Viðhorfsgreining á íslenskum texta

(Sentiment-Analysis on Icelandic text)

Instructions

Setting Up a Virtual Environment

To avoid conflicts with other projects or system-wide Python packages, it's recommended to set up a virtual environment for this project. Here's how to do it:

Prerequisites

- Python 3.x (Ensure Python 3 is installed on your system.)

Creating a Virtual Environment

1. **Navigate to Your Project Directory**: Open a terminal or command prompt and navigate to the root directory of this project.
2. **Create a Virtual Environment**: Run the following command to create a virtual environment named `env` (you can choose any name you prefer):

   ```
   python -m venv env
   ```

   This command creates a new directory `env` within your project where all dependencies will be installed.
3. **Activate the Virtual Environment**:

- On **Windows**, run: `.\env\Scripts\activate`
- On **macOS or Linux**, run: `source env/bin/activate`

Installation of Dependencies

To run the scripts, you need to install the dependencies. Follow the steps below to set up your environment.

Prerequisites

- Python 3.x (Make sure Python 3 is installed on your system.)

Installation Steps

1. Ensure Python 3.x is installed.
2. Install Requirements: `pip install -r requirements.txt`
3. Install PyTorch: It's **recommended** that you use the GPU version (CUDA) of PyTorch, visit the PyTorch Get Started page, select your preferences, and run the provided installation command.

Machine-translate

This section provides instructions for using the machine translation scripts included in this project: `translate_google.py` and `translate_mideind.py`. These scripts are used for translating text data into Icelandic for sentiment analysis.

Using `translate_google.py`

Overview `translate_google.py` is a Python script for translating text data using Google's translation service. It translates reviews from the `"IMDB-Dataset.csv"` file located in the `Datasets` directory and saves the translated text in a new file. The script uses multithreading to enhance performance and includes error handling for translation failures.

Prerequisites

- Python 3.x
- Pandas library
- `googletrans` version 3.1.0a0
- Other dependencies: `concurrent.futures`, `threading`, `logging`

Usage

1. Ensure the `"IMDB-Dataset.csv"` file is located in the `Datasets` directory.
2. Run the script:

   ```
   python src/translate_google.py
   ```

3. The script will translate the data and output two files in the `Datasets` directory:
   - `IMDB-Dataset-GoogleTranslate.csv`: Contains translated reviews and sentiments.
   - `failed-IMDB-Dataset-GoogleTranslate.csv`: Logs failed translation attempts.

Custom Dataset To use a different dataset:

- Place your CSV dataset in the `Datasets` directory.
- The dataset should have 'review' and 'sentiment' columns.
- Modify the script if your dataset columns have different names.
- Modify the script's `dataset` variable to match your dataset's filename.

Using `translate_mideind.py`

Overview `translate_mideind.py` is a Python script for translating text data using the `"mideind/nmt-doc-en-is-2022-10"` model. It translates reviews from the `"IMDB-Dataset.csv"` file in the `Datasets` directory and saves the translated text in a new file.

Prerequisites

- Python 3.x
- PyTorch
- `transformers` library
- `Pandas` library
- Other dependencies: `re`, `logging`

Note

- If you plan to use GPU acceleration with PyTorch, make sure your CUDA version is compatible with the installed PyTorch version.

Usage

1. Run the script:

   ```
   python src/translate_mideind.py
   ```

2. The script will process the data and output two files in the `Datasets` directory:
   - `IMDB-Dataset-MideindTranslate.csv`: Contains translated reviews and sentiments.
   - `failed-IMDB-Dataset-MideindTranslate.csv`: Logs failed translation attempts.

Custom Dataset To use a different dataset:

- Place your CSV dataset in the `Datasets` directory.
- The dataset should have 'review' and 'sentiment' columns.
- Modify the script if your dataset columns have different names.
- Modify the script's `dataset` variable to match your dataset's filename.

Process

Processing Icelandic Text

This section provides instructions for using the `process.py` script, which performs text normalization and preprocessing for Icelandic text using IceNLP.

Prerequisites

- Python 3.x
- Pandas library

- IceNLP tool (https://github.com/hrafnl/icenlp)
- Other dependencies: `multiprocessing`, `os`, `string`, `sys`, `time`, `tkinter`, `re`, `joblib`, `nefnir`

Installation

1. Download IceNLP from IceNLP GitHub Repository and extract it.

Usage

1. Run the script: `python src/process.py`
2. When prompted, select the `icetagger.bat` file located in the extracted IceNLP directory (`IceNLP-1.5.0\IceNLP\bat\icetagger`).
3. Ensure the dataset file (`IMDB-Dataset-MideindTranslate.csv`) is located in the `Datasets` directory relative to the script.
4. The script will process the dataset and output the processed data to `Datasets/IMDB-Dataset-MideindTranslate-processed-nefnir.csv`.

Custom Dataset To use a different dataset:

- Place your CSV dataset in the `Datasets` directory.
- The dataset should have 'review' and 'sentiment' columns.
- Modify the `dataset_path` variable in the script to match your dataset's filename.

Processing English Text

This section provides instructions for using the `process_eng.py` script, which performs text normalization and preprocessing for English text.

Prerequisites

- Python 3.x
- Pandas library
- NLTK library
- Other dependencies: `os`, `time`, `re`, `joblib`

Installation

1. Download necessary NLTK data: `python -m nltk.downloader punkt stopwords wordnet`

Usage

1. Ensure the dataset file (`IMDB-Dataset.csv`) is located in the `Datasets` directory.

2. Run the script: `python src/process_eng.py`
3. The script will process the dataset and output the processed data to `Datasets/IMDB-Dataset-Processed.csv`.

Custom Dataset To use a different dataset:

- Place your dataset in the `Datasets` directory.
- The dataset should be in CSV format with a 'review' column.
- Modify the `dataset_path` variable in the script to match your dataset's filename.

Baseline Classifiers

This section provides instructions for using the `BaselineClassifiersBinary.ipynb` script, which trains SVC, Logistic Regression and Naive Bayes on English, Icelandic Google and Icelandic Miðeind datasets, it also generates classification reports for each model.

Prerequisites

- Python 3.x
- PyTorch
- Pandas library
- Scikit-learn library
- Other dependencies: `os`, `time`, `numpy`

Usage

Go into `BaselineClassifiersBinary.ipynb` and run the cells. You have to change the `ICELANDIC_GOOGLE_CSV`, `ICELANDIC_MIDEIND_CSV` and `ENGLISH_CSV` variables to point to the correct datasets. The cell will train and print out the classification reports for each model. It will also show a diagram. You can refer to the next cell if you want to print out the most important features, although this is not necessary.

Transformer Models

This section provides instructions for using the `train.py` script, which trains a transformer model for sentiment analysis.

Prerequisites

- Python 3.x
- Transformers library
- PyTorch
- Pandas library
- Scikit-learn library

- Other dependencies: `os`, `time`, `numpy`

Note

- If you plan to use GPU acceleration with PyTorch, make sure your CUDA version is compatible with the installed PyTorch version.

Usage

1. Place the dataset file (default: `"IMDB-Dataset-GoogleTranslate.csv"`) in the `Datasets` directory relative to the script.
2. Modify the script if you want to use a different pre-trained model or dataset.
3. Run the script: `python src/train.py`
4. The script will train the model using the specified dataset and save the trained model and tokenizer in the `Models` directory.

Custom Dataset

To use a different dataset:

- Place your dataset in the `Datasets` directory.
- The dataset should be in CSV format with 'review' and 'sentiment' columns.
- Modify the `dataset_path` variable in the script to match your dataset's filename.

Generating Classification Reports

This section provides instructions for using the `generate_report.ipynb` script, which generates a classification report for a trained model. This is useful mostly for the transformer models, as the baseline classifiers generate their own reports via the same libraries.

This function will call the model and generate a classification report for the model. What it expects is the path to a folder of the model, the device to use, the pandas columns to use as X and y, and whether to return the accuracy or the classification report.

Usage

1. Import generate_classification_report.py `import generate_classification_report as gcr`
2. Load the CSV file with the data to be tested `df = pd.read_csv('IMDB-Dataset-GoogleTranslate.csv'`
3. Invoke the function call call_model, which takes the parameters

- X_all: All review columns
- y_all: All sentiment columns
- model: The model to be used (This is a path to a file, something like `'./electra-base-google-batch8-remove-noise-model/'`)

- device: The device to be used (CUDA, cpu)
- accuracy: Whether to return accuracy or return a classification report

Example

Example of how to generate a report can be seen in `generate_report.ipynb` - also the `generate_classification_report.py` `eval_files()` function, which is loading multiple models.

License

MIT

Authors

Ólafur Aron Jóhannsson
Eysteinn Örn
Birkir Arndal