Classification of articles based on the applied research methodology

This notebook was created with the aim of supporting the methodological classification of scientific articles. By using it, the individual articles of the input article set are classified into two groups: **articles using qualitative research methodology** and **articles using quantitative research methodology**. The notebook is divided into 3 main chapters, which are as follows:

- **Basic settings**: Installation of the necessary packages and completion of the required basic settings for running.
- **Preprocessing of articles for classification**: Preparation of scientific articles for the classification model.
- **Classification of articles**: Classification of articles using XGBoost model.

To run the notebook, copy the articles to be classified into the *./articles* folder. The articles should be separate PDF files. After running the codes in the notebook, the classification result of the articles will be available in the *./output/article_classification_result.csv* file.

When using the notebook, please cite the scientific article describing the methodology:

Zsolt T. Kosztyán, Tünde Király, Tibor Csizmadia, Attila I. Katona, Ágnes Vathy-Fogarassy. Automated Research Methodology Classification Using Machine Learning. Journal, Vol. p.

1. Basic settings

Mount your Google Drive to the /content/drive directory in a Google Colab environment:

```
from google.colab import drive
drive.mount('/content/drive')
```

Install the packages required:

```
!pip install pdfplumber
!pip install tika
!pip install fitz
!pip install frontend
!pip install sympy
!pip install xgboost
!pip install pymupdf
!pip install openpyxl
!pip install nltk
import nltk
nltk.download('wordnet')
```

Set the working directory and paths:

```
import os

#working directory
os.chdir('/content/drive/set/your/working/directory/here/') #set the working directory

articlesPath = './articles'
outputFolder   = './output'
termsFile = './sources/terms.xlsx'
```

2. Preprocessing of articles for classification

Import the required packages:

```
import sys

# Add 'lib' directory to the system path
sys.path.append(os.path.abspath('lib'))

# Import the necessary classes
from logger.logger import Logger
from cache_handler.cache_creator import CacheCreator
from matrix_generator.matrix_generator import MatrixGenerator
```

Generate the document-term matrix, the input file for the classification model:

```
# Functions
def generate_cache_file():
    global cache_file

    c = CacheCreator(
            articles_location = articlesPath
    )
    c.start_generating()
    cache_file = '_article_cache.pkl'

def generate_document_term_matrix():
    m = MatrixGenerator(
        cache_file = cache_file,
        output_folder = outputFolder,
        words_file = termsFile,
        lemmatize = True,
        cutted = False,
        types_csv = '',
        generate_model = False,
        binarize = True
    )

    m.generate_matrix()


# Generate the document-term matrix
cache_file = ''
generate_cache_file()
generate_document_term_matrix()
```

3. Classification of articles

Classify the articles using the classification model developed:

```
import pandas as pd
import pickle
import xgboost as xgb

# Import document-term matrix from csv file
df = pd.read_csv(outputFolder + '/document_term_matrix.csv')

# Import the classification model
with open('./model/XGB_model.pkl', 'rb') as f:
  model = pickle.load(f)

# Set the input variables and run the model
X = df[df.columns[1:]]
y_pred = model.predict(X)

# Save the results into a dataframe and recode it
df_pred = pd.DataFrame(y_pred, columns=['predicted_class'])
result = pd.concat([df, df_pred], axis=1)
result = result[['title', 'predicted_class']]
result['predicted_class'].replace(0, 'quantitative', inplace=True)
result['predicted_class'].replace(1, 'qualitative', inplace=True)

# Count the articles in the classes
counts = result['predicted_class'].value_counts()
quantitative_count = counts.get('quantitative', 0)
qualitative_count = counts.get('qualitative', 0)

# Save the result into a csv file
result.to_csv(outputFolder + '/article_classification_result.csv', index=False)

# Print results
print('\nNumber of articles classified: ', df.shape[0])
print('\nNumber of articles based on quantitative research: ', quantitative_count)
print('Number of articles based on qualitative research: ', qualitative_count)
print('\nThe detailed results have been saved in the article_classification_result.csv file in the output folder.')
```