# PROGRAMMING ASSIGNMENT 5

**Issue Date : 16.05.2025 - Friday**
**Due Date : 01.06.2025 - Sunday (23:59)**
**Advisor:** Res. Asst. İsmail Furkan Atasoy
**Programing Language** : Python

The goal of this assignment is to use transformers-based models to provide an end-to-end solution for a real-life problem.

**Scenario:** You are managing a news website and have a collection of articles gathered from various online sources. While the category labels are missing, the titles are embedded within the article texts and have not been explicitly extracted. You aim to enrich these articles using AI models. First, you will perform a classification task using a BERT model to determine the category of each article. Then, you will use Google Gemini's free API service to separate and professionally rewrite the article and its headline. After completing these steps, each article will be enriched with a predicted category and a polished, publication-ready version of the article along with its headline.

## Dataset

To simulate the scenario, a part of the "AG News" dataset will be used. The original dataset contains 120k training examples and 7,600 test examples. For faster BERT training, the training set has been reduced to 30k examples. There are a total of 4 different categories in the dataset. For more information about the dataset, you can visit it's page on Hugging Face.

## Tools and Requirements

- Develop your project using Jupyter Notebook. The number of code cells is up to you.
- Above each code cell, use markdown cell to provide commentary and report on the concept or result you're about to demonstrate.
- Below each code cell, include the output of that cell. Make sure to save your Jupyter Notebook with all the output cells included before submitting your assignment.
- You may use Google Colab with T4 GPU or any environment of your choice to develop your project. However, ensure that your final Jupyter Notebook runs smoothly on Google Colab. (Remember that Colab's free tier has running time limitations.)
- Use "BERT Base Uncased" as your classifier model. The use of the transformers library and it's BertForSequenceClassification class is recommended.
- You should use Google Gemini or Gemma API services as the LLM. You can use Google AI Studio to configure your settings, obtain the template code of your settings to use it in your own project and obtain an API code. Note that these API services come with free usage limits. If you encounter errors due to quota limits, consider switching to another model.

# 1. Category Classificataion

In this task, your goal is to train a BERT model using the provided train.csv dataset and evaluate its performance using test.csv. You are expected to fine-tune the BERT Base Uncased model. Since the dataset contains 4 classes, your model should be capable of handling a 4-class classification problem.

Before starting the modeling process, it's important to understand the dataset. After loading the data, perform a basic exploratory data analysis (EDA), then report and print your findings. This may include checking class distributions, text length statistics, etc.

You are free to apply any preprocessing techniques on the training data if you think it is necessary. If you require a validation set, you can split the training set accordingly. Make sure that the test.csv file is not used during training. It should only be used after training is complete for final evaluation.

Train your model using train.csv (or a derived training/validation set). After training, it is recommended to save the model so you don't have to retrain in case of interruptions.

Once the model is trained, evaluate it on test.csv by calculating the F1 score. Try improving your F1 results by experimenting with data processing or hyperparameters. You are expected to achieve the best possible F1 score. (For guidance, an F1 score greater than 0.90 is expected for this dataset)

The ultimate goal of this section is to develop a high-performing classification model using BERT Base Uncased and to report the experiments conducted throughout the process. You are encouraged to experiment with the training data and/or model hyperparameters to achieve the best possible performance.

You should have a saved BERT model that can classify a news article into one of 4 categories. As a final test, use the model to classify the first 5 samples from test.csv and print the predictions and true labels (categories).

# 2. Title and Well-Written Article Generation

In this section, your goal is to generate (or extract) a title and a polished version of a given news article using Google's LLM API services. A polished article refers to a well-written, professionally structured version of the original text, produced by the LLM.

You can use the Google AI Studio platform for quick experimentation. However, to integrate it into your own code, you will need to obtain an API key and implement it within your script.

You are expected to produce two separate outputs (title and well-written article) from a single article. You may use two separate API calls, or use a single prompt with single API call.

Make sure you can extract the title and the well-written article easily from the LLM's response. Avoid extra model-generated phrases like "Here is an example title below:". You can use one-shot or few-shot prompting techniques to ensure that the LLM returns the title and a well-written article in a specific format, making it easy to extract the desired parts using tools like regex.

## 3. Combining 2 Tasks

In this section, you are expected to combine the work from Section 1 and Section 2. When an article is provided, you should first perform category prediction using the pre-trained and saved BERT model. Obtain the corresponding category names for the 4 labels in the dataset from the dataset's page, and store the prediction as text rather than a number. Then, use the LLM API to generate a title and a well-written version of the article. Afterward, extract the title and well-written article separately. Finally, print them to the user in a dictionary format with the keys "category", "title", and "article" and their corresponding values.

## Expectations and Hints

- Please divide your code into separate code cells where appropriate. For each cell, include relevant markdown annotations explaining your comments and experiments. Also, make sure each code cell has its corresponding outputs.
- Briefly analyze the dataset and include necessary outputs and observations in markdown format.
- If you perform any processing on the train.csv file, make sure to show it and explain why it was necessary.
- While training the BERT model, you may experiment with parameters such as learning rate, number of epochs, weight decay, learning rate scheduler, loss function or more. You do not need to test all of them, but experiment with the ones you find important. Then, document in markdown what you tried, the results you obtained, and the challenges you encountered in finalizing your code.
- To ensure efficient BERT training, make sure to utilize hardware acceleration (such as GPU via CUDA). For efficiency, you can increase the batch size as long as your hardware memory allows. You may also consider reducing BERT's maximum token length from 512 to 256 or less, but make sure that doing so does not significantly affect the F1 scores.
- After training the BERT model, evaluate it using the test.csv file and compute the F1 score. Also, print the result (F1 score). In addition, display the first 5 examples with both the predicted label and the true label.
- Keep in mind that the LLM may not always return results in the expected format. Changing the prompt, adjusting the temperature and top-p values can be helpful for this purpose. Do your best to obtain high-quality outputs as you expected.
- At the end of your code, take the last 5 samples from the train.csv file and process them through all the steps in your pipeline. Finally, print the resulting output as a list of dictionaries. Each dictionary should contain the following keys: category (as a text label, not a numerical ID), title, and article (the polished version generated by the LLM).
- Your code will be evaluated based on markdowns and outputs. However, in some cases, if the code needs to be executed, ensure that your entire code is ready to run.

## Submission

- You will submit your programming assignment using the **HADI** system. You have to upload a single **zip file** (`.zip`, `.gzip` or `.rar`) holding a **Jupyter notebook**.
- The name of your solution file should be: `hw05_NameLastname.ipynb` replacing the `NameLastname` with your actual first name and last name. The name of your zip file should match `hw05_NameLastname` with a corresponding extension (`.zip`, `.gzip`, or `.rar`).
- Don't forget to remove your Google API key which is special to you, from the Jupyter notebook before submitting the assignment.

**Late Policy:**

- You must submit your programming assignment **before its due date**.
- You can also submit your assignment up to **three days late**, but with a penalty:
  - **1 day late:** 10% penalty
  - **2 days late:** 20% penalty
  - **3 days late:** 30% penalty

**DO YOUR PROGRAMMING ASSIGNMENTS YOURSELF!**

- **Do not share your programming assignments with your friends.**
- **Do not complete your entire assignment using AI tools. Comment on the parts where you have received support from AI tools.**
- **Cheating will be punished.**