ThilinaRajapakse / **pytorch-transformers-classification**

Based on the Pytorch-Transformers library by HuggingFace. To be used as a starting point for employing Transformer models in text classification tasks. Contains code to easily train BERT, XLNet, RoBERTa, and XLM models for text classification.

#pytorch-transformers #text-classification #natural-language-processing #transformer-models #huggingface

49 commits | **1** branch | **0** packages | **0** releases | **2** contributors | Apache-2.0

Branch: **master** | New pull request | Create new file | Upload files | Find file | Clone or download

ThilinaRajapakse Update data_download.sh — Latest commit `f2e87fc` 28 days ago

.gitignore	Initial commit	9 months ago
LICENSE	Initial commit	9 months ago

README.md	Update README.md	4 months ago
colab_quickstart.ipynb	columns parameter specified when writing tsv file for compatibility w...	9 months ago
data_download.sh	Update data_download.sh	28 days ago
data_prep.ipynb	columns parameter specified when writing tsv file for compatibility w...	9 months ago
run_model.ipynb	Update run_model.ipynb	6 months ago
utils.py	fix return statement	8 months ago

README.md

This repository is now deprecated. Please use Simple Transformers instead.

Update Notice

The underlying Pytorch-Transformers library by HuggingFace has been updated substantially since this repo was created. As such, this repo might not be compatible with the current version of the Hugging Face Transformers library. This repo will not be updated further.

I recommend using Simple Transformers (based on the updated Hugging Face library) as it is regularly maintained, feature rich, as well as (much) easier to use.

Pytorch-Transformers-Classification

This repository is based on the Pytorch-Transformers library by HuggingFace. It is intended as a starting point for anyone who wishes to use Transformer models in text classification tasks.

Please refer to this Medium article for further information on how this project works.

Check out the new library simpletransformers for one line training and evaluating!

Table of contents

- Setup
  - Simple Transformers
  - Quickstart using Colab
  - With Conda
- Usage
  - Yelp Demo
  - Custom Datasets
  - Current Pretrained Models
  - Evaluation Metrics
- Acknowledgements

Setup

Simple Transformers - Ready to use library

If you want to go directly to training, evaluating, and predicting with Transformer models, take a look at the Simple Transformers library. It's the easiest way to use Transformers for text classification with only 3 lines of code required. It's based on this repo but is designed to enable the use of Transformers without having to worry about the low level details. However, ease of usage comes at the cost of less control (and visibility) over how everything works.

Quickstart using Colab

Try this Google Colab Notebook for a quick preview. You can run all cells without any modifications to see how everything works. However, due to the 12 hour time limit on Colab instances, the dataset has been undersampled from 500 000 samples to about 5000 samples. For such a tiny sample size, everything should complete in about 10 minutes.

With Conda

1. Install Anaconda or Miniconda Package Manager from here
2. Create a new virtual environment and install packages.
   ```
   conda create -n transformers python pandas tqdm jupyter
   conda activate transformers
   ```
   If using cuda:
   ```
   conda install pytorch cudatoolkit=10.0 -c pytorch
   ```
   else:
   ```
   conda install pytorch cpuonly -c pytorch
   conda install -c anaconda scipy
   conda install -c anaconda scikit-learn
   pip install pytorch-transformers
   pip install tensorboardX
   ```
3. Clone repo. `git clone https://github.com/ThilinaRajapakse/pytorch-transformers-classification.git`

Usage

Yelp Demo

This demonstration uses the Yelp Reviews dataset.

Linux users can execute data_download.sh to download and set up the data files.

If you are doing it manually;

1. Download Yelp Reviews Dataset.
2. Extract `train.csv` and `test.csv` and place them in the directory `data/`.

Once the download is complete, you can run the data_prep.ipynb notebook to get the data ready for training.

Finally, you can run the run_model.ipynb notebook to fine-tune a Transformer model on the Yelp Dataset and evaluate the results.

Current Pretrained Models

The table below shows the currently available model types and their models. You can use any of these by setting the `model_type` and `model_name` in the `args` dictionary. For more information about pretrained models, see HuggingFace docs.

Architecture	Model Type	Model Name	Details
BERT	bert	bert-base-uncased	12-layer, 768-hidden, 12-heads, 110M parameters. Trained on lower-cased English text.
BERT	bert	bert-large-uncased	24-layer, 1024-hidden, 16-heads, 340M parameters. Trained on lower-cased English text.
BERT	bert	bert-base-cased	12-layer, 768-hidden, 12-heads, 110M parameters. Trained on cased English text.
BERT	bert	bert-large-cased	24-layer, 1024-hidden, 16-heads, 340M parameters. Trained on cased English text.
BERT	bert	bert-base-multilingual-uncased	(Original, not recommended) 12-layer, 768-hidden, 12-heads, 110M parameters. Trained on lower-cased text in the top 102 languages with the largest Wikipedias

Architecture	Model Type	Model Name	Details
BERT	bert	bert-base-multilingual-cased	(New, recommended) 12-layer, 768-hidden, 12-heads, 110M parameters. Trained on cased text in the top 104 languages with the largest Wikipedias
BERT	bert	bert-base-chinese	12-layer, 768-hidden, 12-heads, 110M parameters. Trained on cased Chinese Simplified and Traditional text.
BERT	bert	bert-base-german-cased	12-layer, 768-hidden, 12-heads, 110M parameters. Trained on cased German text by Deepset.ai
BERT	bert	bert-large-uncased-whole-word-masking	24-layer, 1024-hidden, 16-heads, 340M parameters. Trained on lower-cased English text using Whole-Word-Masking
BERT	bert	bert-large-cased-whole-word-masking	24-layer, 1024-hidden, 16-heads, 340M parameters. Trained on cased English text using Whole-Word-Masking
BERT	bert	bert-large-uncased-whole-word-masking-finetuned-squad	24-layer, 1024-hidden, 16-heads, 340M parameters. The bert-large-uncased-whole-word-masking model fine-tuned on SQuAD
BERT	bert	bert-large-cased-whole-word-masking-finetuned-squad	24-layer, 1024-hidden, 16-heads, 340M parameters The bert-large-cased-whole-word-masking model fine-tuned on SQuAD
BERT	bert	bert-base-cased-finetuned-mrpc	12-layer, 768-hidden, 12-heads, 110M parameters. The bert-base-cased model fine-tuned on MRPC

Architecture	Model Type	Model Name	Details
XLNet	xlnet	xlnet-base-cased	12-layer, 768-hidden, 12-heads, 110M parameters. XLNet English model
XLNet	xlnet	xlnet-large-cased	24-layer, 1024-hidden, 16-heads, 340M parameters. XLNet Large English model
XLM	xlm	xlm-mlm-en-2048	12-layer, 2048-hidden, 16-heads XLM English model
XLM	xlm	xlm-mlm-ende-1024	6-layer, 1024-hidden, 8-heads XLM English-German Multi-language model
XLM	xlm	xlm-mlm-enfr-1024	6-layer, 1024-hidden, 8-heads XLM English-French Multi-language model
XLM	xlm	xlm-mlm-enro-1024	6-layer, 1024-hidden, 8-heads XLM English-Romanian Multi-language model
XLM	xlm	xlm-mlm-xnli15-1024	12-layer, 1024-hidden, 8-heads XLM Model pre-trained with MLM on the 15 XNLI languages
XLM	xlm	xlm-mlm-tlm-xnli15-1024	12-layer, 1024-hidden, 8-heads XLM Model pre-trained with MLM + TLM on the 15 XNLI languages
XLM	xlm	xlm-clm-enfr-1024	12-layer, 1024-hidden, 8-heads XLM English model trained with CLM (Causal Language Modeling)

Architecture	Model Type	Model Name	Details
XLM	xlm	xlm-clm-ende-1024	6-layer, 1024-hidden, 8-heads XLM English-German Multi-language model trained with CLM (Causal Language Modeling)
RoBERTa	roberta	roberta-base	125M parameters RoBERTa using the BERT-base architecture
RoBERTa	roberta	roberta-large	24-layer, 1024-hidden, 16-heads, 355M parameters RoBERTa using the BERT-large architecture
RoBERTa	roberta	roberta-large-mnli	24-layer, 1024-hidden, 16-heads, 355M parameters roberta-large fine-tuned on MNLI.

Custom Datasets

When working with your own datasets, you can create a script/notebook similar to data_prep.ipynb that will convert the dataset to a Pytorch-Transformer ready format.

The data needs to be in `tsv` format, with four columns, and no header.

This is the required structure.

- `guid` : An ID for the row.
- `label` : The label for the row (should be an int).
- `alpha` : A column of the same letter for all rows. Not used in classification but still expected by the `DataProcessor` .
- `text` : The sentence or sequence of text.

Evaluation Metrics

The evaluation process in the run_model.ipynb notebook outputs the confusion matrix, and the Matthews correlation coefficient. If you wish to add any more evaluation metrics, simply edit the `get_eval_reports()` function in the notebook. This function takes the predictions and the ground truth labels as parameters, therefore you can add any custom metrics calculations to the function as required.

Acknowledgements

None of this would have been possible without the hard work by the HuggingFace team in developing the Pytorch-Transformers library.