Projet MongoDB

Objectifs du module

- Analyser et extraire des informations pertinentes à partir d'une base de connaissances textuelle.

- Utiliser MongoDB pour stocker, rechercher et interroger ces données, en exploitant les fonctionnalités NoSQL avancées (texte intégral, recherche, graphes).

- Développer une API RESTful en Python (Flask ou FastAPI) pour exposer la base de connaissance à une application (ex. : chatbot).

MongoDB

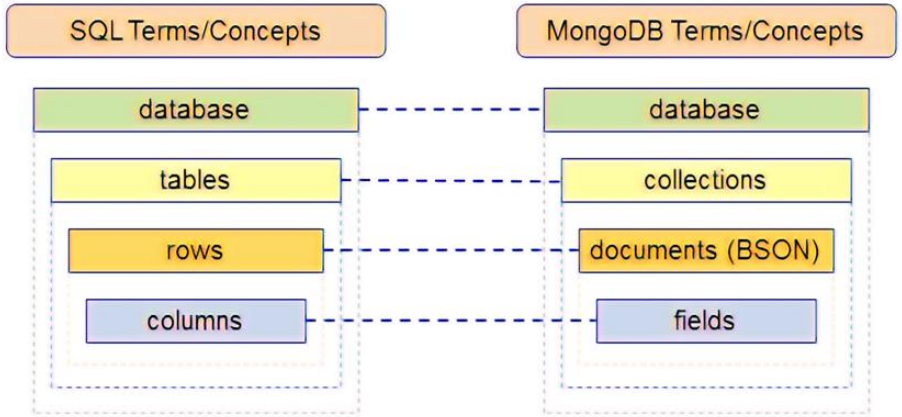
- Base de données NoSQL orientée documents (JSON-like)

- Stocke les données sous forme de collections et de documents

- Idéal pour les données semi-structurées, les projets agiles, les applications temps-réel

Structure d'une base MongoDB : Collections > Documents

```
{
  "_id": "123",
  "question": "Comment contacter le support ?",
  "reponse": "Par email à support@entreprise.com",
  "tags": ["contact", "support"]
}
```



Opérations CRUD

C : Create: insertOne(), insertMany()

Méthode	Description
db.collection.insertOne()	Il est utilisé pour insérer un seul document dans la collection.
db.collection.insertMany()	Il est utilisé pour insérer plusieurs documents dans la collection.
db.createCollection()	Il est utilisé pour créer une collection vide.

R : Read : find(), filtres simples ou complexes

Méthode	Description
db.collection.find()	Il est utilisé pour récupérer des documents de la collection.
db.collection.findOne()	Récupère un seul document qui correspond aux critères de requête.

U : Update: updateOne(), updateMany(), opérateurs $set, $push, $addToSet

Méthode	Description
db.collection.updateOne()	Il est utilisé pour mettre à jour un seul document de la collection qui répond aux critères donnés.
db.collection.updateMany()	Il est utilisé pour mettre à jour plusieurs documents de la collection qui répondent aux critères donnés.
db.collection.replaceOne()	Il est utilisé pour remplacer un seul document de la collection qui satisfait aux critères donnés.

D : Delete: deleteOne(), deleteMany()

Méthode	Description
db.collection.deleteOne()	Il est utilisé pour supprimer de la collection un seul document qui répond aux critères donnés.
db.collection.deleteMany()	Il est utilisé pour supprimer de la collection plusieurs documents qui répondent aux critères donnés.

Recherche texte dans MongoDB

Indexation texte

•Création : db.collection.createIndex({ champ: "text" })

=> une seule indexation texte est autorisée par collection !

Mais cette indexation peut concerner plusieurs champs.

```
db.faq.createIndex({ question: "text", reponse: "text" })
```

Requêtes texte

•find({ $text: { $search: "support email" } })

=> MongoDB va rechercher tous les documents contenant le mot “support” ou “email”, dans les champs indexés.

!! on peut utiliser les guillemets pour chercher une expression exacte : $text: { $search: "\"support technique\"" }

Quand on stocke des documents textuels (FAQ, des descriptions produits ou de la documentation)

=> on veut retrouver les documents pertinents à partir de mots-clés ou d’une requête utilisateur.

Exemples :

•Trouver tous les articles qui parlent de « support client »

•Rechercher dans une base de mails les messages contenant le mot « urgent »

Trier par pertinence (score)

MongoDB attribue un score de pertinence à chaque document trouvé. Pour le voir et trier par score

=> Cela permet d’obtenir les résultats les plus pertinents en premier.

```
db.faq.find(
  { $text: { $search: "support" } },
  { score: { $meta: "textScore" } }
).sort({ score: { $meta: "textScore" } })
```

Recherche texte dans MongoDB

Limitations et bonnes pratiques

•Une seule indexation texte par collection

•Poids accordé aux champs ("score")

•L'index texte ignore les mots très communs ("le", "la", "et"...) → stop words

•Par défaut, la recherche est non accentuée et non sensible à la casse

•Si vous voulez plus de flexibilité (filtres, pondérations, synonymes…), MongoDB propose aussi Atlas Search, plus avancé.

Comment effectuer une recherche textuelle dans MongoDB : échantillonnage et filtrage | Filtrage MongoDB | Échantillonnage MongoDB

Indexation avancée dans MongoDB

Types d'index

1.Index simple : db.collection.createIndex({ champ: 1 })

2.Index composé : db.collection.createIndex({ champ1: 1, champ2: -1 })

3.Index texte : db.collection.createIndex({ champ: "text" })

Objectifs des index

1.Accélérer les recherches

2.Permettre des tris efficaces

3.Supporter les requêtes textuelles ou relationnelles

Analyse avec explain(): `db.faq.find({ question: /support/ }).explain("executionStats")`

Recherche texte contextuelle

Recherche textuelle avec score

```
db.faq.find(
  { $text: { $search: "compte client" } },
  { score: { $meta: "textScore" } }
).sort({ score: { $meta: "textScore" } })
```

Pondération des champs

```
db.collection.createIndex(
  { titre: "text", contenu: "text" },
  { weights: { titre: 10, contenu: 5 } })
```

Recherches combinées

•Filtres + recherche texte :

```
db.docs.find({
  $text: { $search: "chatbot" },
  type: "faq" })
```

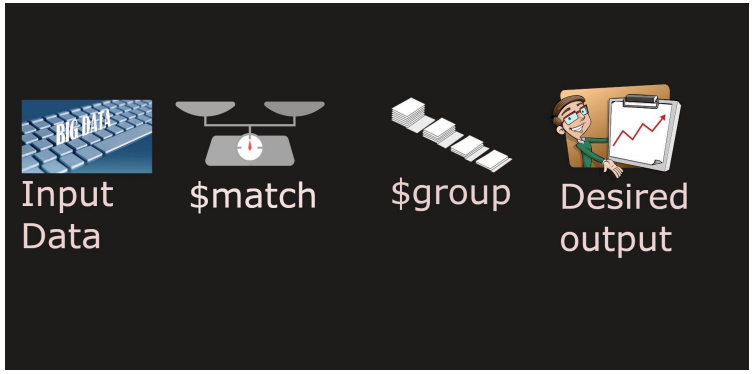
Pipelines d'agrégation

Qu'est-ce que l'agrégation ?

Un ensemble d'étapes permettant de transformer ou regrouper les données

Opérateurs utiles

$match, $group, $project, $sort, $limit, $unwind



Exemple

```
db.faq.aggregate([
  { $match: { tags: "support" } },
  { $group: { _id: "$theme", total: { $sum: 1 } } },
  { $sort: { total: -1 } }])
```

Cas d'usage :

•Compter les questions par thème

•Créer une liste de suggestions par mot-clé

Comment utiliser le pipeline d'agrégation dans MongoDB ?

Les relations dans MongoDB

MongoDB est une base NoSQL dite "documentaire" — elle est conçue pour stocker des documents JSON imbriqués.
Mais dans certains cas, on a besoin de lier des données entre collections, un peu comme on ferait avec des jointures en SQL (ex. : relier une question à son thème, un article à son auteur, etc.).

MongoDB propose des solutions pour ces cas :

•**DBRef** (peu utilisé, plus historique): DBRef est une manière standardisée de faire une référence d'un document vers un autre

•$**lookup** (pour faire des jointures locales): C'est la méthode native et efficace pour relier des documents entre collections dans un pipeline d'agrégation.

$**graphLookup** (pour explorer des structures hiérarchiques, comme des arbres ou des graphes) : permet de parcourir des structures récursives : par exemple un arbre d'organisation, des catégories imbriquées, ou une chaîne de dépendances.

Données relationnelles et graphes

Modélisation avec DBRef

Stocker une référence à un autre document :

=> Mais MongoDB ne résout pas ces liens automatiquement → c'est à l'application de faire le find correspondant.

Pas recommandé sauf cas très spécifiques.

```
{
  question: "...",
  theme: { "$ref": "themes", "$id": ObjectId("...") }
}
```

Jointure avec $lookup

Joindre deux collections localement

=>Utile pour enrichir les résultats avec des infos liées (auteurs,

```
db.questions.aggregate([
  {
    $lookup: {
      from: "themes",
      localField: "theme_id",
      foreignField: "_id",
      as: "theme_info"
    }
  }
])
```

```
{
  "_id": 1,
  "texte": "Comment se connecter ?",
  "theme_id": 101,
  "theme_info": [
    { "_id": 101, "nom": "Connexion" }
  ]
}
```

Exploration en profondeur avec $graphLookup

Idéal pour des hiérarchies ou des dépendances multiples

=> Utile si vous avez des catégories imbriquées, des FAQ liées entre elles, ou des dépendances.

```
db.employes.aggregate([
  {
    $graphLookup: {
      from: "employes",
      startWith: "$manager_id",
      connectFromField: "manager_id",
      connectToField: "_id",
      as: "hierarchie"
    }
  }
])
```

Application

https://learn.mongodb.com/learning-paths/introduction-to-mongodb:

1. Comprendre les concepts fondamentaux de MongoDB.
2. Créer un cluster MongoDB Atlas et s'y connecter
3. Réaliser des opérations CRUD simples
4. Réaliser des opérations d'agrégation, l'indexation, la modélisation des données
5. Créer des index tester des recherches

API et webservice

Qu’est-ce qu’une API ?

* Une API (Application Programming Interface) est un ensemble de règles permettant à deux applications distinctes de communiquer entre elles et d’échanger des données.
* Elle agit comme une interface entre des données existantes et un programme indépendant, évitant ainsi de redévelopper entièrement une application pour y intégrer de nouvelles informations.

* Les API sont essentielles pour permettre à différentes applications ou systèmes de partager des données ou des fonctionnalités, que ce soit au sein d’une entreprise ou avec des services externes.

* Elles facilitent l’intégration de services tiers, comme les passerelles de paiement ou les plateformes de cartographie, dans des applications existantes.

API : qu’est-ce-que c’est et à quoi ça sert ?

Qu’est-ce qu’une API ?

* Les API définissent des points de terminaison (endpoints) que les applications peuvent utiliser pour envoyer des requêtes et recevoir des réponses, souvent via des protocoles comme HTTP.

* Elles permettent une communication standardisée, indépendamment des langages de programmation utilisés par les différentes applications.

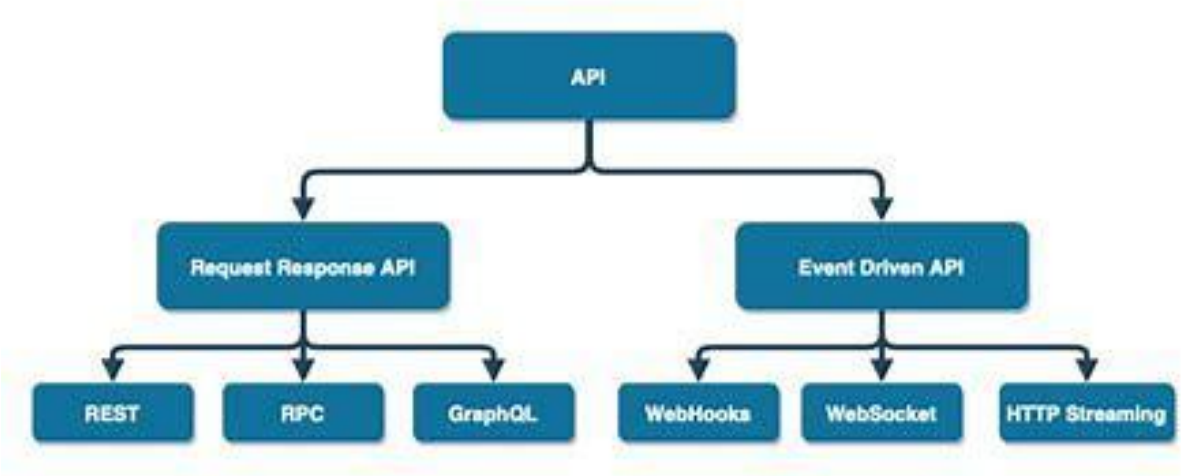
Avantages des API

- **Automatisation** : Permettent d’interagir avec des services tiers sans intervention manuelle.
- **Interopérabilité** : Facilitent la communication entre différentes applications ou systèmes.
- **Réutilisabilité** : Un même service peut être exploité par plusieurs clients ou applications.
- **Mise à jour centralisée** : Toute amélioration de l’API bénéficie immédiatement à toutes les applications qui l’utilisent.
- **Sécurité** : Accès aux données restreint selon des permissions définies.

https://www.redhat.com/fr/topics/api/what-is-a-rest-api

Types d'API

- **API REST** : Utilisent les méthodes HTTP standard (GET, POST, PUT, DELETE) pour interagir avec les données.
- **API SOAP** : Utilisent XML pour les échanges, souvent employées dans les services d'entreprise nécessitant un haut niveau de sécurité.
- **API GraphQL** : Permettent de récupérer uniquement les données nécessaires en une seule requête, optimisant ainsi la performance des échanges.
- **API WebSocket** : Permettent une communication bidirectionnelle en temps réel, très utile pour des applications comme les messageries instantanées.



Sécurité des API

L’utilisation d’API nécessite des mesures de sécurité appropriées pour protéger les données échangées, notamment :

Authentification : Vérification de l’identité des utilisateurs ou applications accédant à l’API.

Autorisation : Définition des permissions pour accéder à certaines ressources ou fonctionnalités.

Chiffrement : Protection des données échangées contre les interceptions.

Aperçu des API REST avec Python

- API : Interface permettant à des applications de communiquer entre elles
- REST : Style d'architecture utilisant les verbes HTTP (GET, POST, PUT, DELETE)

Exemple minimal (avec Flask ou FastAPI)

```python
from flask import Flask, jsonify
app = Flask(__name__)

@app.route("/faq", methods=["GET"])
def get_faq():
    return jsonify({"question": "...", "reponse": "..."})
```

Voir la suite dans le Notebook!

```python
from fastapi import FastAPI

app = FastAPI()

@app.get("/")
def read_root():
    return {"message": "Hello, World!"}
```

uvicorn main:app --reload
main : nom du fichier sans .py
app : nom de l'objet FastAPI
--reload : pour recharger automatiquement en développement

Pour tester l'API :
http://127.0.0.1:8000/
Documentation interactive Swagger : http://127.0.0.1:8000/docs
Redoc : http://127.0.0.1:8000/redoc

Documentation interactive automatique avec /docs

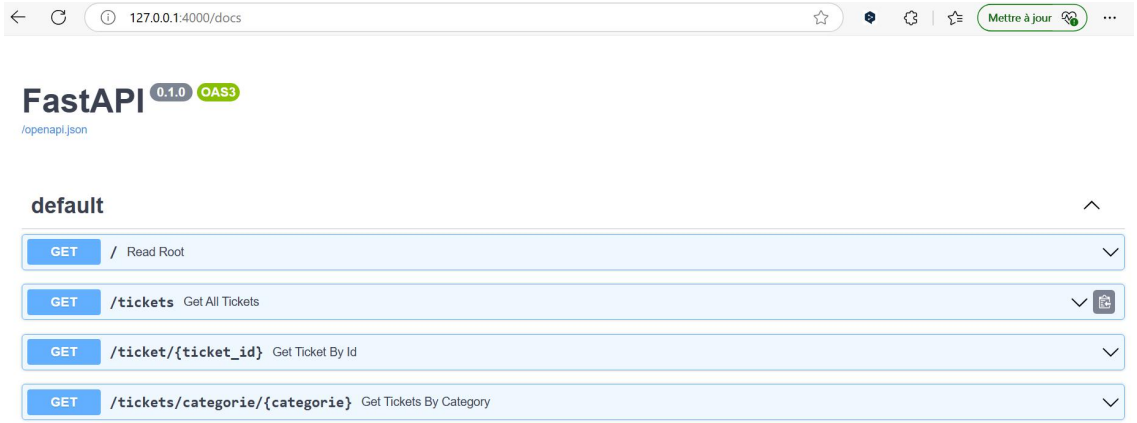
FastAPI génère automatiquement une documentation interactive de votre API grâce à Swagger UI, accessible par défaut à l'adresse /docs.Cette interface permet de :

•Visualiser tous les endpoints disponibles.

•Tester les requêtes directement depuis le navigateur.

•Voir les schémas de données et les réponses attendues.

=>Cette fonctionnalité repose sur la spécification OpenAPI, garantissant une documentation conforme aux standards.



GeeksforGeeks+4Python in Plain English+4DEV Community+4

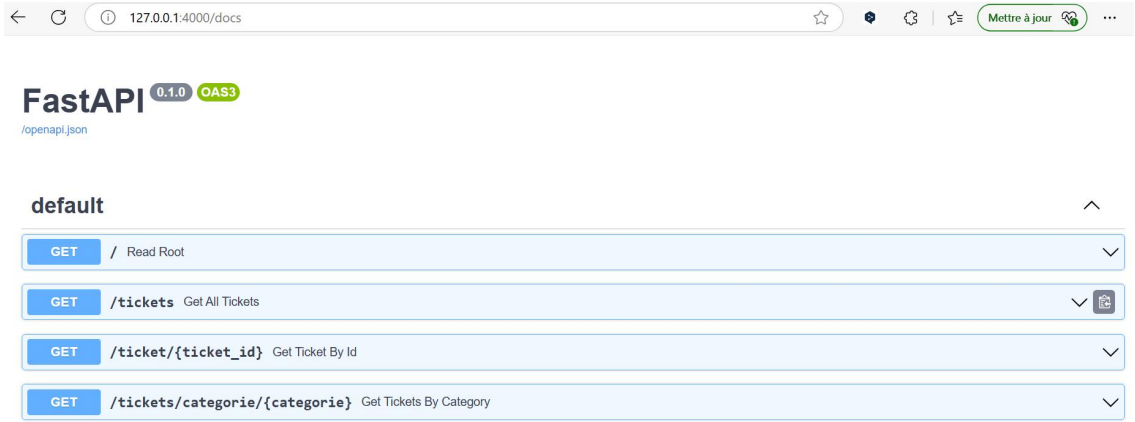
Documentation interactive automatique avec /docs

FastAPI offre la possibilité de personnaliser l'interface Swagger UI selon vos besoins :

- Modifier l'URL d'accès à la documentation avec le paramètre docs_url.

- Désactiver l'interface en définissant docs_url=None.

- Configurer des paramètres supplémentaires via swagger_ui_parameters, tels que le thème ou la mise en page.



TP2

Créer une API RESTful permettant de gérer des données en utilisant FastAPI pour le backend et MongoDB comme base de données.

Getting Started With MongoDB and FastAPI | MongoDB

L’objectif est de construire une application qui permet de :

•Créer de nouveaux documents dans la base de données

•Lire des documents existants

•Mettre à jour des documents

•Supprimer des documents

=> Ces opérations sont exposées via des endpoints RESTful, facilitant ainsi l'interaction avec la base de données MongoDB.

Best Practices for Using Flask and MongoDB | MongoDB

NLP & Machine Learning

Base de connaissance en entreprise

•Réservoir d'informations structurées et pertinentes permettant de répondre à des questions

Exemples de sources

•FAQ internes, manuels techniques, emails, documentation produits

Cas d'usage

•Support client automatisé (chatbots)

•Assistance à la décision (agents intelligents)

•Onboarding / formation interne

Qu'est-ce que le NLP ?

Le NLP (Traitement du Langage Naturel) consiste à faire comprendre, générer ou analyser le langage humain par des machines.

Tâches classiques

•Classification de textes (ex: spam, thème)

•Résumé automatique

•Détection de mots-clés / entités nommées

•Analyse de sentiment

•Traduction automatique

Pourquoi c'est utile ici ?

•Mieux structurer une base de connaissance

•Générer des tags ou catégories automatiquement

•Identifier les entités pour des recherches plus fines

spaCy

spaCy est une bibliothèque Python open-source spécialisée dans le traitement du langage naturel (NLP). Elle est conçue pour être rapide, légère et facile à intégrer dans des applications.

Ce qu'elle peut faire :

Tokenisation : découper un texte en mots

Lemmatisation : identifier la racine des mots

Part-of-speech tagging : détecter les rôles grammaticaux

NER (Named Entity Recognition) : extraire des entités comme des noms, lieux, dates

Détection de dépendances syntaxiques

Extraction de mots-clés simples (noun chunks)

Avantages :

Très rapide

Facile à utiliser

Parfait pour des cas d'usage simples à modérés

Limites :

Moins performant sur du texte très complexe ou subtil

Ne gère pas nativement des modèles "Transformer"

spaCy

spaCy : *pour extraire des entités nommées*

•Rapide, simple à utiliser

•Parfait pour les tâches classiques : NER, POS, dépendances

•Modèles légers (fr, en, etc.)

```python
import spacy

nlp = spacy.load("en_core_web_md")  # Utilise le modèle anglais car l'annonce est en anglais
text = """
This spacious and bright apartment is located in the heart of Paris,
near the Eiffel Tower and the Champs-Élysées. Perfect for families or business travelers.
WiFi and breakfast included.
"""
doc = nlp(text)

for ent in doc.ents:
    print(ent.text, ent.label_)
```

Paris GPE
Eiffel Tower FAC
Champs-Élysées LOC

Types d'entités que spaCy peut reconnaître (fr_core_news_md) :

•PER : personne

•LOC : lieu géographique

•GPE : geo-political entity (pays, ville)

•FAC : facility (bâtiment, infrastructure)

•ORG : organisation

•MISC : autre (ex : marques)

•DATE, TIME, PERCENT, MONEY, QUANTITY, ORDINAL, CARDINAL

Utilisation typique en base MongoDB après enrichissement

```json
{
  "description": "This spacious and bright apartment is located in the heart of Paris...",
  "entities": [
    { "text": "Paris", "label": "GPE" },
    { "text": "Eiffel Tower", "label": "FAC" },
    { "text": "Champs-Élysées", "label": "LOC" }
  ]
}
```

Transformers : Des modèles profonds de NLP de dernière génération

Un Transformer est une architecture de réseau de neurones utilisée pour comprendre le langage humain. Elle a été introduite dans l'article "Attention is All You Need" (Vaswani et al., 2017).

Utilisé dans les modèles célèbres comme :

BERT, GPT ,RoBERTa, CamemBERT (version francophone) ,DistilBERT (version légère) ,T5, BART, mBART (résumé, traduction)

Fonctionnalités :

- Compréhension contextuelle des textes longs
- Résumé automatique
- Traduction
- Classification
- Extraction d’information (NER avancée)
- Génération de texte

Transformers

HuggingFace Transformers : *pour classifier ou résumer*

- Basé sur des modèles pré-entraînés de pointe (BERT, CamemBERT, etc.)
- Meilleure précision, mais plus lourds
- Idéal pour la classification ou le résumé

```python
from transformers import pipeline

# Crée un pipeline pour la reconnaissance d'entités nommées en français
ner = pipeline("ner", model="Jean-Baptiste/camembert-ner", aggregation_strategy="simple")

text = "Bienvenue chez Orange ! Pour contacter le service client, appelez le 3900 du lundi au samedi."
results = ner(text)

# Affiche les entités détectées
for r in results:
    print(f"{r['word']} ({r['entity_group']}) - score: {r['score']:.2f}")
```

Orange (ORG) - score: 0.99
3900 (MISC) - score: 0.95
lundi (DATE) - score: 0.85
samedi (DATE) - score: 0.84

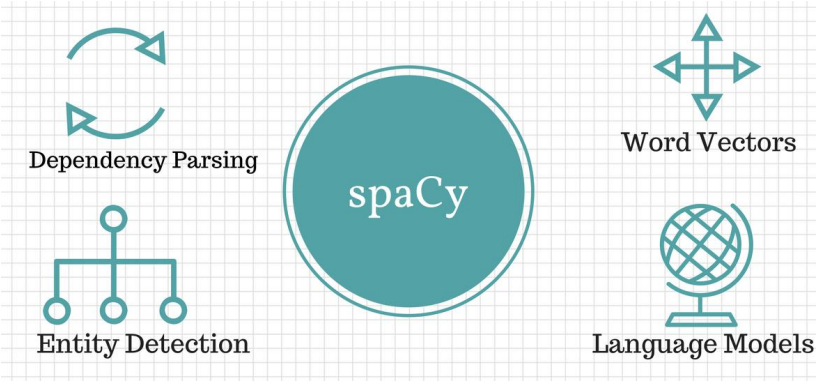
Utilisation en base MongoDB après enrichissement

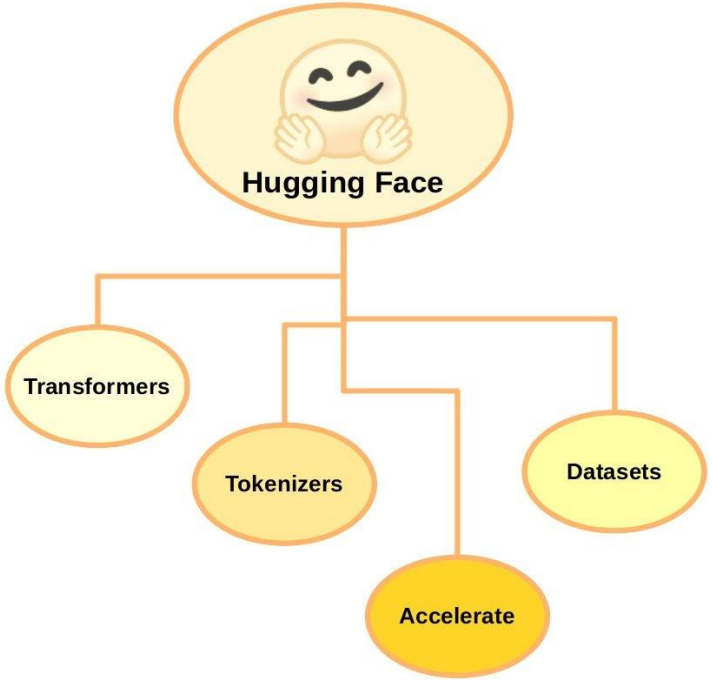
```json
{
  "text": "Bienvenue chez Orange ! Pour contacter le service client...",
  "entities": [
    { "text": "Orange", "label": "ORG" },
    { "text": "3900", "label": "MISC" },
    { "text": "lundi", "label": "DATE" },
    { "text": "samedi", "label": "DATE" }
  ]
}
```

Hugging Face Transformers : l'API au service du NLP

spaCy vs Transformers

Caractéristique	spaCy	Transformers (via Hugging Face)
Type	Bibliothèque NLP	Modèles préentraînés profonds (deep learning)
Vitesse	Très rapide	Plus lent (surtout sans GPU)
Langues	Anglais, français, etc.	Multilingue (selon le modèle)
Performance	Bonne pour des tâches simples	Excellente sur des tâches complexes
Installation	Léger	Plus lourd (nécessite PyTorch ou TensorFlow)
Cas d'usage	Extraction de base, tag, entités	Résumé, classification, NER avancée, QA





Cas d'usage en entreprise

A. Classification de documents

•Déterminer à quel thème appartient une question ou un article

•Peut être supervisé (avec un modèle fine-tuné) ou non supervisé (clustering)

B. Résumé automatique

•Résumer des longues fiches produit ou tutoriels pour l'affichage rapide

•Gain de temps pour les utilisateurs

C. Extraction d'entités / mots-clés

•Identifier des noms de produits, d’outils, des organisations, etc.

•Utilisable comme tags dans MongoDB pour la recherche

Application à MongoDB

Exemple d’enrichissement :

Avant :

```
{
  "question": "Comment contacter le support technique ?"
}
```

Après Analyse avec NLP:

```
{
  "question": "Comment contacter le support technique ?",
  "tags": ["support", "contact"],
  "entites": [
    { "texte": "support technique", "type": "ORG" }
  ]
}
```

TP3

Objectif: Appliquer un traitement NLP sur une base de connaissance.Enrichissement des documents MongoDB avec des tags automatiques ou entités nommées.

1. Choix de l'outil NLP

•spaCy pour extraire des entités nommées

•HuggingFace pour classifier ou résumer (optionnel)

2. Traitement des documents

•Charger les textes depuis MongoDB

•Appliquer un modèle NLP sur chaque document

•Extraire les entités ou tags clés

3. Mise à jour de la base

•Ajouter les résultats NLP comme nouveaux champs :{

"tags": [...], "entites": [...], "resume": "..."}

4. Test des nouvelles fonctionnalités de recherche

•Recherche par tags , Recherche par entité , Recherche pondérée combinant les deux

sample_mflix: Base de données de films

•Collections : movies, comments, theaters, users

•Données textuelles :

- comments: commentaires utilisateurs sur les films
- movies: synopsis, titres, genres

pour :

- Résumé de synopsis
- Extraction d'entités (films, acteurs)
- Classification des genres à partir de la description