News Articles Sorting

High Level Design (HLD)

Author: Tanjina Proma

Date: 1/08/2024

Contents

Abstract

The text classification task using BERT base uncased transformers on news articles involves employing a state-of-the-art natural language processing model to accurately categorize news content. BERT (Bidirectional Encoder Representations from Transformers) processes text by capturing contextual information bidirectionally, enabling a nuanced understanding of language. In this task, the model learns to discern between various news categories such as politics, sports, technology, and more. By fine-tuning BERT's pre-trained weights on labeled news article data, it adapts to recognize patterns and semantic nuances within the text. Leveraging its deep contextual understanding, BERT assigns probabilities to different categories, enabling precise classification. The model's effectiveness lies in its ability to comprehend the intricacies of language, improving the accuracy and efficiency of news article classification, thereby enhancing information retrieval and organization in diverse domains.

The primary objective of this project is to develop an automated system capable of accurately classifying news articles into predefined categories using state-of-the-art NLP models. The project involves fine-tuning bert-base-uncased on the collected dataset to adapt its knowledge to the specific classification task. This step enables the model to learn the nuances and patterns within the news articles.

1 Introduction

1.1 Why the HLD?

High-Level Design (HLD) for text classification tasks with BERT base uncased transformers in news articles serves several crucial purposes. Firstly, it outlines the architecture and components involved in the classification pipeline, such as data preprocessing, model selection, and evaluation metrics. This blueprint ensures a structured approach to implementing BERT, optimizing its performance, and accommodating scalability for large datasets. Secondly, the HLD delineates the integration of BERT within the existing infrastructure, ensuring compatibility and efficient utilization of resources. It also defines strategies for model fine-tuning, hyperparameter optimization, and handling diverse news article categories. Additionally, the HLD encompasses strategies for model monitoring, versioning, and updating to maintain classification accuracy as news content evolves. Overall, the HLD serves as a roadmap, guiding the implementation, deployment, and maintenance of the BERT-based text classification system for news articles, ensuring robustness and adaptability in real-world scenarios.

1.2 Scope

The High-Level Design (HLD) for text classification using BERT base uncased transformers on news articles encompasses architectural planning, data flow, and system components. It defines the overall structure of the classification system, including the integration of BERT, data preprocessing pipelines, model training, and inference mechanisms. HLD outlines the system's scalability, handling of large volumes of news data, and deployment strategies. Additionally, it details the incorporation of multiple news categories, model evaluation methods, and potential extensions for continuous improvement, ensuring a robust, efficient, and adaptable framework for accurate news article classification using BERT base uncased transformers.

2 General Description

2.1 Product Perspective

The text classification system is designed to categorize news articles into predefined classes based on their content. It serves as an intelligent information filtering tool, aiding users in accessing relevant news efficiently.

2.2 Problem Statement

In the vast landscape of news articles, users often struggle to find information relevant to their interests. Traditional keyword-based searches may yield inaccurate or incomplete results. The challenge is to develop a system that can accurately classify articles into categories, enhancing user experience and content discovery.

2.3 Proposed Solution

The proposed solution involves the integration of a transformer-based BERT model for its superior natural language understanding capabilities. The system will undergo a training phase on labeled data to fine-tune the pre-trained BERT model for news classification. The trained model will be deployed to provide real-time predictions on new articles.

2.4 Further Improvements

Future enhancements may include:

- Continuous model retraining to adapt to evolving language patterns.
- Integration of user feedback for personalized recommendations.
- Multimodal analysis by incorporating image and video content for more comprehensive classification.

2.5 System Specifications:

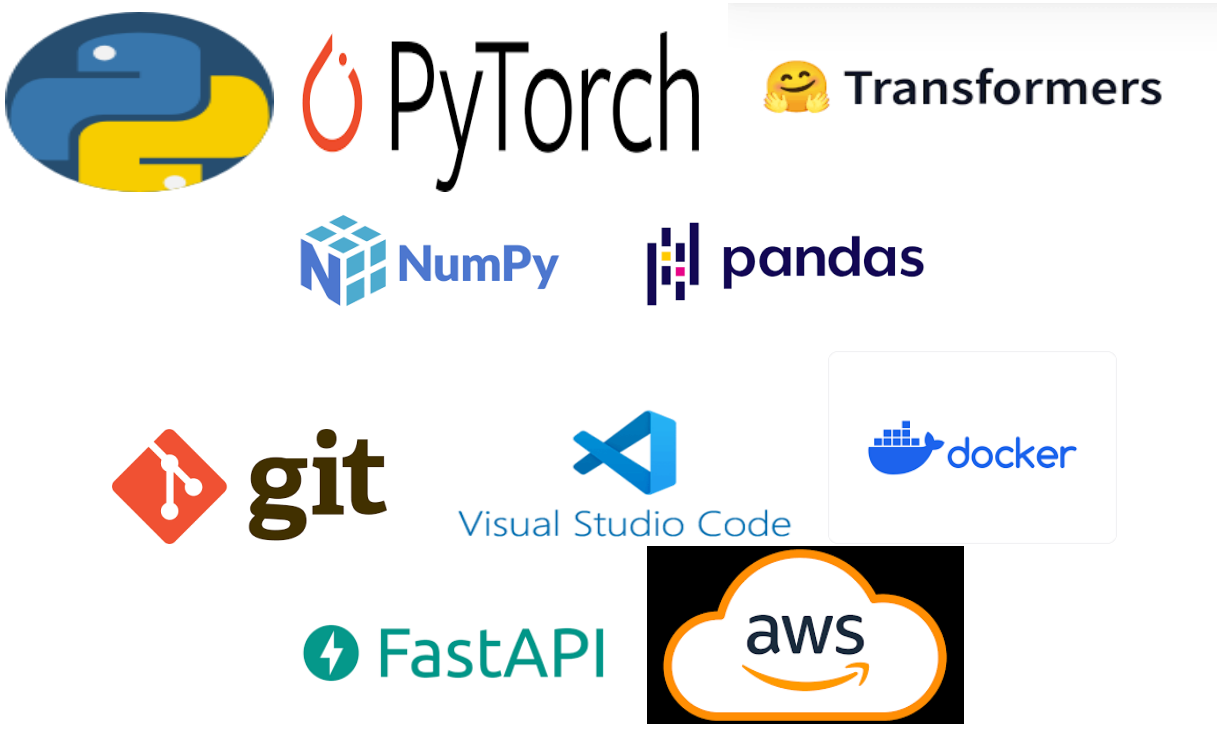
Hardware Requirements:

- **Processor:** High-performance CPU/GPU (e.g., multi-core CPU, NVIDIA GPU for faster training).
- **Memory:** Significant RAM for handling large models and datasets.
- **Storage:** Adequate storage for model files, datasets, and intermediate results.

Software Requirements:

- **Python Environment:** Python 3.8 for coding.
- **Machine Learning Libraries:** PyTorch libraries supporting Transformers.
- **Transformer Models:** Hugging Face's **transformers** for utilizing pre-trained model BERT.
- **Data Processing:** Pandas, NumPy, and other data manipulation libraries.

- **Development Tools:** Jupyter Notebooks, IDEs VS Code
- **Dependencies:** Ensure compatible versions of all required packages.



- **Version Control:** Git for tracking code changes.
- **CI/CD:** GitHub Actions.
- **Containerization:** Docker
- **Deployment:** AWS ECR, EC2.

2.6 Data Requirements

Dataset: Labeled dataset of news articles with corresponding categories for model training. For this project BBC News Data has been used.

File descriptions:

- BBC News Train.csv - the training set of 1490 records
- BBC News Test.csv - the test set of 736 records
- BBC News Sample Solution.csv - a sample submission file in the correct format

Data fields

- ArticleId - Article id unique # given to the record
- Article - text of the header and article
- Category - category of the article (tech, business, sport, entertainment, politics)

Data link:

https://www.kaggle.com/c/learn-ai-bbc/data

2.7 Constraints

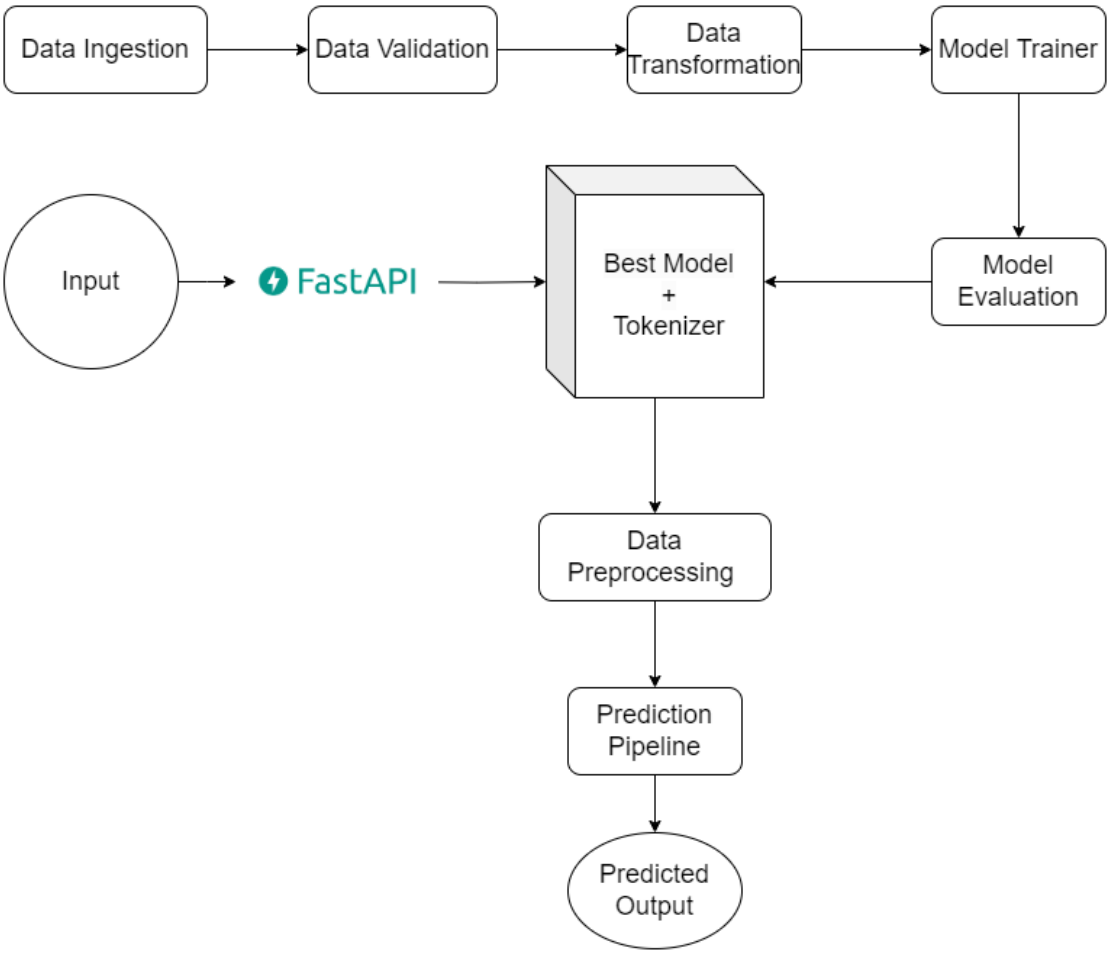
- Computational Resources: Availability of GPU resources for efficient model training.
- Latency: Real-time inference may be subject to latency constraints.
- Data Privacy: Adherence to data privacy regulations when handling potentially sensitive news content.

2.8 Assumptions

- Training Data Quality: Assumes the availability of a high-quality labeled dataset for effective model training.
- Model Generalization: Assumes the trained model can generalize well to classify diverse news articles accurately.

3 Design Details

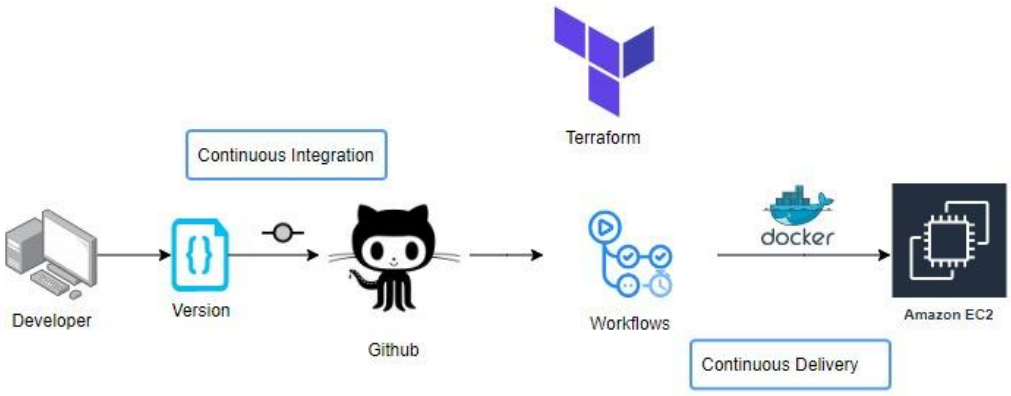
Flowchart



3.1 Process Flow

1. **Data Ingestion:**
    - Collect news articles from Kaggle dataset link which included dataset created from BBC News data.
2. **Data Preprocessing:**
    - Tokenize the cleaned text into BERT-compatible tokens.

3. **Model Integration:**
    - Utilize a pre-trained BERT model i.e. bert-base-uncased and add a classification layer on top.
    - Fine-tune the model on labeled news articles for the desired classification task.
4. **Training:**
    - Split the preprocessed data into training, validation, and test sets.
    - Train the integrated BERT-based model on the training set.
5. **Evaluation:**
    - Assess the model's performance using evaluation metrics on the validation set.
    - Fine-tune hyperparameters based on evaluation results.
6. **Inference:**
    - Deploy the trained model to make predictions on new, unseen articles.
    - Classify incoming articles into predefined categories using the deployed model.
7. **Deployment Process**
    - **API endpoint:** FastAPI serves as access points for interacting with the BERT model through APIs, allowing users to input text data and receive processed results. Accepts text inputs and categorizes them into predefined classes or labels, like news topics (politics, sports, etc.) based on the BERT model's classification capabilities.
    - **Containerization**: Using Docker to package applications and their dependencies into containers, ensuring consistency across various environments, such as development, testing, and production.
    - **Continuous Integration/Continuous Deployment (CI/CD):** GitHub Actions automates CI/CD pipelines by triggering workflows on events, facilitating testing, building, and deploying code changes within a repository.
    - **Cloud :** AWS ECR (Elastic Container Registry) is a service for storing, managing, and deploying Docker container images. EC2 (Elastic Compute Cloud) provides scalable virtual servers for various computing needs on AWS.



3.2 Event Log

- **Data Ingestion Log:** Records the sources of collected news articles.
- **Data Validation Log:** checks information about files ingested are correct or not.
- **Data Transformation Log:** Stores information about the preprocessing steps applied to the articles.
- **Training Log:** Records training progress, including metrics like loss and accuracy.
- **Evaluation Log:** Stores evaluation metrics and results.
- **Inference Log:** Logs predictions made by the deployed model along with confidence scores.

3.3 Error Handling

- **Data Ingestion Errors:** Log failed attempts to collect or preprocess articles, notify administrators, and retry mechanisms.
- **Data Validation Errors:** checks information about files ingested are correct or not.
- **Preprocessing Errors:** Handle tokenization or formatting failures, logging details for debugging.
- **Training Errors:** Monitor for convergence issues, log and alert for anomalies in training behavior.

- **Evaluation Errors:** Alert on unexpected evaluation metric behavior, triggering reevaluation.
- **Inference Errors:** Handle errors in inference due to model or input issues, logging details for debugging and improvement.

4 Performance

4.1 Reusability

- **Modular Components:** Design the system with modular components for easy integration of other transformer models or classification tasks in the future.
- **Configurability:** Create configurations to adjust hyperparameters, model architecture, and input formatting for different text classification needs.

4.2 Application Compatibility

- **API Design:** Implement a flexible API interface allowing compatibility with various applications and platforms.
- **Output Format:** Ensure the output format of the classification results is easily integrable with downstream applications or systems.

4.3 Resource Utilization

- **Efficient Batching:** Optimize input data batching during training and inference to maximize GPU utilization.
- **Memory Management:** Employ memory-efficient techniques for tokenization and inference to reduce resource consumption.

4.4 Deployment

- **Scalability:** Design the deployment architecture to handle increased user demand by horizontally scaling resources.
- **Containerization:** Use containerization (e.g., Docker) for consistent deployment across different environments.
- **Monitoring and Auto-scaling:** Implement monitoring tools to track system performance and automatically scale resources based on usage patterns.

5 Dashboards

5.1 KPIs (Key Performance Indicators)

- **Model Performance Metrics:** Tracking key metrics like accuracy, precision, recall, and F1-score obtained during model evaluation on the validation set.
- **Inference Metrics:** Monitor inference latency, throughput, and error rates for real-time performance assessment.
- **Training Metrics:** Keep track of training loss, convergence, and learning rate for assessing model training progress.

6 Conclusion

Employing the BERT Base Uncased transformer model for news article classification yields robust results. Its fine-tuned capabilities demonstrate exceptional accuracy in categorizing articles, enhancing comprehension and organization of vast news data. This model's proficiency in capturing nuanced context and semantics ensures precise classification across diverse news topics. The BERT Base Uncased transformer, through its contextual understanding, significantly advances text classification tasks, offering a potent tool for efficient and accurate news article categorization in real-world applications.