# **Patent Classification with Fine-Tuned BERT**

## **Overview**

This documentation provides an in-depth guide for the patent classification application using a fine-tuned BERT model. The application performs the following tasks:

1. Loads a dataset of patents.
2. Prepares the dataset for classification.
3. Fine-tunes a pre-trained BERT model on the dataset.
4. Allows users to interact with the model training process via a Streamlit web application.

## **Requirements**

To run this application, you need the following libraries:

* **Streamlit**: For creating the web application interface.
* **Transformers**: For accessing pre-trained models and tokenizers.
* **Datasets**: For handling datasets.
* **Pandas**: For data manipulation.
* **Scikit-learn**: For data splitting.
* **Requests**: For fetching data from the web.

You can install these dependencies using pip:

pip install streamlit transformers datasets pandas scikit-learn requests

### **Functions**

#### **load\_data()**

* **Purpose**: Downloads and loads the dataset from a URL.
* **Returns**: A Pandas DataFrame containing the patent data.

#### **load\_tokenizer\_and\_model(model\_name, num\_labels)**

* **Purpose**: Loads a tokenizer and a pre-trained BERT model.
* **Parameters**:
  + model\_name: The name of the pre-trained model (e.g., "bert-base-uncased").
  + num\_labels: The number of labels for classification.
* **Returns**: A tuple containing the tokenizer and model.

#### **prepare\_data(df, tokenizer)**

* **Purpose**: Prepares the dataset for classification by tokenizing text and creating a Dataset object.
* **Parameters**:
  + df: A Pandas DataFrame containing the patent data.
  + tokenizer: The tokenizer used to convert text into model-compatible format.
* **Returns**: A tuple containing:
  + A Dataset object with tokenized data.
  + The number of unique labels.

#### **main()**

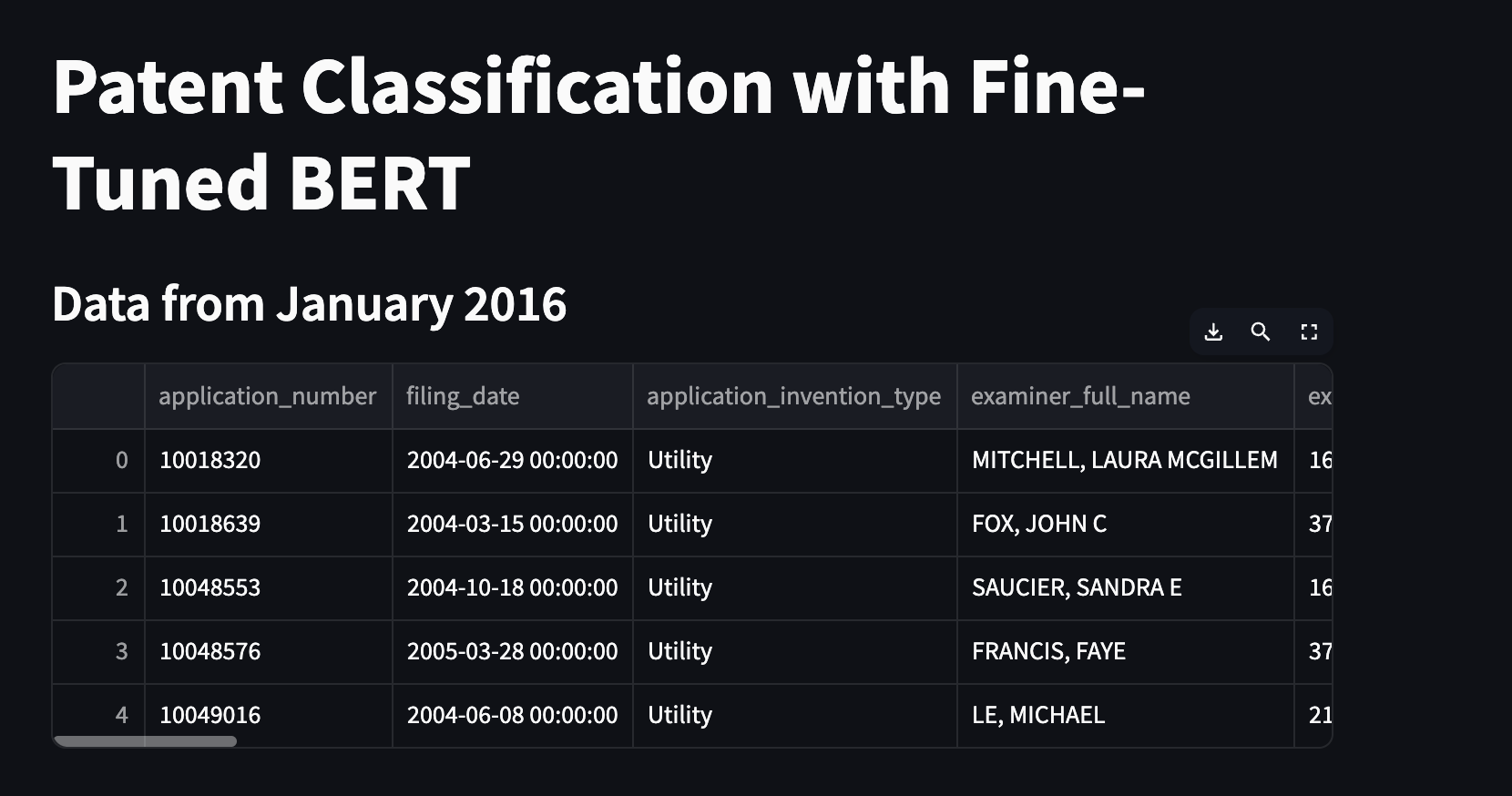
* **Purpose**: Defines the Streamlit web application interface and functionality.
* **Components**:
  + **Data Loading**: Fetches and displays a sample of the patent dataset.
  + **Model Preparation**: Loads the tokenizer and model, prepares the dataset, and displays sample data.
  + **Training**: Fine-tunes the BERT model and provides options to view training progress and save the model.
  + **Model Files**: Provides options to view the contents of the saved model directory and the files within it.

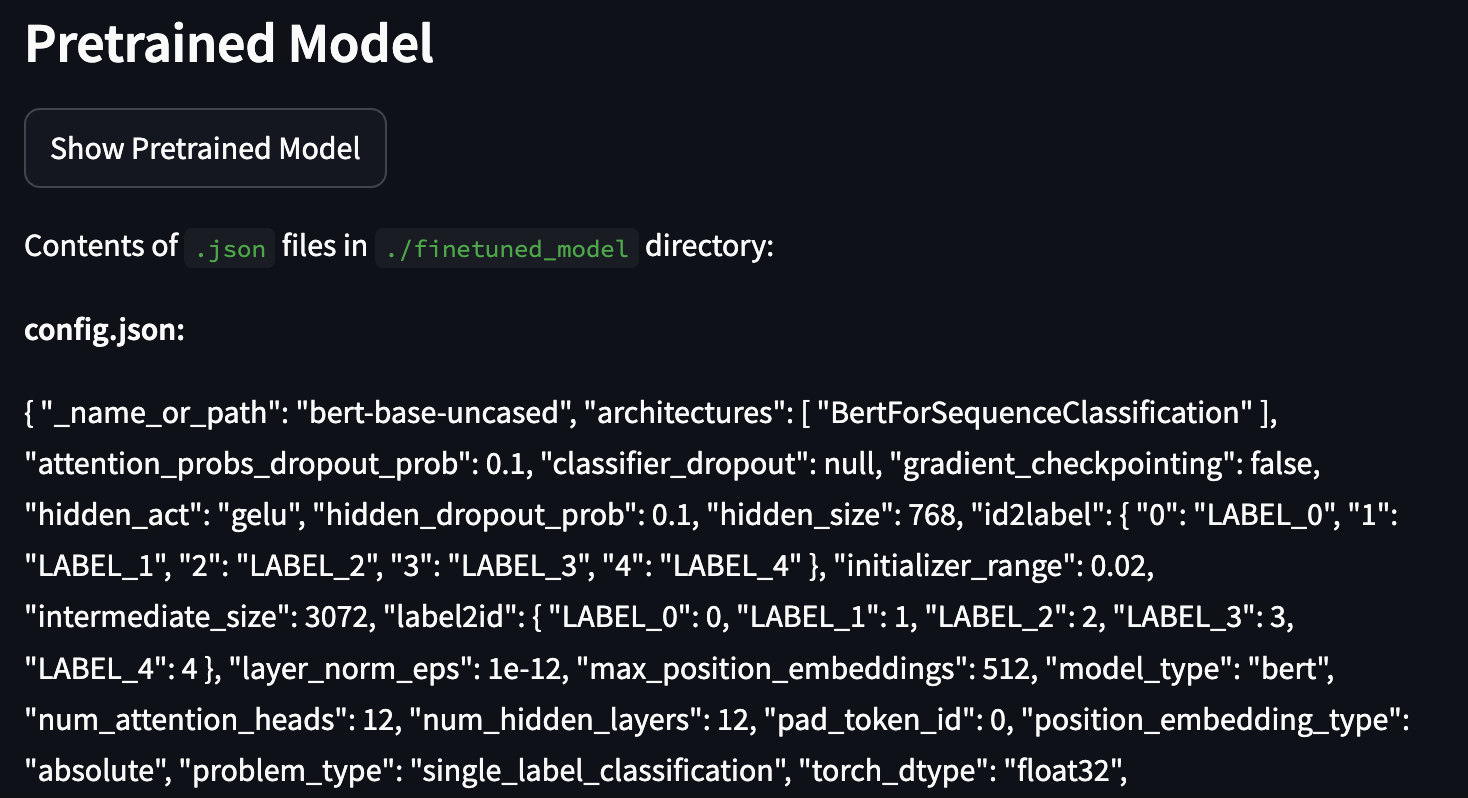
### **Model Training**

* **Training Arguments**:
  + output\_dir: Directory where the model checkpoints and final model are saved.
  + evaluation\_strategy: Defines when to evaluate the model (here, after each epoch).
  + learning\_rate: The learning rate for the optimizer.
  + per\_device\_train\_batch\_size: Batch size for training.
  + per\_device\_eval\_batch\_size: Batch size for evaluation.
  + num\_train\_epochs: Number of epochs for training.
  + weight\_decay: Weight decay for regularization.
* **Trainer**: A class from the transformers library that simplifies the training and evaluation of models.

### **Running the Application**

1. Save the code to a file named app.py.
2. Open a terminal and navigate to the directory containing app.py.
3. Run the Streamlit app using the command:  
   streamlit run app.py
4. A new browser window should open displaying the web application. If it doesn’t open automatically, you can access it by navigating to http://localhost:8501 in your web browser. Refer to some snaps.
5. Application link : https://huggingface.co/spaces/talktorhutika/patentdataset-finetune





## **Troubleshooting**

* **No Data Displayed**: Ensure that the dataset URL is correct and accessible. Check your internet connection.
* **Training Issues**: Verify that the model and tokenizer are correctly loaded. Ensure that you have enough computational resources for training.
* **File Not Found**: Make sure the specified directory exists and contains the expected files. Check for typos in file paths.

## **Conclusion**

This application provides a user-friendly interface for fine-tuning a BERT model on a patent classification task. It demonstrates how to handle real-world data, prepare it for model training, and interact with the training process using Streamlit. This setup can be further customized and extended for different datasets and classification tasks.