
La stratégie de modification des poids synaptiques est plus complexe car elle dépend du gradient de l'erreur (pour un poids donné, si la dérivée de l'erreur est positive, il faut baisser le poids, sinon il faut l'augmenter)
La fonction d'activation est dans ce cas la fonction sigmoïde. (Salotti, s. d.)

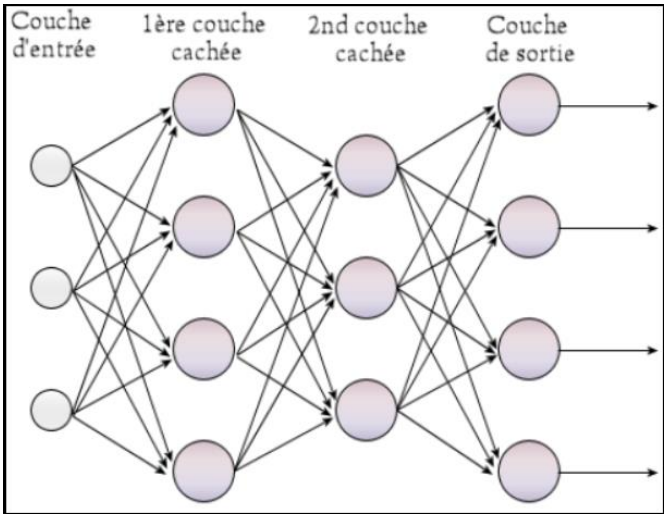



*Figure 18. Illustration d'un MLP*

## Tests

Dans le code source de l'application figure un dossier tests contenant un fichier testsModeles.py. Comme son nom l'indique, ce fichier un peu brouillon m'a servi tout au long du projet pour tester tout type de fonction. Il y figure des tests concernant le modèle MLPClassifier avec des ratios différents de répartition entre les données de tests et données d'entrainement.

# IV – Pistes d'évolution

## 1. Techniques

### Chatbot

#### a) ELIZA en français

Le chatbot que j'ai récupéré, ELIZA, est codé en anglais, et a été conçu pour la langue anglaise au niveau de la construction des phrases notamment. Pour l'utiliser en français, j'utilise le module Google Traduction pour traduire la phrase entrée par l'utilisateur de français vers anglais. Ce message traduit en anglais est donné en entrée du programme eliza.py qui renvoie la réponse du chatbot en anglais, que je traduis en français pour l'afficher et répondre à l'utilisateur.
Cela peut induire des erreurs de traductions et des petites incohérences, notamment pour des expressions (j'ai corrigé les erreurs que j'ai rencontré comme « j'en ai marre » était traduit par « I am sick of it » et ELIZA reconnaissait le mot-clé « sick » seul et pensait qu'il était question de maladie mais il doit rester d'autres erreurs de ce style).
Pour améliorer cela, on pourrait recoder ELIZA en français directement, en suivant le même principe de mots-clés et de découpage de phrase.

#### b) Ajouter une IA

Comme de nombreux chatbot existants, on pourrait ajouter un algorithme d'intelligence artificielle qui paraitrait plus humain, et qui pourrait faire des blagues ou au moins comprendre les expressions de langage.

#### c) Interface : site-web ou application mobile

Comme je l'avais spécifié dans les exigences secondaires, on pourrait penser à mettre ce chatbot en ligne, sous-forme de site web voire d'application mobile pour le rendre accessible publiquement.

### Détection d'émotion

#### a) Ajouter des bases de données

Il serait intéressant d'entrainer le modèle de détection d'émotion sur d'autres bases de données afin d'avoir d'autres émotions à détecter que les 7 déjà présentes, qui sont surtout des émotions négatives (colère, dégout, peur, culpabilité, joie, tristesse, honte).

b) Algorithme de machine-learning

*Réduction de la taille du vecteur*

Actuellement, l'algorithme prend en entrée les 300 composantes du vecteur et l'associe à une émotion. On pourrait faire un pré-traitement sur ces 300 valeurs en appliquant une PCA ou une autre méthode afin de ne garder uniquement les valeurs les plus significatives.

*Autres algorithmes et paramètres*

On pourrait penser aussi à tester d'autres algorithmes que MLP. J'ai testé les algorithmes de régression linéaire et d'arbre de décision (comme vu en TP) mais il en existe d'autres. Aussi, MLPCLassifier que j'utilise, possède de nombreux paramètres modifiables dont on pourrait tester les combinaisons afin d'avoir les meilleurs résultats.

### Tests utilisateurs

Faire des tests utilisateurs serait une bonne idée afin de mettre en évidence les problèmes techniques ou avoir des idées d'améliorations intéressantes

## 2. Fonctionnelles

### Activité

Comme indiqué dans mes exigences secondaires dès le début du projet, AlBidert pourrait proposer une activité à l'utilisateur en fonction du résultat de son analyse de son émotion.

### Résultats automatiques

On pourrait aussi imaginer que les résultats des émotions de l'utilisateurs s'afficheraient automatiquement sans qu'il ait à rentrer le mot-clé « DONE ». Par exemple, quand une émotion dépasse un certain seuil.

### Validation des résultats

Aussi, une fonctionnalité intéressante serait de demander la confirmation à l'utilisateur concernant ses émotions pour faire en sorte qu'AlBidert apprenne continuellement.

### Choix de la langue

On pourrait penser à faire choisir la langue à l'utilisateur au début, avec le module que j'utilise actuellement de Google Traduction, c'est une fonctionnalité qui s'ajoute facilement mais n'enlèverai rien au problème cité plus haut d'imprécision de la traduction.

### Analyse de la personnalité

Également présente dans les fonctionnalités secondaires de mon cahier des charges, analyser la personnalité de l'utilisateur pour avoir une détection d'émotion personnalisée est envisageable et serait intéressant à mettre en place.

### Compte et historique des conversations

Cette fonctionnalité découle de la précédente : avoir un compte et faire en sorte qu'AlBidert se rappelle d'anciennes conversations et qu'il puisse récupérer les informations de l'utilisateur auquel il parle serait une bonne idée de fonctionnalité à développer.

### Explication de la raison du résultat final

Enfin, une fonctionnalité répondant à une problématique qui concerne l'algorithme de machine learning serait qu'AlBidert explique les raisons des pourcentages des différentes émotions en déterminant quel mot ou phrase a le plus influencé le choix final. C'est une fonctionnalité plus intéressante au niveau apprentissage automatique qu'au niveau de l'utilisateur mais son développement permettrait de bien comprendre le fonctionnement de l'algorithme utilisé.

## V - Conclusion

Ce projet m'a énormément appris, tant au niveau gestion de projet qu'au niveau technique, et m'a permis de monter en compétence sur l'apprentissage automatique et le langage Python. Je compte améliorer et perfectionner ce chatbot car je trouve que ce projet à du potentiel et pourra surement aider des gens à évacuer librement leurs problèmes par écrit, tout en ayant une analyse de leurs émotions ressenties, ce qui peut les aider à en prendre conscience.

# VI - Bibliographie

*ELIZA - Wikipedia*. (s. d.). Consulté 27 avril 2021, à l'adresse https://en.wikipedia.org/wiki/ELIZA

Nair, A. (2019, juin 20). A Beginner's Guide To Scikit-Learn's MLPClassifier. *Analytics India Magazine*. https://analyticsindiamag.com/a-beginners-guide-to-scikit-learns-mlpclassifier/

*Papers with Code—ISEAR Dataset*. (s. d.). Consulté 27 avril 2021, à l'adresse https://paperswithcode.com/dataset/isear

*Perceptron*. (2019, mai 17). DeepAI. https://deepai.org/machine-learning-glossary-and-terms/perceptron

Salotti, J.-M. (s. d.). *Cours perceptron multi-couches*.

Scikit-learn. (2021). In *Wikipedia*. https://en.wikipedia.org/w/index.php?title=Scikit-learn&oldid=1018547386

*spaCy 101 : Everything you need to know · spaCy Usage Documentation*. (s. d.). SpaCy 101: Everything You Need to Know. Consulté 27 avril 2021, à l'adresse https://spacy.io/usage/spacy-101#whats-spacy