Language Translation Project Documentation

Objective

To build a multilingual translation system using pretrained models for translating between English, French, Spanish, and Italian. The project includes dataset preparation, model training, and a user-friendly Streamlit app for performing translations.

1. Dataset Preparation

**Datasets**

The datasets used are translations between:

* English to French (en-fr)
* English to Spanish (en-es)
* English to Italian (en-it)
* French to English (fr-en)
* Spanish to English (es-en)
* Italian to English (it-en)

Each dataset consists of sentences in the source language (en, fr, es, it) paired with their translations in the target language.

**Dataset Splitting**

Each dataset is split into:

* Train Set: 90% of the dataset for model training.
* Validation Set: 5% of the dataset for tuning hyperparameters.
* Test Set: 5% of the dataset for evaluating model performance.

**Preprocessing**

* Tokenization: The text is tokenized using Hugging Face’s AutoTokenizer with padding and truncation applied.
* Uniform Length: Sentences are padded or truncated to a maximum length of 128 tokens.
* Input and Label Preparation: Tokenized sentences from the source language are used as inputs, and tokenized translations are used as labels.
* Batch Processing: Preprocessing is applied to the dataset in batches using the .map() method for efficiency.

2. Model Training

**Pretrained Model**

The project uses the Helsinki-NLP/opus-mt models from Hugging Face’s Transformers library, which are pretrained for various language pairs.

**Training Steps**

Each language pair model was fine-tuned using the following steps:

Loading Tokenizer and Model

* Loaded the pretrained tokenizer and model using AutoTokenizer and AutoModelForSeq2SeqLM.

Defining Data Collator

* Used DataCollatorForSeq2Seq to dynamically pad batches during training.

Training Arguments

* Defined training arguments using Seq2SeqTrainingArguments, specifying parameters such as learning rate, batch size, and evaluation strategy.

Trainer Initialization

* Used Seq2SeqTrainer to handle the training process, passing in the model, dataset, tokenizer, data collator, and training arguments.

Evaluation

* Evaluated the fine-tuned models on the test set to calculate metrics like BLEU scores.

Saving Models

* Saved the fine-tuned models and tokenizers for all language pairs (en-fr, en-es, en-it, fr-en, es-en, it-en).

3. Jupyter Notebook Cells in Detail

Cell 1: Library Installation

Installed required libraries: transformers, datasets, and sentencepiece.

Cell 2: Import Statements

Imported necessary libraries, including pandas, train\_test\_split, and modules from Hugging Face.

Cell 3: Data Loading

Loaded the datasets into Hugging Face’s Dataset format.

Splitted the data into train, validation, and test sets.

Cell 4: Tokenizer Loading

Loaded the pretrained tokenizer for each language pair.

Cell 5: Preprocessing Function

Defined a function to tokenize input and target sentences, with padding and truncation.

Applied the preprocessing function to the datasets.

Cell 6: Model Loading

Loaded the pretrained model for translation tasks.

Cell 7: Data Collator Definition

Defined a data collator for padding during batching.

Cell 8: Training Arguments

Specified training parameters such as learning rate, batch size, and evaluation steps.

Cell 9: Model Training

Initialized Seq2SeqTrainer and trained the model.

Cell 10: Evaluation

Evaluated the trained models using metrics like BLEU scores.

Cell 11: Saving Models

Saved the trained models and tokenizers to a directory.

Cell 12: Inference

Defined a function for translating sentences using the trained models.

Tested the function with example sentences.

4. Streamlit Application (app.py)

**Objective**

The Streamlit app provides an interactive interface for users to perform text and file translations.

**Features**

* Language Selection
* Users can select the input and target languages.
* Supported pairs: en-fr, en-es, en-it, fr-en, es-en, it-en.

**Input Methods**

Manual Entry:

* Users can input multi-line text directly.

File Upload:

* Users can upload .txt files for batch translation.

**Translation Process**

Model Loading:

* Dynamically loads the appropriate model and tokenizer based on selected language pairs.

SentencePiece Tokenizers:

* Uses SentencePiece for additional preprocessing.

Translation Function:

* Translates text using beam search.
* Computes confidence scores for translations.

Output

* Displays translated text and average confidence scores.
* Provides a download link for translated files.

**Code Details**

Model and Tokenizer Loading

* The load\_model\_and\_tokenizer() function loads pretrained models and tokenizers for selected language pairs.

Translation Logic

* The translate() function handles tokenization, inference, and decoding.
* Beam search ensures quality translations.

Batch Translation

* The translate\_multiline() function processes multi-line input or file content line-by-line.

**Streamlit UI**

* User-friendly interface for selecting languages, input methods, and displaying results.
* Provides real-time feedback with confidence scores.

Supported Language Pairs

1. English to French (en-fr)
2. English to Spanish (en-es)
3. English to Italian (en-it)
4. French to English (fr-en)
5. Spanish to English (es-en)
6. Italian to English (it-en)