

PROJET LLM

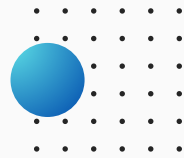
Noah De Gunst - Victor Chassé - Axel Dagnaud



2025



INTRODUCTION



Le Projet

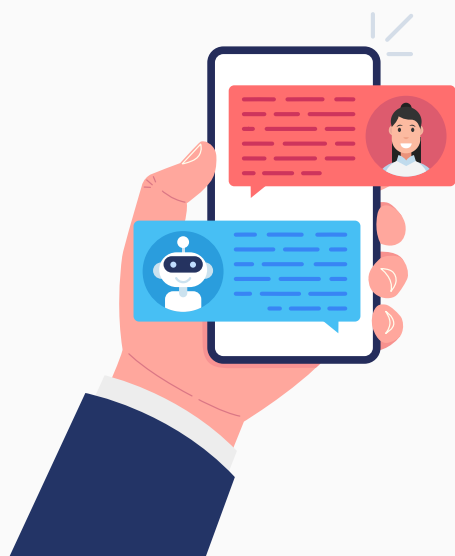


Dans le cadre de la troisième année du BUT Science des Données, ce projet a pour objectif de concevoir un chatbot intelligent capable de rechercher efficacement des informations spécifiques dans des documents contractuels liés à des sinistres. Ce projet a été réalisé pour le Groupe OPTISECURE, spécialisé dans l'assistance et la gestion des sinistres, où la recherche d'informations précises dans des contrats est un processus souvent long et fastidieux.



Contexte du projet

Le Groupe OPTISECURE gère de nombreux contrats relatifs aux sinistres, générant ainsi d'importants volumes de documents. Actuellement, la recherche d'informations spécifiques dans ces documents est une tâche chronophage et complexe pour les employés. L'objectif de ce projet est donc de développer un chatbot basé sur l'approche RAG (Retrieval-Augmented Generation), capable d'automatiser la recherche d'informations et d'améliorer la productivité.



Objectif

- Indexer les contrats en les découpant en segments pertinents et en créant des représentations vectorielles.
- Récupérer des informations grâce à une recherche sémantique.
- Générer des réponses précises et contextualisées en fonction des questions posées par les utilisateurs.

FONCTIONNEMENT D'UN MODÈLE RAG

RETRIEVAL-AUGMENTED GENERATION

Le modèle RAG combine les techniques de recherche d'information et de génération de texte pour fournir des réponses précises et pertinentes. Il fonctionne en deux étapes principales :

Indexation et Récupération :

- Collecte des données et découpage en segments pertinents.

```
def lire_contrats_avec_metadata(repertoire: str):  
    """  
    Lis les fichiers HTML dans un répertoire, extrait uniquement la première liste (<ul> ou <ol>) de chaque section.  
  
    Paramètres:  
    | repertoire (str): Chemin vers le répertoire contenant les fichiers HTML.  
  
    Sortie:  
    | list: Documents extraits (contenus de la première <ul>/<ol> de chaque section).  
    | list: Métadonnées associées à chaque liste.  
    | list: Identifiants uniques pour chaque liste.  
    """
```

- Vectorisation des segments et stockage dans une base de données optimisée.

```
def generer_embeddings_avec_metadata(documents, metadata, ids, modele: str = 'all-MiniLM-L6-v2') -> dict:  
    """  
    Génère des embeddings pour les documents en utilisant Sentence Transformers.  
  
    Paramètres:  
    | documents (list): Liste des textes (sections et sous-sections).  
    | metadata (list): Liste des métadonnées associées à chaque document.  
    | ids (list): Liste des identifiants uniques des documents.  
    | modele (str): Modèle Sentence Transformers à utiliser pour les embeddings.  
  
    Sortie:  
    | dict: Dictionnaire où les clés sont les IDs des documents, et les valeurs sont des embeddings et métadonnées.  
    """
```

- Recherche sémantique pour identifier les segments les plus pertinents par rapport à la question posée.

```
def requete_chromadb(question, collection_name="contrats_collection", modele="all-MiniLM-L6-v2"):  
    """  
    Effectue une requête sur la base de données et compare la similarité entre la question et les documents  
  
    Paramètres:  
    | question (str): question posé par l'utilisateur  
    | collection_name (str): par défaut le nom de la collection  
    | modele (): model d'embedding  
  
    Sortie:  
    | results (dict): dictionnaire des id (documents) et des taux de similarités avec la question |  
    """
```

FONCTIONNEMENT D'UN MODÈLE RAG

RETRIEVAL-AUGMENTED GENERATION

Génération de Réponse

- Les segments récupérés servent de contexte.

```
79 print(requete_chromadb("Je me suis fait volé ma voiture je fais quoi ?"))
```

PROBLEMS OUTPUT DEBUG CONSOLE **TERMINAL** PORTS

Documents et embeddings insérés dans ChromaDB avec succès!

```
{'ids': [['OptiSecure Assurances - Vol.html_sec_1_liste', 'OptiSecure Assurances - Vol.html_sec_2_liste', 'OptiSecure Assurances - Vol.html_sec_0_liste', 'OptiSecure Assurances - Incendie domestique.html_sec_1_liste', 'OptiSecure Assurances - Dégât des eaux.html_sec_0_liste']], 'embeddings': None, 'documents': [['Vol de véhicule :Prise en charge uniquement si le véhicule était assuré tous risques. Vol d'objets à l'extérieur :Couverture pour contrats "OptiConfort" et "OptiPremium".', 'Demander un constat de police ou une attestation officielle. Vérifier les preuves d'achat des biens déclarés volés. Transmettre les informations à l'expert si nécessaire.', 'Couverture :Vol avec effraction uniquement (preuve obligatoire).Biens mobiliers couverts jusqu'à 2 500 €. Conditions :Constat de police ou gendarmerie obligatoire.Franchise de 500 € applicable.', 'Couverture :Dommages immobiliers jusqu'à 10 000 € avec une franchise de 500 €.Frais de déblaiement limités à 1 000 €. Conditions spécifiques :Rapport des pompiers obligatoire.Justificatif d'entretien des installations électriques requis.', 'Couverture :Fuites provenant des installations intérieures.Dommages au logement jusqu'à 5 000 € (franchise de 250 €).Non-couverture : infiltrations par toiture ou façades non entretenues. Conditions de prise en charge :Fournir un constat amiable signé par les deux parties.Preuve d'entretien annuel des installations requise.']], 'uris': None, 'data': None, 'metadatas': [{'header': 'OptiSecure Assurances - Vol', 'section': '2. Cas particuliers', 'source': 'OptiSecure Assurances - Vol.html', 'sous_section': 'aucune', 'type_liste': 'ul'}, {'header': 'OptiSecure Assurances - Vol', 'section': '3. Procédures', 'source': 'OptiSecure Assurances - Vol.html', 'sous_section': 'aucune', 'type_liste': 'ol'}, {'header': 'OptiSecure Assurances - Vol', 'section': '1. Prise en charge selon les contrats', 'source': 'OptiSecure Assurances - Vol.html', 'sous_section': '1.1 Contrat "Optiéco"', 'type_liste': 'ul'}, {'header': 'OptiSecure Assurances - Incendie domestique', 'section': '2. Prise en charge selon les contrats', 'source': 'OptiSecure Assurances - Incendie domestique.html', 'sous_section': '2.1 Contrat "Optiéco" (Entrée de gamme)', 'type_liste': 'ul'}, {'header': 'OptiSecure Assurances - Dégât des eaux', 'section': '1. Prise en charge selon les contrats', 'source': 'OptiSecure Assurances - Dégât des eaux.html', 'sous_section': '1.1 Contrat "Optiéco" (Entrée de gamme)', 'type_liste': 'ul'}], 'distances': [[1.0757604837417603, 1.1113569736480713, 1.1215587854385376, 1.2127324342727661, 1.2463651895523071]], 'included': ['<IncludeEnum.distances: 'distances'>', '<IncludeEnum.documents: 'documents'>', '<IncludeEnum.metadatas: 'metadatas'>']}
```

- Le modèle LLM génère une réponse contextualisée à partir des informations pertinentes.

Deploy

Chatbot OptiSecure Assurances



Bonjour que faire si ma voiture a été volée ?

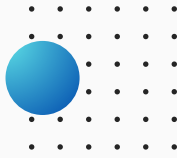


Bonjour ! Je suis agent chez Optisecure. Je suis désolé d'apprendre que votre voiture a été volée. Pour vous aider au mieux, j'ai besoin de quelques informations supplémentaires. Pouvez-vous me fournir :

- Le numéro de votre police d'assurance Optisecure ?
- La date et l'heure du vol ?
- Le lieu exact où la voiture a été volée ?
- Une description détaillée de votre véhicule (marque, modèle, année, couleur, numéro d'immatriculation) ?
- Avez-vous signalé le vol aux autorités (police) ? Si oui, pouvez-vous me fournir le numéro de référence du rapport de police ?

SCHÉMA D'ARCHITECTURE /

OUTILS



BeautifulSoup

Bibliothèque Python utilisée pour extraire et manipuler des données à partir de fichiers HTML. Elle a permis de parser et d'extraire efficacement le contenu des documents HTML pour les convertir en données exploitables.



Hugging Face

Plateforme spécialisée dans les modèles de langage et les outils d'IA. Hugging Face a été utilisé pour accéder et exploiter des modèles de traitement du langage naturel (NLP), notamment les modèles d'embeddings et de génération de texte.



Chroma

Base de données vectorielle utilisée pour stocker les représentations vectorielles des segments de texte. ChromaDB permet une recherche sémantique rapide et efficace grâce à son optimisation pour les requêtes sur des données vectorielles.



Streamlit

Framework Python permettant de créer des interfaces utilisateur interactives. Il a été utilisé pour concevoir l'interface du chatbot, facilitant l'interaction entre l'utilisateur et le modèle. L'interface inclut des champs de saisie pour les questions et l'affichage des réponses générées.



Google AI Studio

Environnement de développement permettant d'expérimenter et de tester des modèles d'IA. Google AI Studio a été utilisé pour entraîner et affiner les modèles de langage utilisés dans le projet, facilitant l'expérimentation et le déploiement.

SCHÉMA D'ARCHITECTURE / OUTILS

