תמונה שמכילה טקסט, גופן, לוגו, גרפיקה

התיאור נוצר באופן אוטומטי

Software Engineering Department  
ORT Braude College

Capstone Project Phase B – 61999

**Metalanguage As an Interdisciplinary Classifier for Mathematics and Computer Science Fields**

Ilya Vorotyntsev | Ilya.Vorotyntsev@e.braude.ac.il

Aleksandr Pitkin | Aleksandr.Pitkin@e.braude.ac.il

Welcome to the **PDF-to-Text & Classification Pipeline**. This system converts PDF documents to text, cleans and preprocesses the text, and performs clustering and classification on it. This guide walks you through the **“successful” flow**—from installing the system to obtaining analysis results—without detailing error-handling scenarios.

**Audience and Purpose**

This guide is intended for **end users** who want to:

1. Convert PDFs into text files using multiple extraction libraries and/or OCR.
2. Run text preprocessing (cleaning, tokenization, lemmatization).
3. Cluster and/or classify the resulting text data using Doc2Vec and XLNet.

**System Requirements**

1. **Operating System**
   * Google Colab tested.
   * Can be adapted for Windows or macOS with appropriate package installation.
2. **Python**
   * Version 3.7+ recommended.
3. **Libraries** (installed via pip or in a virtual environment):
   * **PDF Extraction**:
     + PyMuPDF (fitz)
   * **OCR (optional)**:
     + Tesseract OCR and pytesseract
     + pdf2image, poppler-utils
   * **NLP & ML**:
     + nltk, gensim, scikit-learn, scikit-learn-extra, transformers, torch
     + pandas, numpy, matplotlib, seaborn

**Note**: Ensure Tesseract is installed on your system if you need to process scanned PDFs.

**Installation and Setup**

1. **Clone or Download the Project**
   * Copy the project folder to your cloud environment (e.g., Google Drive/Colab).
2. **Install Dependencies**
   * Using Google Colab, run the provided installation cells (e.g., !pip install ...).
3. **Arrange Your PDFs**
   * Place all PDF files to be processed in a dedicated folder, for example:
4. **Check Dictionaries (If Using Domain Terms)**
   * The system may refer to domain-specific (cs.txt, math.txt) and English (oxford.txt) dictionaries for text cleaning. Ensure these .txt files are accessible in the expected directory.

**5. Operating Instructions (Pipeline)**

**5.1 Step 1: Converting PDFs to Text**

1. **Open the PDF Conversion Script/Notebook**
   * Example: pdf\_to\_txt\_conversion.ipynb.
2. **Set the Folder Path**
   * In the code, locate a line such as:

folder\_path = '/content/drive/MyDrive/YourProject/PDF\_Files'

* + Adjust to your actual directory.

1. **Run the Conversion Function**
   * This automatically:
     + Reads each PDF.
     + Extracts text using **PyMuPDF**.
     + Saves output text files in subfolders. Example structure:

PDF\_Files/

|-- MyPDF.pdf

|-- PyMuPDF/MyPDF.txt

* + *If a PDF is scanned*, it may be queued for Tesseract OCR; run the OCR function (e.g., pdf\_to\_text\_tesseract(pdf\_path)) to process images and generate text.

**5.2 Step 2: Preprocessing Text**

1. **Open the Main Pipeline Notebook** (e.g., final-project.ipynb).
2. **Run preprocess\_data(...)**
   * The script reads the extracted .txt files and:
     + Cleans text (removes special characters, short words).
     + Tokenizes and lemmatizes.
     + Replaces domain terms with placeholders (e.g., <TERM>).
     + Splits text into **chunks** (default of 256 tokens).
   * A summary DataFrame (segmented\_data) is created with columns text and label\_id.

**5.3 Step 3: Clustering (Using Doc2Vec)**

1. **Train or Load Doc2Vec Model**
   * Run the function, for instance:

vector\_array = run\_doc2vec\_model(segmented\_data, path\_to\_save="...")

* + This generates vector embeddings of your text chunks.

1. **Run Clustering**
   * Choose one or more of the clustering methods:
     + **KMeans**: run\_kmeans\_clustering\_with\_doc2vec(segmented\_data, vector\_array, k=...)
     + **PAM (KMedoids)**: run\_pam\_clustering\_with\_doc2vec(...)
     + **DBSCAN**: run\_dbscan\_clustering\_with\_doc2vec(...)
     + **GMM**: run\_gmm\_clustering\_with\_doc2vec(...)
   * Each method:
     + Outputs cluster assignments.
     + Generates 3D PCA plots of data colored by predicted cluster and actual label.
     + Prints evaluation metrics.

**5.4 Step 4: Classification (Using XLNet)**

1. **Split Data**
   * The pipeline automatically splits data into training, validation, and test sets.
2. **Train or Load XLNet**
   * If you have a saved model, specify model\_path="..." to load it directly.
   * Otherwise, run the training cells:

run\_xlnet\_classification(segmented\_data, model\_path=None)

1. **Evaluate**
   * The script prints classification metrics (accuracy, precision, recall, F1).
   * Displays confusion matrix and example predictions with probabilities.