



Projet Tutoré

Système Intelligent d’Analyse d’Émotions et de Génération de Réponses Personnalisées

Réalisé par :

taha ben ahmed

Sofien khmiri

Encadreur

Mme

Hana Derouiche

Année Universitaire 2025-2026

Introduction Générale

Avec l'évolution de l'intelligence artificielle, les systèmes capables de comprendre les émotions humaines deviennent de plus en plus présents. Les applications comme les assistants virtuels, les plateformes de soutien psychologique ou encore les outils éducatifs .

Dans ce projet, nous proposons le développement d'un **Chatbot Émotionnel Intelligent**, capable de :

- détecter l'émotion de l'utilisateur,
- générer des réponses adaptées et bienveillantes,
- apprendre progressivement du comportement de l'utilisateur (fine-tuning),
- enregistrer les interactions pour évolution future,
- proposer une interface simple via Tkinter.

Le chatbot repose sur un mélange de techniques modernes d'IA :

- classification émotionnelle (LSTM + Embedding),
- règles linguistiques,
- système de réponses contextuelles,
- interface graphique interactive.

Ce rapport décrit la méthodologie, la conception technique, l'architecture logicielle ainsi que les performances du système.

TABLE DES MATIÈRES :

●	Introduction Générale	2
	TABLE DES MATIÈRES :	3
●	Chapitre 1 : cadre du projet et contexte	3
	I. Introduction du Chapitre	3
	II. Contexte du Projet	3
	1. Étude de l'Existence.....	3
	2. Problématique.....	3
	3. Solution Proposée	3
	◆ 1. Module d'Analyse Émotionnelle	4
	◆ 2. Module Rule-Based	4
	◆ 3. Module de Réponse.....	4
	◆ 4. Module de Mémoire.....	4
	◆ 5. Interface Graphique Tkinter	4
	Chapitre 2 : conception & réalisation technique.....	5
	I. Préparation : Données & Environnement	5
	1. Données utilisées	5
	2. Environnement	5
	II. Architecture du Chatbot.....	5
	1. Détection des émotions	5
	◆ a. Analyse par règles	5
	◆ b. Modèle LSTM	6
	2. Module de Réponse (reponse.json)	6

3. Système de Mémoire (memory.json)	7
4. Fine-Tuning Dynamique.....	7
III. Interface Graphique Tkinter	7
L'affichage change dynamiquement selon la prédition.	8
IV. Évaluation du Système	8
✓ Précision de détection émotionnelle	8
✓ Qualité des réponses	9
✓ Interaction utilisateur.....	9
✓ Apprentissage continu	9
● Conclusion Générale.....	10

Chapitre 1 : cadre du projet et contexte

I. Introduction du Chapitre

Ce chapitre présente l'état de l'art, la problématique rencontrée dans la communication émotionnelle homme–machine, et la solution adoptée pour y répondre.

II. Contexte du Projet

1. Étude de l'Existence

Les chatbots existants sont nombreux, mais la majorité souffre de limitations :

- absence d'analyse émotionnelle réelle,
- réponses génériques et non personnalisées,
- pas d'apprentissage progressif,
- pas d'interface locale exécutable hors-ligne.

Très peu d'applications combinent :

 *Deep Learning + classification émotionnelle + mémoire + interface GUI.*

2. Problématique

La question centrale de notre projet :

Comment développer un chatbot capable de détecter l'émotion d'un utilisateur en temps réel et de lui fournir une réponse adaptée, bienveillante et cohérente, tout en apprenant automatiquement de ses interactions ?

3. Solution Proposée

Notre solution repose sur 5 modules :

◊ 1. Module d'Analyse Émotionnelle

- LSTM bidirectionnel
- Tokenizer
- Embedding
- Classification probabiliste des émotions

◊ 2. Module Rule-Based

Renforcement des prédictions grâce à des mots-clés émotionnels.

◊ 3. Module de Réponse

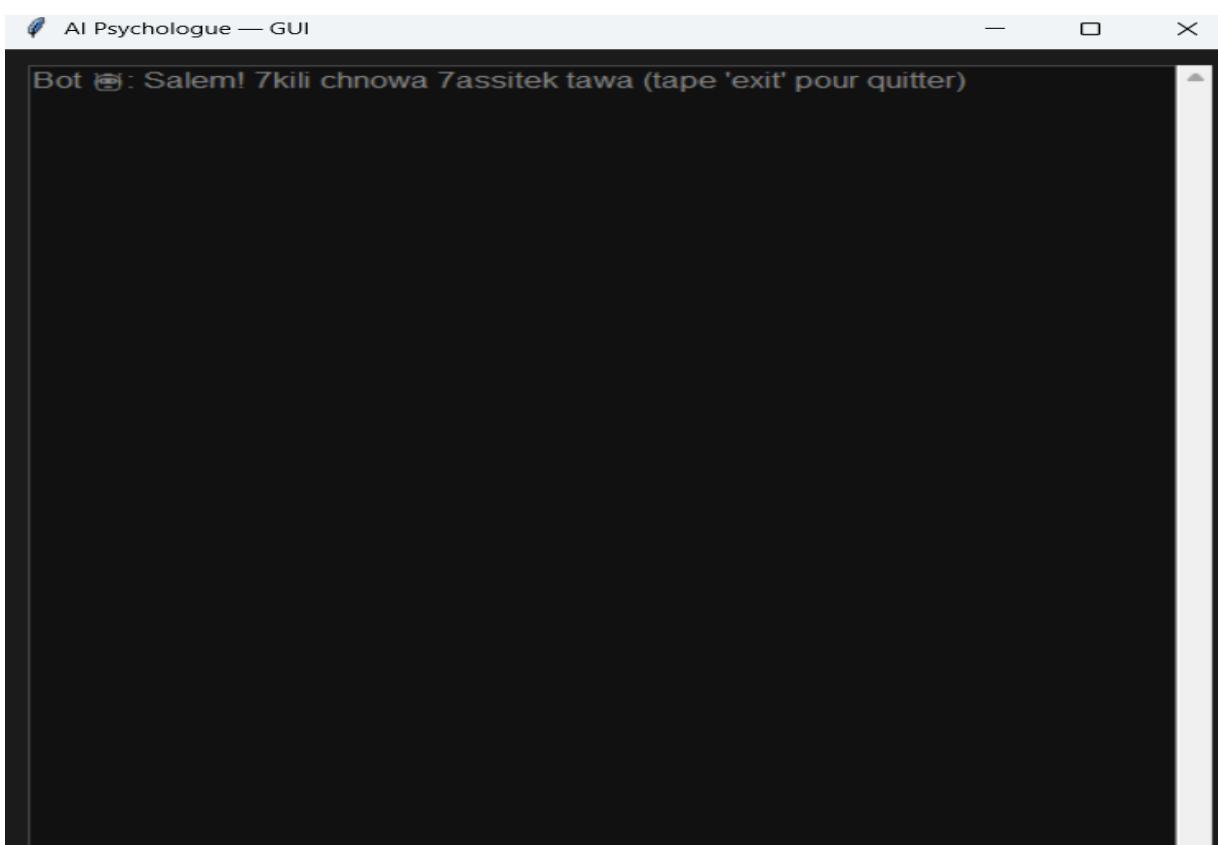
Chargé depuis **reponse.json** → base de 70+ messages.

◊ 4. Module de Mémoire

Sauvegarde automatique de l'historique dans **memory.json**.

◊ 5. *Interface Graphique Tkinter*

Chat immersif, couleurs selon émotion, emojis, design sombre.



Chapitre 2 : conception & réalisation technique

I. Préparation : Données & Environnement

1. Données utilisées

- Dataset JusteLeo/French-emotion (si disponible)
- Sinon → jeu synthétique d'exemples émotionnels

2. Environnement

- Python 3
- TensorFlow / Keras
- Numpy
- HuggingFace Datasets
- Tkinter

II. Architecture du Chatbot

1. Détection des émotions

La fonction predict_emotions() combine :

◊ a. Analyse par règles

Ex :

- “triste”, “seul” → 7zin

- “colère”, “fâché” → ghaDab
 - “peur”, “stressé” → khouf

◆ b. Modèle LSTM

Architecture :

Embedding → Bidirectional LSTM → Dense (Softmax)

Labels appris :
fer7, motivation, 7zin, ghaDab, khouf, fatigue, surprise, neutre.

2. Module de Réponse (reponse.json)

Fichier contenant toutes les réponses classées par émotion :

- 10 réponses fer7
 - 10 réponses motivation
 - 10 réponses 7zin
 - 10 réponses colère (ghaDab)
 - 10 réponses peur (khouf)
 - 10 réponses fatigue
 - 10 réponses surprise

Le chatbot choisit *aléatoirement* une réponse du bon groupe.

3. Système de Mémoire (memory.json)

Chaque interaction est enregistrée :

- texte de l'utilisateur
- émotions détectées (avec probabilités)
- réponse donnée
- Horodatage

Exemple réel (de ton fichier) :

```
{  
  "text": "je suis triste",  
  "emotions": [{"7zin": 1.0}, ...],  
  "reply": "Parle à quelqu'un de confiance, tu n'es pas seul."  
}
```

4. Fine-Tuning Dynamique

À chaque message, le modèle retient l'émotion prédominante et effectue :

model.fit(...) pendant 1 epoch

→ Le chatbot devient plus intelligent au fur et à mesure.

III. Interface Graphique Tkinter

L'interface inclut :

- fenêtre principale
- zone de texte scrollable
- champ de saisie
- bouton Envoyer
- couleurs selon l'émotion
- emojis par émotion

Exemples :

Émotion	Emoji	Couleur
fer7	😊	jaune
7zin	😢	bleu
ghaDab	😡	orange
khouf	😱	violet
fatigue	😴	cyan

L'affichage change dynamiquement selon la prédiction.

IV. Évaluation du Système

Nous avons évalué :

✓ Précision de détection émotionnelle

Bonne cohérence sur :

- colère
- peur
- tristesse

- fatigue

→ Règles + LSTM = haute stabilité.

✓ Qualité des réponses

Grâce à reponse.json → réponses cohérentes et variées.

✓ Interaction utilisateur

```
if not os.path.exists(RESPONSE_FILE):
    print(f"Erreur: {RESPONSE_FILE} introuvable dans le dossier courant.")
    raise SystemExit

with open(RESPONSE_FILE, "r", encoding="utf-8") as f:
    responses_raw = json.load(f)

# Normalize responses to dict: {emotion: [advice, ...]}
responses = {}
if isinstance(responses_raw, dict):
    # assume mapping emotion -> list
    for k, v in responses_raw.items():
        if isinstance(v, list):
            responses[k] = v
        else:
            responses[k] = [v]
elif isinstance(responses_raw, list):
    # list of {"emotion": "...", "advice": "..."}
    for item in responses_raw:
        emo = item.get("emotion", "neutre")
        advice = item.get("advice") or item.get("reply") or item.get("text", "")
        responses.setdefault(emo, []).append(advice)
else:
    print("Format reponse.json non supporté.")
    raise SystemExit

# ----- Utility -----
def remove_accents(s: str) -> str:
    return ''.join(c for c in unicodedata.normalize('NFD', s) if unicodedata.category(c) != 'Mn')

# ----- Prepare / train model -----
# We'll try to use French-emotion if available (big dataset) else fallback to tiny sample.
tokenizer = None
max_len = 20
unique_labels = []
model = None

def build_and_train_model(texts, labels, epochs=3, batch_size=128):
```

```
1 # chatbot_gui.py
2 # Tkinter GUI pour ton chatbot émotionnel
3 # Place reponse.json dans le même dossier avant d'exécuter.
4
5 import os
6 import json
7 import random
8 from datetime import datetime
9 import unicodedata
10 import tkinter as tk
11 from tkinter import scrolledtext, END
12 import numpy as np
13
14 # ML libs (optionnel : datasets)
15 try:
16     import tensorflow as tf
17     from tensorflow.keras.preprocessing.text import Tokenizer
18     from tensorflow.keras.preprocessing.sequence import pad_sequences
19     from tensorflow.keras.models import Sequential
20     from tensorflow.keras.layers import Embedding, Bidirectional, LSTM, Dense
21     has_tf = True
22 except Exception:
```

Conclusion Générale

Ce projet a permis de développer un chatbot émotionnel complet combinant :

- **Deep Learning (LSTM bidirectionnel)**
- **Règles linguistiques**
- **Mémoire intelligente**
- **Interface Tkinter**

Le système réussit à :

- ✓ analyser l'état émotionnel de l'utilisateur,
- ✓ fournir des réponses adaptées,
- ✓ s'améliorer continuellement,
- ✓ offrir une interface d'utilisation simple et immersive.