Cours NLP and GenAI

2-6 Decembre 2024

georgialoukatou@gmail.com

Instructions pour le rendu

Pour le rendu, veuillez fournir les éléments suivants :

1. Repository GitHub :
    - Créez un repository GitHub à partir de votre compte personnel. Ajoutez vos scripts correspondant aux projets.
2. Report détaillé :
    - Accompagnez le repository d'un rapport au format PDF.
    - Le rapport doit inclure un resumé dédié pour chaque projet contenant :
        - Les choix effectués (modèles, techniques, outils, paramètres).
        - Les problématiques rencontrées (s'il y en a).
        - Une photo, image, plot, ou screenshot présentant un résultat représentatif pour chaque projet.
    - **À travailler en binôme (ou individuellement) et à rendre avant le lundi 09/12/2025.**

Cours NLP and GenAI

Projet 1 : Topic Modeling des Avis des Produits

Étape 1 : Prétraitement des Avis de Produits

Les données utilisées proviennent du **dataset Amazon Review Dataset 2023**, spécifiquement du subset **Cell_Phones_and_Accessories**. Les données à votre disposition comprennent uniquement les **1000 premiers avis clients** de ce subset. Le dataset complet est disponible sur le site officiel : https://amazon-reviews-2023.github.io/.

Dans ce travail pratique, vous allez installer et configurer un outil de traitement du langage naturel pour analyser des textes en anglais (par exemple, SpaCy, NLTK). Chargez un modèle de langue anglaise (par exemple, https://spacy.io/models/en) adapté pour des tâches comme la lemmatisation et la suppression des stop words.

1.Chargement des données

- Fichiers disponibles ici: https://drive.google.com/drive/folders/1cb9ynyj2dLVnA_zV1Hhbra-eJbq4If54?usp=sharing
  - meta.jsonl : Contient les métadonnées des avis.
  - reviews.jsonl : Contient les textes des avis clients.
- Sélection des champs :
  - Choisissez les champs pertinents des avis, comme title et text, pour créer une liste de documents qui servira de base au traitement.

2.Traitement linguistique

1. Tokenisation :
   - Découpez les textes en mots (tokens).
2. Lemmatisation :
   - Extrayez les lemmes (formes de base des mots) pour chaque texte.
3. Suppression des stop words :
   - Éliminez les stop words (mots fonctionnels sans pertinence contextuelle).
4. Exclusion des éléments non pertinents :
   - Ignorez les signes de ponctuation.
   - Identifiez et supprimez les termes qui n'ajoutent pas de valeur (e.g., *123, www*).
5. Inspection des données :
   - Examinez les données pour identifier d'autres éléments inutiles ou bruités qui pourraient être exclus.

3.Sauvegarde des données préparées

- Pour chaque avis, créez une liste de tokens filtrés (lemmes sans stop words ni bruit). Sauvegardez les listes résultantes dans un fichier JSON.

Étape 2 : Clustering non supervisé des documents pour identifier des topics et mots-clés

Chargez le fichier **JSON** contenant des tokens pré-traités. Les données doivent être une liste de documents, chaque document correspondant à un avis déjà nettoyé et prêt pour l'analyse.

1.Génération des embeddings

Utilisez un modèle pré-entraîné pour représenter les documents sous forme vectorielle, comme all-MiniLM-L6-v2 via SentenceTransformers Huggingface (autres: all-MiniLM-L12-v2, bert-base-uncased).

Alternative: Employez **TfidfVectorizer** pour obtenir une matrice sparse représentant les documents (sk-learn)

2.Clustering

Appliquez l'algorithme **KMeans** pour regrouper les documents en un nombre défini de clusters (num_clusters).

Alternative : Utilisez **DBSCAN** si les données sont inégalement réparties ou si le nombre de clusters n'est pas connu à l'avance.

3.Analyse des clusters

1. **Association des documents aux clusters** : Pour chaque cluster identifié, regroupez les documents correspondants.
2. **Fréquences des mots** :
    i. Calculez les fréquences des mots dans chaque cluster.
    ii. Identifiez les **10 mots les plus fréquents** pour chaque cluster.
3. **Vérification de pertinence** : Analysez si ces mots fréquents sont représentatifs du thème ou du sujet du cluster.
4. Extraction des mots-clès en utilisant une des methodes suivantes :NER (Named Entity Recognition), bigrams (ou n-grams), c-TFIDF, modèles prè-entrainés comme DistilBERT via HuggingFace.

Bonus

- **Évaluation des clusters** :

Calculez le **Silhouette Score** pour évaluer la qualité des regroupements.

Étape 3 : Analyse des sentiments des avis clients

1. Chargement et préparation des données

- Chargez le fichier **JSON** contenant les avis clients. Extrayez les **notes réelles** (ratings) pour une future évaluation des performances.

---

2. Chargement du modèle pré-entraîné

- Utilisez un modèle de sentiment pré-entraîné (par exemple, **nlptown/bert-base-multilingual-uncased-sentiment** sur Hugging Face) ainsi que son **tokenizer**.

---

3. Prétraitement des données

- Divisez les données en **lots** (batches) à l'aide d'un outil comme un DataLoader. Ajustez la **taille des lots** (batch size) en fonction des ressources disponibles.

---

4. Analyse des sentiments

- **Tokenisation** : Tokenisez les textes avec le tokenizer du modèle.
- **Passage dans le modèle** : Analysez les textes avec le modèle BERT pour obtenir les prédictions. Si necessaire, appliquez une fonction softmax aux **logits** pour obtenir les probabilités des classes.
- Associez chaque prédiction à un **label de sentiment** (1 étoile à 5 étoiles).

---

5. Évaluation des performances

- Associez les **sentiments prédits** (1 étoile à 5 étoiles) à des scores numériques (de 1 à 5).
- **Corrélation** : Calculez la corrélation de Pearson entre les **notes réelles** et les **prédictions**.

Bonus

- **Expérimentation** : Essayez d'autres modèles de sentiment disponibles sur Hugging Face.
- **Visualisation** : Créez des graphiques (par exemple avec Matplotlib) pour comparer les distributions des **notes réelles** et des **notes prédites**.

Projet 2 : Création d'une architecture Retrieval Augmented Generation sur des descriptions de produits Amazon

Pour ce projet, il sera nécessaire d'utiliser des **LLM** (Large Language Models). Il y a deux options :

A. **Créer un token OpenAI** : Cela coûte **5 euros**, qui correspond à la somme minimum pour activer un token OpenAI. Pendant le projet, nous ne dépenserons que des centimes, ce qui signifie que le solde restant pourra être utilisé pour d'autres projets à venir.

   UPDATE: Vous pouvez utiliser
`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`

   Il s'agit d'une clé fournie par l'école et destinée à être partagée entre tous les étudiants. Merci de ne pas l'utiliser pour d'autres projets, car nous n'aurons pas de seconde clé à disposition 🙂.

B. **Utiliser des bibliothèques gratuites** : Des bibliothèques comme **Ollama** permettent d'accéder à des modèles tels que **Llama 3**, qui sont suffisants pour répondre aux besoins de ce projet.

---

1. Chargement des données :
   - Téléchargez le fichier JSONL **meta.jsonl**, qui contient des descriptions de produits.

---

2. Prétraitement et segmentation des textes
   - Installer la librairie LangChain (alternative: LlamaIndex)
   - Divisez les descriptions longues en morceaux de taille appropriée (par exemple, avec **RecursiveCharacterTextSplitter**).
     i. Taille des morceaux (chunk_size) : par exemple, 512 caractères.
     ii. Chevauchement entre les morceaux (chunk_overlap) : 128 caractères.

---

3. Création d'un index vectoriel :
   - Générez des **embeddings** pour chaque morceau de texte à l'aide d'un modèle d'embedding (Sentence Embeddings de HuggingFace, par exemple all-MiniLM-L6-v2 ou OllamaEmbeddings).

---

4. Création d'une base de données vectorielle:
   - Stockez les embeddings générés dans une **base de données vectorielle** (par exemple, **Chromadb**).

---

5. Création d’un système de récupération (retrieval system) :
    - Configurez un **retriever** à partir de la base vectorielle pour rechercher les descriptions pertinentes correspondant aux requêtes des utilisateurs.

---

6. Réglage du LLM :
    - Structurer le prompt en utilisant des Techniques de **Prompt Engineering**. Indiquez au modèle d'utiliser uniquement les informations issues de la base de données. Ajoutez une instruction claire pour qu'il ne fasse pas appel à des connaissances internes. Précisez que le modèle doit indiquer qu'il ne sait pas lorsqu'il ne peut pas trouver une réponse appropriée. Expliquez dans le prompt que le modèle agit comme un assistant destiné à répondre aux questions des utilisateurs à partir des documents fournis. Demandez au modèle de fournir les passages exacts ou les documents spécifiques d'où il a extrait les informations. Précisez que le modèle ne doit pas générer d'informations sensibles, inappropriées ou potentiellement incorrectes.
    - Exemple d’un prompt trop simple qui doit être amelioré :
      "Vous êtes un assistant intelligent qui répond aux questions en utilisant uniquement les informations contenues dans les documents pertinents fournis. Voici les documents pertinents pour cette question :
      {{documents}}
      Basé uniquement sur ces documents, répondez précisément et de manière concise à la question suivante :
      {{question}}."

---

7. Rechercher des Documents Pertinents :

    - Une fois la base vectorielle créée, utilisez un retriever pour rechercher les documents pertinents en fonction de la requête utilisateur.
        - Transformez la requête utilisateur en embedding avec le même modèle utilisé pour les documents.
        - Utilisez la méthode de recherche de ChromaDB pour trouver les documents les plus proches dans l'espace vectoriel.

---

8. Chaîne de récupération :

    - Combinez les descriptions des produits pertinentes ({{documents}}) avec la question utilisateur, passez-les dans un **modèle LLM (par exemple, gpt-3.5-turbo pour OpenAI ou llama 3.1 pour Ollama)**, et générez une réponse structurée.

---

9. Exécution de requêtes utilisateur :

- Formulez des questions utilisateur, comme : *"Tell me about OnePlus 6T"*. Faites passer la requête dans la chaîne RAG et imprimez la réponse générée.

---

10. Test de différents paramètres pour le modèle LLM (*temperature, top-p*).

---

11. Intégrez le système RAG dans une interface utilisateur interactive en utilisant **Streamlit**.