🤖 Chatbot pour l’ISET – Explication Technique

Ce chatbot est une application web destinée à répondre aux questions des étudiants de l’ISET à l’aide du **traitement automatique du langage naturel (NLP)** et de **l’apprentissage automatique**.

---

1. 🔧 Configuration initiale (`config.py`)

- Définit des **raccourcis** (`/contact`, `/horaires`, etc.) associés à des **réponses prédéfinies** et des **URLs**.
- Exemple :
  - `/contact` → "Voici les infos de contact", URL : `https://isetsf.rnu.tn/contacts/administration`
- Permet des réponses immédiates sans traitement NLP.

---

2. 🧹 Prétraitement des données (`data_processing.py`)

2.1 Chargement des données

- Les questions et réponses sont chargées depuis un fichier JSON.
- Indexées avec **Whoosh** pour des recherches rapides.
- Sorties principales : `questions`, `answers`, `categories`.

2.2 Prétraitement du texte

- Détection de la langue avec `langdetect`.
- Nettoyage, tokenisation, suppression des stopwords, **stemming** (SnowballStemmer).
- Exemple : *"Quelles sont les horaires ?"* → *"quel horair"*
- Résultat : `processed_questions`.

2.3 Vectorisation TF-IDF

- Utilisation de `TfidfVectorizer` :
  - Paramètres : `ngram_range=(1,2)`, `min_df=2`, `max_df=0.9`
- Résultat : `tfidf_matrix`.

2.4 Embeddings

- **Word2Vec** :
  - Moyenne des vecteurs de mots (`get_document_vector_w2v`)
- **FastText** :
  - Utilise des sous-mots → mieux pour les mots rares.

Les deux modèles sont sauvegardés pour réutilisation sans réentraînement.

---

3. 🧠 Entraînement des modèles de classification (`models.py`)

3.1 Données

- `X_train`, `X_test`, `y_train`, `y_test` obtenus par `train_test_split`.

3.2 Modèle Naive Bayes (MultinomialNB)

- Entrée : `X_train_tfidf`.
- Paramètre : `alpha=0.1`
- Mesures : `accuracy`, `F1-score`
- Sauvegarde : `models/nb_classifier.pkl`

3.3 Modèle KNN

- Testé avec `n_neighbors` de 3 à 7.
- Distance : **cosine**
- Le meilleur modèle est sauvegardé : `models/knn_classifier.pkl`

3.4 Sauvegarde du vectorizer

- Le `TfidfVectorizer` est sauvegardé dans `models/vectorizer.pkl`.

---

4. Logique du chatbot (`chatbot_logic.py`)

4.1 Vérification des raccourcis

- Ex : `/contact` → réponse directe via dictionnaire `shortcuts`.

4.2 Prétraitement utilisateur

- Appelle `preprocess_text`.

4.3 Recherche de réponse

Méthodes testées successivement jusqu'à une réponse satisfaisante :

Méthode	Seuil de similarité
TF-IDF	> 0.65
Word2Vec	> 0.8
FastText	> 0.8
Ensemble	> 0.7
KNN	distance < 0.7
Whoosh	fallback (0.5)

4.4 Prédiction de catégorie

- Via Naive Bayes ou KNN

4.5 Suggestions proactives

- Mots-clés → suggestions (ex : "horaire" → " Horaires")

4.6 Sauvegarde des nouvelles questions

- Si similarité < 0.8 : ajout dans `data/new_questions.json`

4.7 Structure de la réponse

```
{
  "answer": "Texte",
  "url": "Lien",
  "similarity": 0.82,
  "category": "Horaires",
  "is_shortcut": false,
  "method": "tfidf",
  "suggestions": [...]
}
```

---

5. Gestion des embeddings (`embeddings_utils.py`)

- Calcule la similarité entre la question utilisateur et les questions existantes :
  - Word2Vec → `get_best_match_with_word2vec`
  - FastText → `get_best_match_with_fasttext`
  - Ensemble → combine les deux méthodes

---

6. Interface utilisateur (`base.html` , `chat.html` , `about.html`)

`base.html`

- Barre latérale, thème clair/sombre, partage de sessions.

`chat.html`

- Messages utilisateur/bot, feedback , raccourcis dynamiques.

`about.html`

- Présentation des fonctionnalités du chatbot.

---

7. Backend Flask (`app.py`)

- Route `/` :
  - GET : charge session
  - POST : traite un message
- Routes additionnelles :
  - `/metrics` , `/report` , `/embeddings-metrics` , etc.
- Les sessions sont enregistrées dans `data/chat_sessions.json` .

---

8. Résumé du fonctionnement

1. **Lancement** :
   - Chargement des données et modèles.
2. **Interaction utilisateur** :
   - Via l'interface web.
3. **Traitement** :
   - Raccourci ou NLP → réponse générée.

4. **Affichage** :
    - Réponse + URL + score de similarité + catégorie.
5. **Sessions** :
    - Historique sauvegardé.
6. **Feedback** :
    - Évaluations enregistrées.
7. **Rapports** :
    - Comparaison des performances via `/report`.

Exemple d’interaction

Utilisateur : *"Quand est-ce que la bibliothèque ouvre ?"*

Processus :

1. Prétraitement → `"quand bibliotèqu ouvr"`
2. TF-IDF trouve une question similaire (>0.65)
3. Réponse : "La bibliothèque est ouverte du lundi au vendredi de 8h à 18h."
4. URL associée : `https://isetsf.rnu.tn/services/bibliotheque`
5. Suggestions : `" Bibliothèque"`
6. Catégorie prédite : `"Bibliothèque"`

Souhaite-tu aussi une version `.md` téléchargeable ?