Projet NLP

Consignes

- **Travail en binôme** : chaque groupe doit être composé de **2 étudiants**.
- Vous devez **choisir un projet** parmi la liste proposée ou en proposer un (à valider au préalable).
- Vous serez évalués sur deux éléments :
  1. **Le code source**, livré sur **GitHub** (qualité, organisation, documentation, clarté),
  2. **Une présentation orale**, comprenant :
     - Une **présentation du projet** (objectifs, choix techniques, architecture, etc.),
     - Une **démonstration de l'application**.

Projet 1

"RédacAO : Assistant intelligent de rédaction d'appels d'offres basé sur l'historique des AO passés"

Objectif principal :

Développer un assistant de rédaction automatique d'appels d'offres (AO) à l'aide d'un modèle de langage (LLM), en s'appuyant sur un corpus d'AO historiques similaires. L'outil doit être capable de :

- Analyser les anciens AO pour extraire les structures, formulations types, contraintes récurrentes.

- Générer une première ébauche d’un nouvel AO à partir de quelques entrées (titre, thématique, contraintes spécifiques).
- Suggérer des reformulations ou compléter automatiquement des sections manquantes.

Livrables attendus :

1. **Corpus d'appels d'offres annotés ou structurés**
   - Scraping ou utilisation d’un jeu de données existant.
   - Nettoyage et structuration des données (séparation des sections, extraction des métadonnées).
2. **Module d’analyse automatique des AO**
   - Extraction de patrons (templates) récurrents par type d’appel d’offre.
   - Clustering ou classification des AO pour suggérer des styles/structures selon le domaine.
3. **Assistant LLM (fine-tuning ou prompt-engineering)**
   - Génération de textes à partir de prompts structurés.
   - Complétion contextuelle de sections : "objectif", "prestations attendues", "critères de sélection", etc.
   - Réécriture automatique selon un ton ou registre choisi (formel, accessible, technique…).
4. **Interface utilisateur (optionnelle mais fortement recommandée)**
   - Application Streamlit ou Gradio pour tester l'assistant.
   - Entrée des paramètres de l’AO et génération dynamique de documents.

Tech stack suggéré :

- Langage : Python
- LLM : GPT-4, Mistral, LLaMA 3, fine-tuned ou via API
- Librairies NLP : spaCy, HuggingFace Transformers, LangChain
- Front-end : Streamlit
- Stockage : JSON / base SQLite pour le corpus ou autres

Axes d'approfondissement possibles :

- Évaluation automatique de la qualité du texte généré vs historique.
- Détection de plagiat ou de redondance avec des AO passés.
- Suggestion de formulations juridiques ou normatives correctes.
- Adaptation au contexte pays/langue (ex. AO en français, exigences spécifiques…).

Projet 2

"AlerteAO+ : Système de veille intelligente des appels d’offres basé sur LLMs avec alertes personnalisées par email"

Objectifs enrichis avec LLMs :

1. **Scraper automatiquement** des AO depuis des plateformes ciblées.
2. **Classer sémantiquement** les AO par thématique via un LLM.
3. **Résumer intelligemment** le contenu avec un LLM (résumé extractif + interprétatif).
4. **Évaluer la pertinence** de chaque AO pour l'utilisateur (scoring personnalisé via un prompt).

5. **Formater dynamiquement** les emails avec des résumés clairs, le lien, et un message d’intro généré.
6. **Option avancée : personnalisation du ton** du message (formel, synthétique, etc.)

Tâches clés avec les LLMs :

Étape	LLM utilisé	Fonction
1. Classification	OpenAI GPT-4 / Mistral / LLaMA + prompt	"Dans quelle thématique cet appel d'offre s'inscrit-il ?"
2. Résumé	`text-davinci-003`, GPT-4, ou T5/Bart	"Résume cet appel d’offre en 4 lignes en conservant les points essentiels."
3. Scoring	LLM ou modèle fin-tuné	"Ce texte est-il pertinent pour le domaine *énergie* ? Score sur 10."
4. Génération de mail	GPT-4 / Claude / LLaMA	Génère le contenu d’un email personnalisé (avec intro + résumés + call-to-action)

Livrables attendus :

- Script de scraping + stockage
- Module de traitement par LLM (via OpenAI API, Mistral, ou modèle local)
- Générateur de mails HTML personnalisés
- Rapport ou logs des AO traités

- Documentation et exemple de configuration (préférences thématiques, fréquence...)

Technos recommandées :

- **Scraping** : `BeautifulSoup`, `playwright`, `requests-html`
- **LLMs** :
  - API : OpenAI (`gpt-4`, `gpt-3.5`), Anthropic, Mistral via HuggingFace
  - Locaux : `llama.cpp`, `transformers`, `LangChain` pour l'orchestration
- **Stockage** : SQLite / MongoDB

Bonus avancés :

- **Personnalisation du résumé** par utilisateur (ex : "met en avant les critères de sélection")
- **Détection automatique de doublons** ou de rééditions d’AO
- **Application web de visualisation** (facultative)