HATANBAATAR VAN Erkhembileg & MICHEL Victor

In [ ]:
```python
import pandas as pd
import numpy as np
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.model_selection import train_test_split, cross_val_score
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import classification_report
import re
from gensim.models import KeyedVectors
import nltk
nltk.download('stopwords')
from nltk.corpus import stopwords
stop_words = stopwords.words('english')
```

```
[nltk_data] Downloading package stopwords to
[nltk_data]     C:\Users\erkhe\AppData\Roaming\nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
```

In [ ]:
```python
def parse_data(file_path):
    with open(file_path, 'r', encoding='utf-8') as file:
        lines = file.readlines()

    data = []
    current_id = ''
    for line in lines:
        if line.startswith('###'):
            current_id = line.strip().lstrip('###')
        elif line.strip():
            parts = line.strip().split('\t')
            if len(parts) == 2:
                category, text = parts
                data.append({'id': current_id, 'category': category, 'text': text})

    return pd.DataFrame(data)

def clean_text(text):
    text = re.sub(r'[^a-zA-Z\s]', '', text, re.I|re.A)
    text = text.lower()
    text = text.strip()
    return text
```

In [ ]:
```python
df_train = parse_data('train.txt')
df_test = parse_data('test.txt')

df_train['text_cleaned'] = df_train['text'].apply(clean_text)
df_test['text_cleaned'] = df_test['text'].apply(clean_text)
```

In [ ]:
```python
vectorizer = TfidfVectorizer(stop_words=stop_words, max_features=3000, preprocessor=clean_text)
X_train = vectorizer.fit_transform(df_train['text_cleaned'])
y_train = df_train['category']
X_test = vectorizer.transform(df_test['text_cleaned'])
y_test = df_test['category']

lr_model = LogisticRegression(max_iter=1000, solver='sag', tol=0.1)
lr_model.fit(X_train, y_train)
lr_predictions = lr_model.predict(X_test)

print("Classification modèle bag of words + régression logistique :\n")
print(classification_report(y_test, lr_predictions))

cv_scores = cross_val_score(lr_model, X_train, y_train, cv=5, scoring='accuracy')
print("Validation croisée modèle bag of words, accuracy :", cv_scores.mean())
```

```
C:\Users\erkhe\AppData\Local\Packages\PythonSoftwareFoundation.Python.3.11_qbz5n2kfra8p0\LocalCache\local-packages\Python311\site-packages\sklearn\feature_extraction\text.py:409: UserWarning: Your stop_words may be inconsistent with your preprocessing. Tokenizing the stop words generated tokens ['are
nt', 'couldnt', 'didnt', 'doesnt', 'dont', 'hadnt', 'hasnt', 'havent', 'isnt', 'mightnt', 'mustnt', 'neednt', 'shant', 'shes', 'shouldnt', 'shouldve', 'thatll', 'wasnt', 'werent', 'wont', 'wouldnt', 'youd', 'youll', 'youre', 'youve'] not in stop_words.
  warnings.warn(
Classification modèle bag of words + régression logistique :

              precision    recall  f1-score   support

  BACKGROUND       0.58      0.49      0.53      2663
 CONCLUSIONS       0.69      0.66      0.67      4426
     METHODS       0.81      0.89      0.85      9751
   OBJECTIVE       0.70      0.58      0.63      2377
     RESULTS       0.82      0.83      0.83     10276

    accuracy                           0.77     29493
   macro avg       0.72      0.69      0.70     29493
weighted avg       0.77      0.77      0.77     29493

Validation croisée modèle bag of words, accuracy : 0.7700890791529936
```

In [ ]:
```python
model_path = 'BioWordVec_PubMed_MIMICIII_d200.vec.bin'
embedding_model = KeyedVectors.load_word2vec_format(model_path, binary=True)

def sentence_vector(sentence, model):
    words = sentence.split()
    word_vectors = [model[word] for word in words if word in model]
    if len(word_vectors) == 0:
        return np.zeros(model.vector_size)
    else:
        return np.mean(word_vectors, axis=0)

X_train_embedded = np.array([sentence_vector(text, embedding_model) for text in df_train['text_cleaned']])
X_test_embedded = np.array([sentence_vector(text, embedding_model) for text in df_test['text_cleaned']])

lr_model_embedded = LogisticRegression(max_iter=1000)
lr_model_embedded.fit(X_train_embedded, y_train)
embedded_predictions = lr_model_embedded.predict(X_test_embedded)

print("Classification pour le modèle avec embeddings pré-entraînés :\n")
print(classification_report(y_test, embedded_predictions))
```

```
Classification pour le modèle avec embeddings pré-entraînés :

              precision    recall  f1-score   support

  BACKGROUND       0.60      0.48      0.53      2663
 CONCLUSIONS       0.68      0.70      0.69      4426
     METHODS       0.82      0.87      0.85      9751
   OBJECTIVE       0.65      0.61      0.63      2377
     RESULTS       0.84      0.84      0.84     10276

    accuracy                           0.78     29493
   macro avg       0.72      0.70      0.71     29493
```

```
weighted avg       0.77      0.78      0.77     29493
```

```
In [ ]:
```