Ministère de l'Enseignement Supérieur et de la Recherche Scientifique

*****

Université de Sousse

*****

Institut Supérieur d'Informatique et des Techniques de Communication

Projet de LA NUIT DE L'INFO 2023

Thème : le changement climatique

Défi de la nuit 2023

Réalisé par : Nexus nights

Propose par :

Année universitaire 2022-2023

Problématique :

La complexité du changement climatique rend la diffusion d'informations précises et véridiques d'une importance vitale.

Cependant, la propagation de fausses informations sur ce sujet peut semer la confusion, entraver les efforts de sensibilisation et compromettre la prise de décision éclairée. La problématique centrale de notre défi est donc de développer une solution capable de trier efficacement les informations en provenance de diverses sources en ligne, afin de fournir une base solide pour des discussions et des actions informées sur le changement climatique

Etude Technique :

Dans cette étude technique, nous examinons la mise en œuvre de notre solution de détection de fausses informations sur le changement climatique. Du scraping des données à l'entraînement du modèle, nous détaillons les choix techniques, les adaptations spécifiques au contexte climatique, et les mesures d'évaluation.

Environnement matériel

Caractéristiques	Machine personnelle
System d'exploitation	Windows 10
Processeur	Intel Core i3 8Gen
RAM	8 GO
Disque dur	256 GO HDD
Carte graphique	MX

Cas d'utilisation :

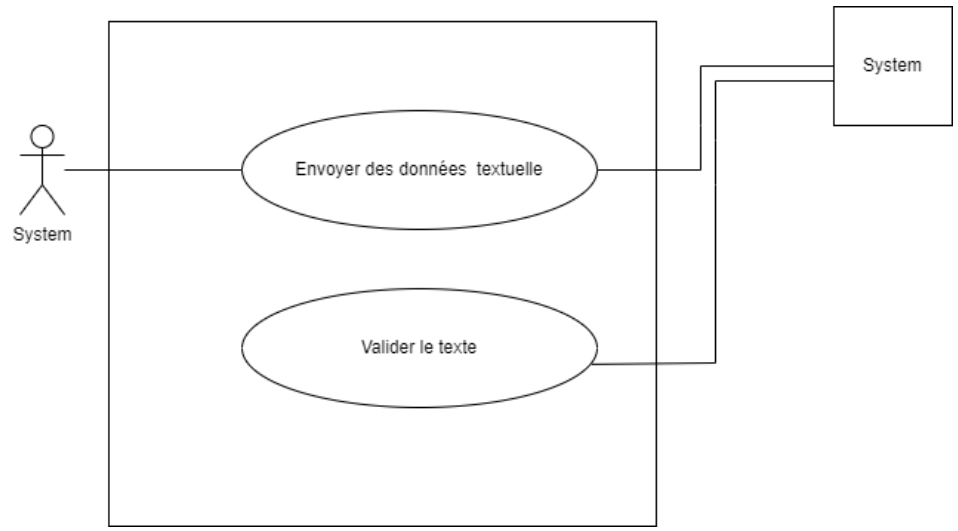

Figure 1 diagramme de cas d'utilisation

Ce bout de code illustre l'aspiration de données à partie des sites web de nuit de l’info

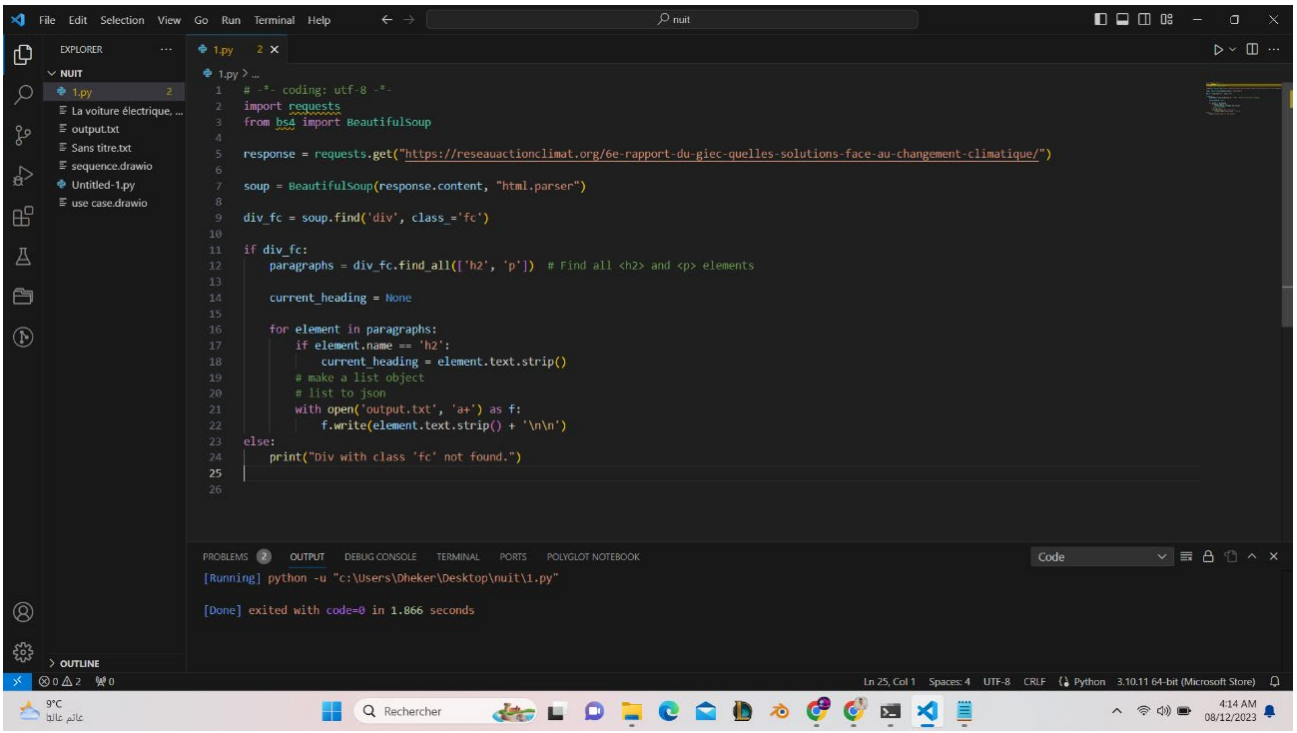

Figure 2 aspiration de données web

Ce fragment on va extraire de chaque paragraphe des tokens.

```
def extract_nouns(tokens):
    # Join tokens into a space-separated string
    text = ' '.join(tokens)

    # Load the French language model
    nlp = spacy.load("fr_core_news_sm")

    # Process the text with spaCy
    doc = nlp(text)

    # Get nouns and filter out stop words
    nouns = [token.text.lower() for token in doc if token.pos_ == 'NOUN' and token.text.lower() not in STOP_WORDS]

    return nouns

def tokenize_text(text):
    # Use word_tokenize for tokenization
    tokens = word_tokenize(text)
    return extract_nouns(tokens)
```

Figure 3 génération de Token

Ici on va comparer la similarité de mots entrées comme input par rapport aux tokens

```
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.metrics.pairwise import cosine_similarity

def calculate_cosine_similarity(input_tokens, token_list):
    # Convert token lists to strings for vectorization
    input_text = ' '.join(input_tokens)
    token_list_text = [' '.join(tokens) for tokens in token_list]

    # Combine input and list for vectorization
    all_text = [input_text] + token_list_text

    # Check if there's more than one document for vectorization
    if len(all_text) > 1:
        # Vectorize using TF-IDF
        vectorizer = TfidfVectorizer()
        vectors = vectorizer.fit_transform(all_text)

        # Calculate cosine similarity
        similarity_scores = cosine_similarity(vectors[0], vectors[1:])[0]

        return similarity_scores
    else:
        # Return an empty array if there's not enough data for comparison
        return []
```

Figure 4 comparaison des mots aux tokens

Finalement cette figure nous montre la valeur de similarité d'un exemple :

```
Average similarity is 0.6941091013860373, which is less than 0.8
```

Voici un tableau descriptif qui montre les avantages et les inconvénients de notre model

Avantage	Inconvénient
Elimination de maximum des fausses nouvelles.	Processus lent.