IAB Article Category Classification

This project aims to classify articles into one of 26 predefined IAB categories using various machine learning techniques. The dataset consists of a collection of articles with their respective IAB category labels, and the task is to predict the category of unseen articles.

Project Structure

- **train.csv**: Contains the text of articles and their respective target IAB category labels.
- **test.csv**: Contains only the text of articles (without labels) for which predictions need to be made.
- **sample_submission.csv**: A sample submission file showing the expected format for the results.

Models Used

1. Naive Bayes and XGBoost:

- **Preprocessing**:
  - **CountVectorizer** and **TFIDF Transform** were used for feature extraction.
  - Text cleaning steps included:
    - Removing stopwords
    - Stemming (using a stemming algorithm)
  - The models were trained on the preprocessed text using the Naive Bayes and XGBoost classifiers.
- **Results**: Baseline performance with traditional machine learning methods.

2. DistilBERT Fine-Tuning:

- The raw text data (without text cleaning or stemming) was used to fine-tune a `DistilBERT` model.
- The model was trained for multi-class classification (26 categories).
- **Preprocessing**:
  - Tokenization was handled using the DistilBERT tokenizer (`distilbert-base-uncased`).
  - Tokenization was done without stopword removal or stemming since BERT models tend to perform better with raw text.
- **Memory Management**:
  - Due to the large dataset size (over 690,000 rows), a custom `iterative DataLoader` was implemented using PyTorch and Hugging Face's `datasets` library to handle memory constraints effectively.
- **Training Hardware**:
  - Model was trained on **GPU T4 x2** for almost 1 hour on Kaggle to expedite the process.
- **Model**: {https://huggingface.co/LakshitKava/FibeVibeToHack2.0-IAB-DistillBert}

Training Details

Hyperparameters and Trainer Arguments

Argument	Value	Description
output_dir	"results"	Directory to store the training outputs.
overwrite_output_dir	True	Whether to overwrite the output directory.
num_train_epochs	2	Number of epochs to train the model.
per_device_train_batch_size	32	Batch size for training.
per_device_eval_batch_size	32	Batch size for evaluation.
warmup_steps	500	Number of warmup steps to gradually increase learning rate.
weight_decay	0.01	Weight decay to apply to the optimizer.
logging_dir	"logs"	Directory to store logs for monitoring.
logging_steps	5000	Number of steps after which to log training progress.
save_steps	5000	Number of steps after which to save a checkpoint.
learning_rate	1e-5	Learning rate for the optimizer.
do_train	True	Whether to perform training.
do_eval	True	Whether to perform evaluation on the validation set.
seed	42	Seed for reproducibility of

Argument	Value	Description
		results.
`gradient_accumulation_steps`	`1`	Number of steps for gradient accumulation to mimic larger batch sizes.
`fp16`	`True`	Enable mixed precision training to speed up computations and reduce memory usage.
`push_to_hub`	`True`	Whether to push the model to Hugging Face Hub.
`hub_model_id`	`"<Your Model Id On HuggingFace>"`	Model ID on Hugging Face Hub for version control.
`report_to`	`"all"`	Report logs to both TensorBoard and Hugging Face Hub.

Model Architecture

Component	Value	Description
`model`	`DistilBERT`	The base transformer model used for classification.
`num_labels`	`26`	Number of target labels (IAB categories) for classification.
`tokenizer`	`DistilBERTTokenizer`	Tokenizer used for input text processing.

Trainer Arguments

Argument	Value	Description
`model`	`DistilBertForSequenceClassification`	The pre-trained DistilBERT model used for fine-tuning.
`args`	`training_args`	Training configuration defined in the TrainingArguments section.
`compute_metrics`	`compute_metrics`	Function to compute performance metrics such as F1 score.
`train_dataset`	`train_dataset`	The tokenized training dataset.
`eval_dataset`	`test_dataset`	The tokenized evaluation dataset (test set).

DataLoader Configuration

Parameter	Value	Description
`batch_size`	`32`	Batch size for both training and evaluation.
`shuffle`	`True`	Shuffle the dataset during training.
`num_workers`	`4`	Number of CPU workers for loading data in parallel.

Inference

```
model_name = "LakshitKava/Fibe_IAB_DB_v2"
tokenizer_name = "distilbert-base-uncased"
```

```
from transformers import DistilBertForSequenceClassification,
DistilBertTokenizerFast
model =
DistilBertForSequenceClassification.from_pretrained("LakshitKava/FibeVibeToH
ack2.0-IAB-DistillBert")
tokenizer = DistilBertTokenizerFast.from_pretrained("distilbert-base-
uncased", num_labels=26)


class NewsDataset(torch.utils.data.Dataset):
    def __init__(self, texts, tokenizer, max_length=512):
        self.texts = texts
        self.tokenizer = tokenizer
        self.max_length = max_length

    def __getitem__(self, idx):
        encoding = self.tokenizer(
            self.texts[idx],
            truncation=True,
            padding="max_length",
            max_length=self.max_length,
            return_tensors="pt"
        )
        item = {key: val.squeeze(0) for key, val in encoding.items()}
        return item

    def __len__(self):
        return len(self.texts)

infer_dataset = NewsDataset(infer_text, tokenizer)
infer_loader = torch.utils.data.DataLoader(infer_dataset, batch_size=32,
shuffle=False)

model.eval()
predictions = []
with torch.no_grad():
    for batch in tqdm(infer_loader, desc="Inference Progress"):
        batch_gpu = {key: val.to('cuda') for key, val in batch.items()}

        # Run model inference
        outputs = model_gpu(**batch_gpu) ## GPU
        logits = outputs.logits

        # Get predictions (taking the argmax)
        batch_predictions = torch.argmax(logits, dim=1)
        predictions.extend(batch_predictions.cpu().numpy())

## Convert predictions back to text via mapper
```

Index Mapping List

```
['academic interests',
 'books and literature',
 'healthy living',
 'careers',
 'news and politics',
 'shopping',
 'style and fashion',
 'family and relationships',
 'business and finance',
 'automotives',
 'pharmaceuticals, conditions, and symptoms',
 'arts and culture',
 'sports',
 'pets',
 'hobbies and interests',
 'real estate',
 'food and drinks',
 'home and garden',
 'video gaming',
 'movies',
 'travel',
 'personal finance',
 'technology and computing',
 'music and audio',
 'television',
 'health']
```

Evaluation Metric

The model is evaluated based on the **weighted F1 score** metric, which is calculated as follows:

```
from sklearn.metrics import f1_score
f1 = f1_score(actual_labels, predicted_labels, average='weighted')
```

Further Improvements

Data Augmentation: Adding more diverse training examples for underrepresented categories. Hyperparameter Tuning: Exploring different learning rates, batch sizes, and epochs for better performance. Other Models: Comparing performance with other transformer models like BERT, RoBERTa, LLama-2 etc.

Results

The DistilBERT model showed significant improvement over traditional Naive Bayes and XGBoost models, especially on handling the complexity of raw text data with more context-aware embeddings.

Acknowledgements

Thanks to Fibe for this amazing opportunity and HackerEarth for amazing platform.
Thanks to the Hugging Face library for providing easy access to pre-trained models like DistilBERT and to the Interactive Advertising Bureau (IAB) for the dataset.