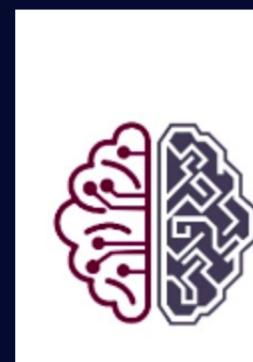
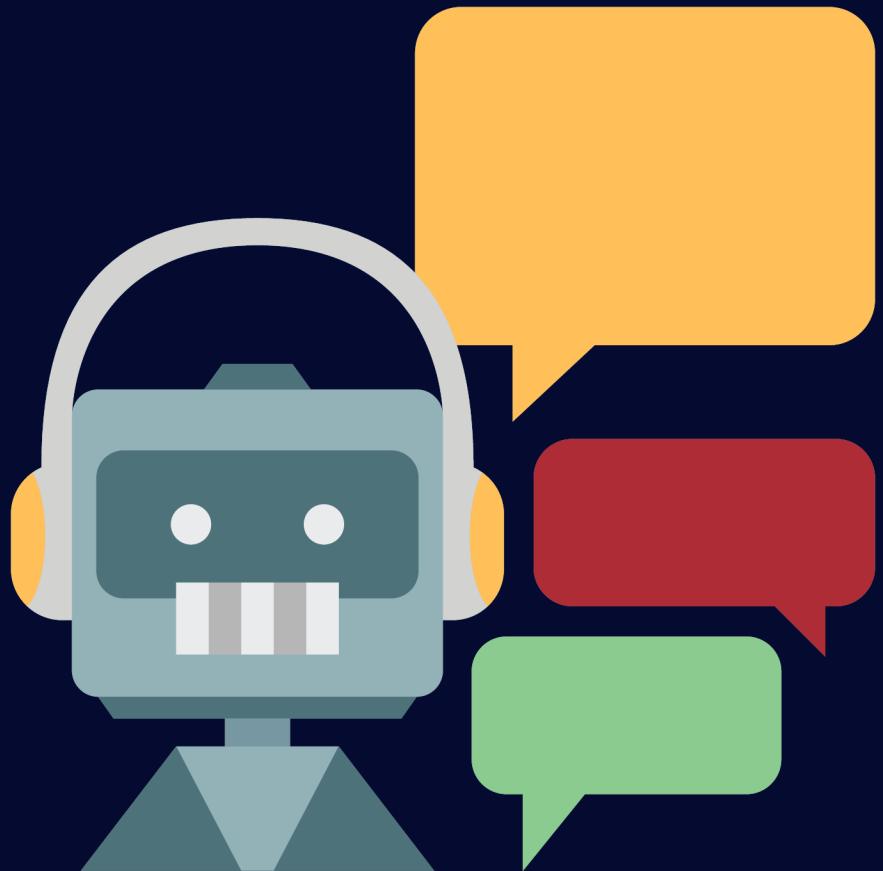




# Assistant Client Intelligent

Projet IA - NLP, Emotion & Dialogue Contextuel



SMART AUTOMATION  
TECHNOLOGIES

Realisé par :  
Taibi Mohamed Yassin  
10/09/2025

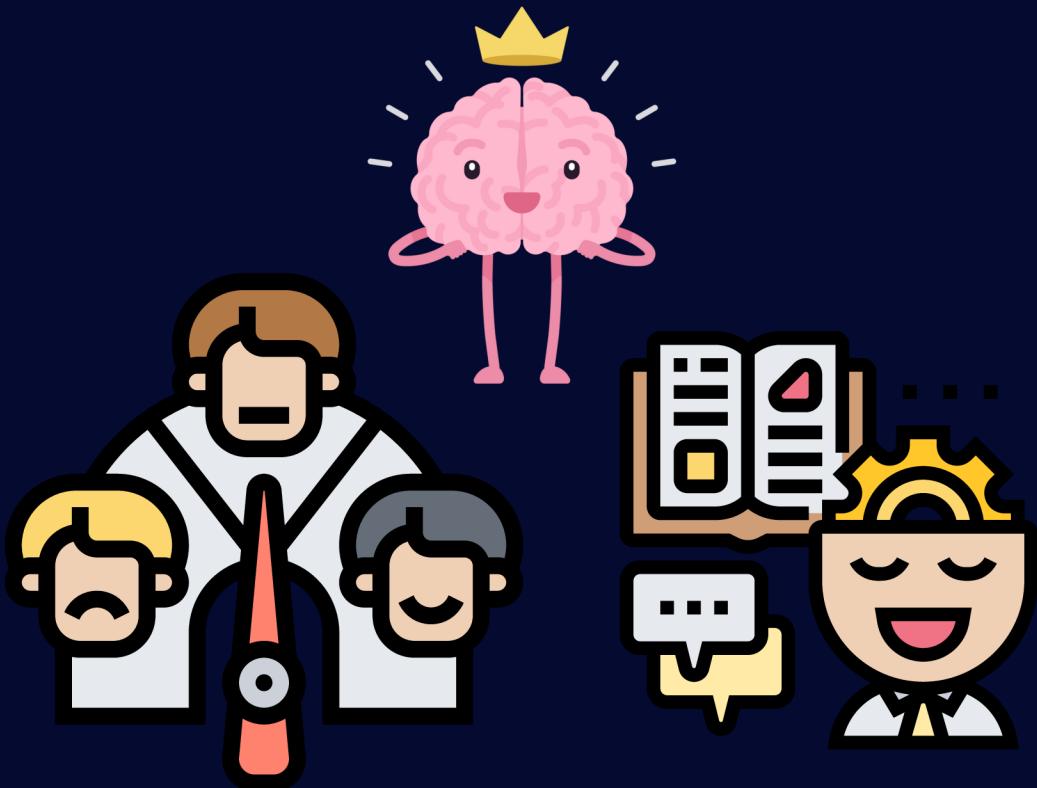
Encadrée par :  
Madame Salma LIICHI

## Objectif du projet

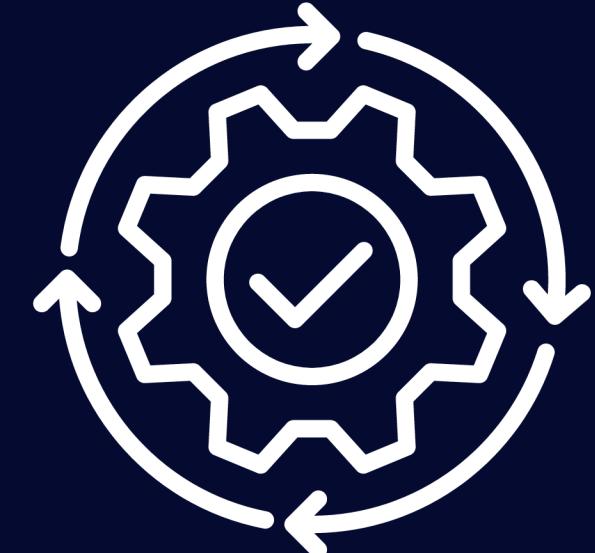


**Créer un assistant client intelligent capable de :**

- Comprendre le langage naturel (intentions + émotions)
- Répondre de manière personnalisée
- Maintenir une session de dialogue cohérente
- S'adapter au ton émotionnel du client



# Architecture globale du système



Détection  
d'intention  
(BERT)

Détection  
émotionnelle

Extraction  
d'entités

Génération de  
réponse (Bitext +  
Mistral-7B)

Interface  
utilisateur (Web /  
Colab)

# Détection d'intention (BERT)

- Modèle BERT pour classifier les intentions (commande, retour, question...)
- Seuil de confiance pour filtrer les doutes
- Fallback vers recherche sémantique si besoin



```
preds = np.argmax(predictions.predictions, axis=-1)
labels = predictions.label_ids

print(f"\n Accuracy: {accuracy_score(labels, preds):.2%}")
print("\n Rapport complet:\n", classification_report(labels, preds, target_names=)

[25] import os
os.environ["WANDB_DISABLED"] = "true"
os.environ["WANDB_MODE"] = "disabled" # Double protection
train_results = trainer.train() # Démarre l'entraînement

Epoch Training Loss Validation Loss
1 0.014000 0.016995
2 0.008200 0.016640
3 0.000300 0.010743
```

```
[25] exemples = [
    "can i buy ?", # → Devrait retourner "facturation"
    "check my order", # → Devrait retourner "probleme_technique"
    "cancel mine ?" # → Devrait retourner "annulation"
]

for texte in exemples:
    result = classifier(texte)
    print(f"Texte: {texte}")
    print(f"Intention prédictive: {result[0]['label']} (Confiance: {result[0]['score']:.2%})")
    print("---")

    Texte: can i buy ?
    Intention prédictive: LABEL_19 (Confiance: 99.98%)
    ---
    Texte: check my order
    Intention prédictive: LABEL_25 (Confiance: 99.22%)
    ---
    Texte: cancel mine ?
    Intention prédictive: LABEL_0 (Confiance: 99.93%)
    ---

for texte in exemples:
    result = classifier(texte)
    label_id = int(result[0]['label'].split("_")[1]) # Extrait le nombre de "LABEL_21"
    intent_name = le.inverse_transform([label_id])[0] # Convertit en texte
    print(f"Texte: {texte}")
    print(f"Intention prédictive: {intent_name} (Confiance: {result[0]['score']:.2%})")
    print("")

    Texte: can i buy ?
    Intention prédictive: place_order (Confiance: 99.98%)
    --
    Texte: check my order
    Intention prédictive: track_order (Confiance: 99.99%)
```

# Détection émotionnelle

- 4 types : Stressé, Pressé, Curieux, Neutre
- Analyse via embeddings et phrases-clés
- Adaptation du ton et des emojis en conséquence



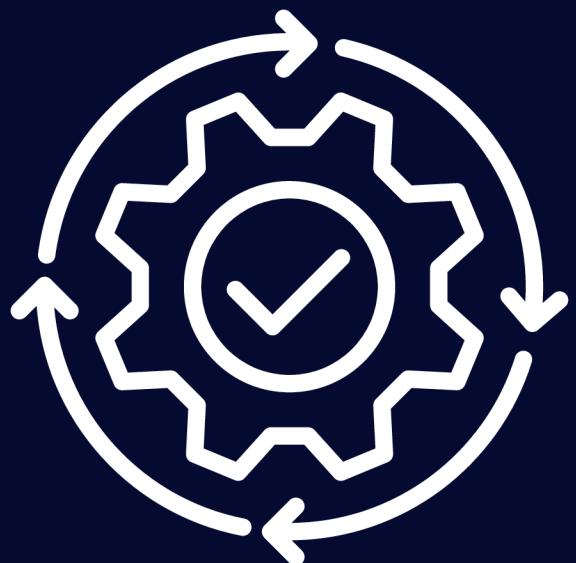
```
# Mapping des émotions vers types de clients
emotion_mapping = {

    # Stressé
    "nervousness": "stressé",
    "fear": "stressé",
    "anxiety": "stressé",
    "embarrassment": "stressé",
    "disappointment": "stressé",
    "confusion": "stressé",
    "grief": "stressé",

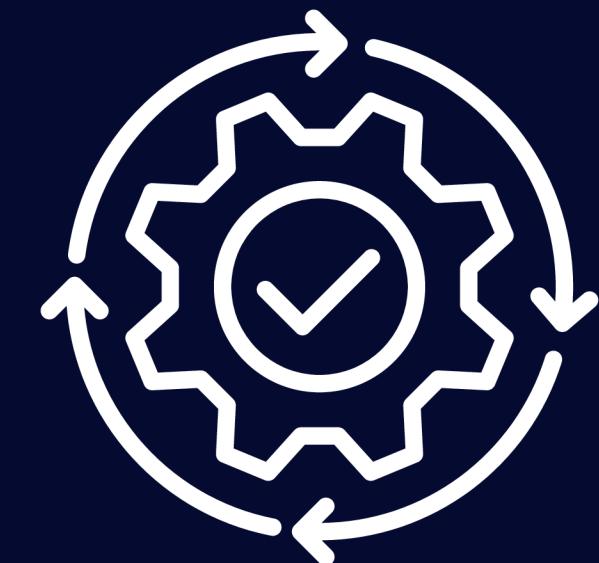
    # Curieux
    "curiosity": "curieux",
    "surprise": "curieux",
    "excitement": "curieux",
    "desire": "neutre",

    # Pressé
    "impatience": "pressé",
    "urgency": "pressé",

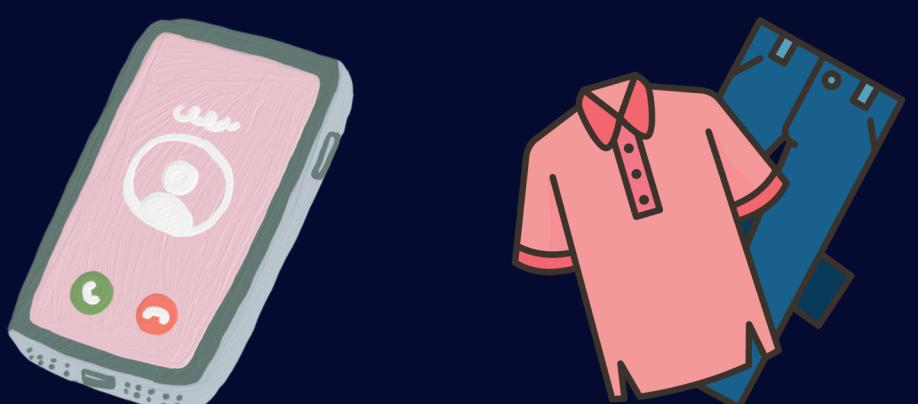
    # Neutre / Positif
    "neutral": "neutre",
    "calmness": "neutre",
    "approval": "neutre",
    "joy": "neutre",
    "love": "neutre",
    "gratitude": "neutre",}
```



# Extraction d'entités



SpaCy et PhraseMatcher pour détecter automatiquement des entités produits (comme la marque ou la couleur) dans les messages utilisateurs, en s'appuyant sur des listes fermées extraites du catalogue (dataset des produits qu'on a )



```
# Exemple de texte
sample_text = "I ordered a red iPhone 14 from Apple in large size on September 29 urgent in Rabat."

# Exécuter la fonction
entities = extract_entities(sample_text)

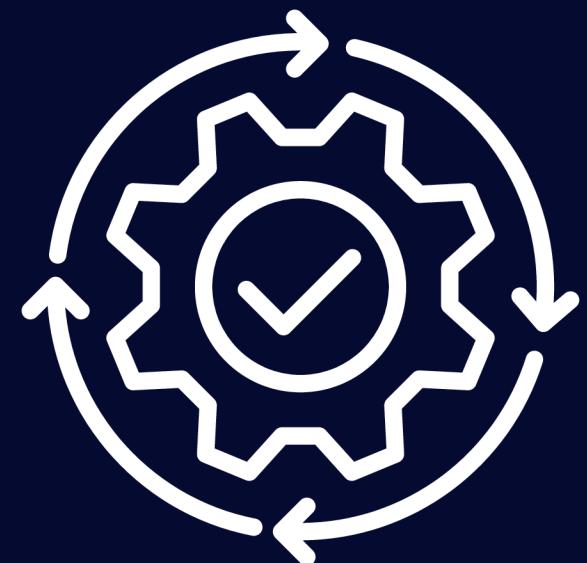
# Afficher les entités extraites
print("\nEntités extraites :")
for entity_type, values in sorted(entities.items()):
    print(f"{entity_type}: {', '.join(values)}")
```

I ordered a red COULEUR iPhone PRODUCT 14 from Apple ORG in large TAILLE size on September 29 DATE urgent CONTEXTE in Rabat GPE .

Entités extraites :

CONTEXTE: urgent  
COULEUR: red  
DATE: September 29  
GPE: Rabat  
ORG: Apple  
PRODUCT: iPhone  
TAILLE: large

# Génération de réponse (Bitext + Mistral-7B)



## 🧠 Compréhension et gestion du dialogue

- L'assistant comprend les messages types :
- "bonjour", "aide", "ce n'est pas ce que je voulais dire", "annule ça"
- Il gère :
  - Small Talk (salutations, aide)
  - Corrections et annulations naturelles
  - Suivi de session : mémorise les champs manquants et relance l'utilisateur

```
# %% [13] Small talk
# %% [13] Small talk - Révisé
small_talk_responses = {
    "hi": "Bonjour ! Comment puis-je vous aider aujourd'hui ?",
    "hello": "Bonjour ! En quoi puis-je vous assister ?",
    "how are you": "Je suis un programme mais merci de demander ! Comment puis-je vous aider ?",
    "what's up": "Tout va bien, prêt à vous aider !",
    "who are you": "Je suis un assistant client. Que puis-je faire pour vous ?",
    "what do you do": "Je peux vous aider avec les commandes, comptes et questions techniques.",
    "help": "Voici ce que je peux faire :",
    "assistance": "Je suis là pour vous aider. Que souhaitez-vous faire ?",
    "which services you have": "Voici nos services :",
    "what can you do": "Je peux vous aider avec :"
}
small_talk_examples = list(small_talk_responses.keys())
```

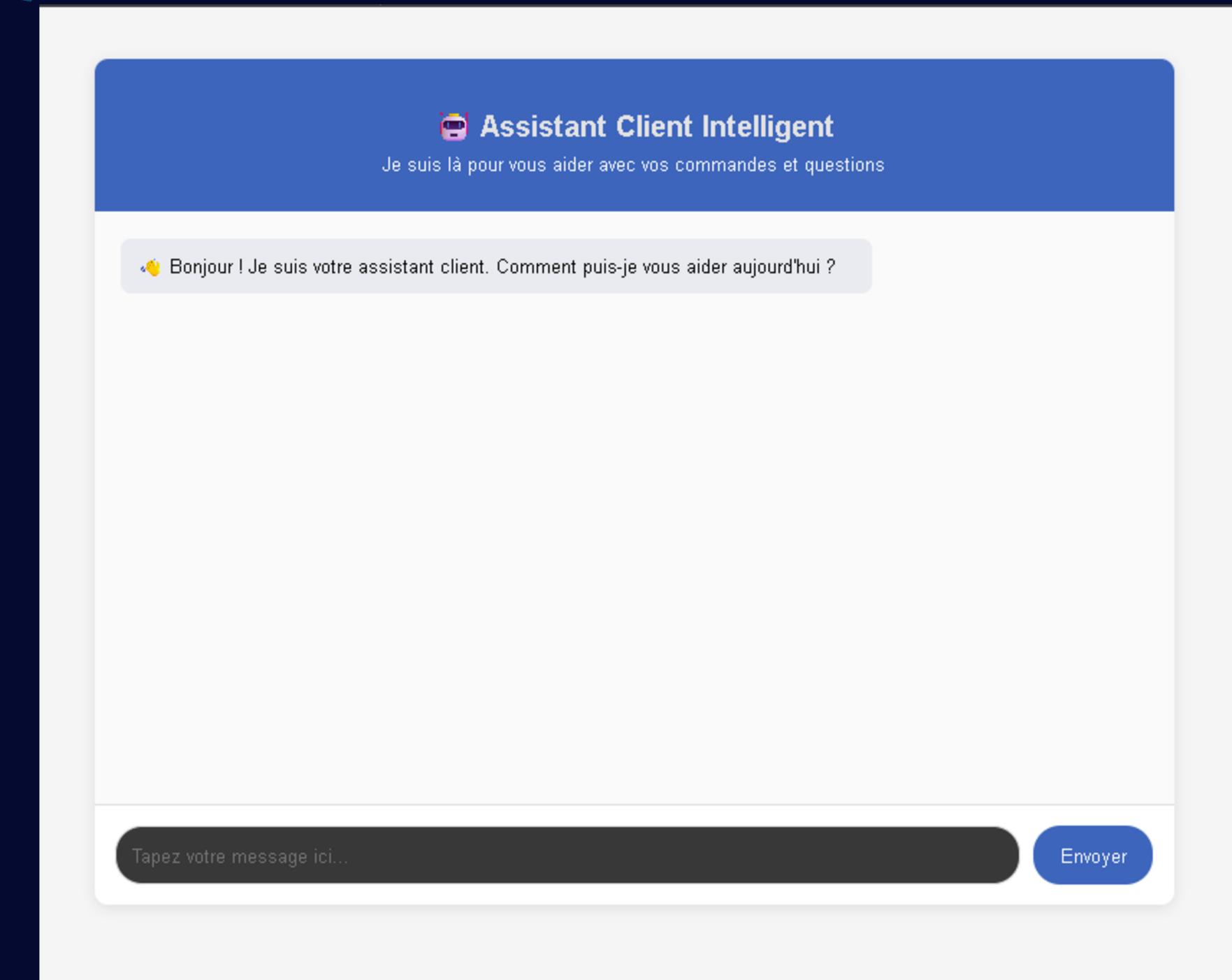
## 🤖 Génération de réponses intelligentes

- Deux méthodes combinées :
  - Base Bitext : réponses précises pré-apprises
  - LLM (Mistral-7B) : pour les cas ambigus ou inconnus
- Réponses :
  - Construites selon l'intention détectée
  - Adaptées au profil émotionnel (stressé, curieux, pressé)
  - Courtes, fiables, sans hallucination

```
# %% [16] Nouveau modèle de génération avec Mistral-7B
generation_model_name = "mistralai/Mistral-7B-Instruct-v0.2"
try:
    generation_tokenizer = AutoTokenizer.from_pretrained(generation_model_name)
    generation_model = AutoModelForCausalLM.from_pretrained(
        generation_model_name,
        device_map="auto",
        torch_dtype=torch.float16,
        load_in_4bit=True
    )
    generation_tokenizer.pad_token = generation_tokenizer.eos_token
except Exception as e:
    generation_model_name = "HuggingFaceH4/zephyr-7b-beta"
    generation_tokenizer = AutoTokenizer.from_pretrained(generation_model_name)
    generation_model = AutoModelForCausalLM.from_pretrained(
        generation_model_name,
        device_map="auto",
        torch_dtype=torch.float16
    )
    generation_tokenizer.pad_token = generation_tokenizer.eos_token
```

# Interface utilisateur (Web / Colab)

- Interface Streamlit : test local
- Interface Colab Web : HTML + JS pour chatbot embarqué
- Console Colab : test rapide et debug



## Améliorations futures

- Enrichir le dataset Bitext
- Ajouter la voix (Web Speech API ?)
- Connexion à une vraie base de données
- Interface mobile



## Conclusion

- Projet complet combinant NLP, émotion, LLM et UX
- Assistant IA fonctionnel, interactif et adaptable
- Prêt à être intégré dans un environnement client réel

