Rapport technique

Chatbot de support client pour un site e-commerce de vêtements

Auteurs : Oumekelthoum Mohamed & Ahmed Oumarou

Document technique décrivant l'architecture, le pipeline NLP, la base de connaissances et le déploiement d'un chatbot dédié au e-commerce de mode.

Table des matières

1 Introduction **2**

2 Objectifs du projet **2**

3 État de l'art (travaux connexes) **2**

4 Méthodologie générale **2**

5 Architecture technique **3**
5.1 Composants 3
5.2 Flux de données 3

6 Base de connaissances **3**

7 Pipeline NLP **3**
7.1 Prétraitement 3
7.2 Vectorisation 4
7.3 Classification d'intentions 4

8 Interface utilisateur **4**

9 Entraînement et évaluation **4**
9.1 Entraînement 4
9.2 Évaluation 5

10 Déploiement **5**
10.1 Azure Container Apps (réalisé) 5
10.2 Configuration MongoDB Atlas (réalisé) 5
10.3 Initialisation des données (réalisé) 5

11 Limites du projet **5**

12 Perspectives **6**

13 Conclusion **6**

1. Introduction

Le support client est un enjeu central dans le e-commerce de mode. Les clients demandent souvent des informations similaires (tailles, livraison, retours, promotions, disponibilité). Un chatbot conversationnel permet de réduire la charge du service client tout en offrant des réponses rapides et cohérentes.

Objectif général du projet : concevoir un chatbot intelligent capable de comprendre les intentions des utilisateurs et de proposer des réponses utiles, tout en facilitant la recherche de produits.

2. Objectifs du projet

— Concevoir un chatbot adapté au domaine de la mode en ligne.
— Mettre en place un pipeline NLP complet (nettoyage, tokenisation, vectorisation).
— Entraîner un modèle de classification d'intentions.
— Construire une base de connaissances (FAQ + produits).
— Développer une interface web conversationnelle.
— Évaluer la performance du chatbot sur des scénarios réalistes.

3. État de l'art (travaux connexes)

Les chatbots e-commerce modernes combinent :

— Une base de connaissances (FAQ, catalogue produit).
— Des modèles de classification d'intentions.
— Une interface conversationnelle.

Exemples d'approches :

— **Chatbots à règles + FAQ :** réponses fixes, faible coût.
— **Chatbots ML avec classification d'intentions :** meilleure robustesse.
— **Chatbots avec modèles génératifs :** réponses plus riches, mais coût et contrôle plus difficiles.

4. Méthodologie générale

Le projet est organisé en quatre couches principales :

— **NLP Layer :** prétraitement du texte et extraction d'intentions.
— **Base de connaissances :** produits et FAQ stockés en JSON ou MongoDB.
— **Backend :** logique de conversation et recherche de produits.
— **Frontend :** interface Streamlit pour dialoguer et visualiser les produits.

5. Architecture technique

5.1. Composants

— **Interface :** Streamlit (`app.py`)

— **Moteur de chatbot :** `chatbot/chatbot_engine.py`

— **Classifieur d'intentions :** `nlp/intent_classifier.py`

— **Prétraitement :** `nlp/preprocessing.py`

— **Données :** `data/faq.json`, `data/products.json`, `data/training_data.json`

— **Base de données :** MongoDB (local ou distant)

5.2. Flux de données

1. L'utilisateur envoie un message.
2. Prétraitement NLP (nettoyage, tokenisation, suppression des stopwords, stemming).
3. Vectorisation TF-IDF.
4. Classification d'intention par *Logistic Regression*.
5. Sélection de la réponse (FAQ, actions, recherche de produits).
6. Affichage dans l'interface.

6. Base de connaissances

Deux sources principales :

— **FAQ :** questions fréquentes et réponses standardisées (retours, paiements, livraison).

— **Produits :** catalogue e-commerce avec nom, catégorie, genre, prix, description.

Les fichiers JSON sont utilisés pour initialiser la base MongoDB via :

`scripts/init_database.py`

7. Pipeline NLP

7.1. Prétraitement

Le texte est nettoyé avec :

— normalisation en minuscules,

— suppression des caractères spéciaux,

— tokenisation,

— stopwords français,

— stemming (*SnowballStemmer*).

7.2. Vectorisation

Le modèle utilise **TF-IDF** pour transformer le texte en vecteurs numériques.

7.3. Classification d'intentions

Modèle : Logistic Regression
Sortie : intention + probabilité de confiance.

Intentions supportées :

- `recherche_produit`
- `livraison`
- `paiement`
- `retour`
- `promotion`
- `contact`
- `salutation`
- `au_revoir`

8. Interface utilisateur

L'application Streamlit offre :

- une interface chat,
- une page produits avec recherche,
- un tableau de bord (dashboard) de statistiques,
- une page configuration.

Cette interface permet une démonstration claire du fonctionnement du chatbot.

9. Entraînement et évaluation

9.1. Entraînement

Le modèle est entraîné via :

```
python scripts/train_model.py
```

Ce script charge les données d'intentions, entraîne le classifieur, et sauvegarde le modèle et le vectoriseur.

9.2. Évaluation

L'évaluation se base sur :

— la précision de classification des intentions,

— des scénarios de test manuels (questions clients typiques),

— la satisfaction utilisateur (qualité des réponses).

10. Déploiement

10.1. Azure Container Apps (réalisé)

Le déploiement Azure a été effectué avec :

— Azure Container Registry,

— Container Apps Environment,

— Container App.

10.2. Configuration MongoDB Atlas (réalisé)

Pour rendre la base accessible en cloud :

— Création d'un utilisateur MongoDB Atlas avec mot de passe.

— Autorisation réseau `0.0.0.0/0` pour accepter les connexions externes.

— Récupération de l'URI `mongodb+srv://...` et ajout dans Azure comme variable `MONGODB_URI`.

— Ajout des variables `DATABASE_NAME=chatbot_commerce` et `COLLECTION_NAME=chatbot_commerce`

— Redémarrage de la Container App après mise à jour des variables.

10.3. Initialisation des données (réalisé)

Exécution du script :

```
python scripts/init_database.py
```

Résultats :

— 1000 produits insérés.

— 5 entrées FAQ insérées.

11. Limites du projet

— Les réponses sont basées sur intentions + FAQ, pas de génération libre.

— La qualité dépend des données d'entraînement.

— La base locale ne convient pas pour le déploiement cloud.

12. Perspectives

Améliorations possibles :

— Passage à un modèle BERT pour une meilleure compréhension.

— Intégration d'un système de recommandation.

— Ajout d'une interface vocale.

— Apprentissage continu à partir des conversations.

13. Conclusion

Ce projet propose un chatbot fonctionnel et modulable pour le e-commerce de vêtements. L'architecture sépare clairement le NLP, la base de connaissances et l'interface, ce qui facilite l'évolution du système. Le déploiement sur Azure montre la faisabilité d'une mise en production.