ML-GroupProject2

Authors: Lia Bifano, Filippa Bång, Roman Bachmann

Files and Folders Overview

- **data.py**: Helper file containing function to create a Kaggle submission csv
- **helpers.py**: Helper file containing useful functions and classes
- **./analysis/**: Folder containing jupyter notebooks for the data analysis part
- **./ensemble/**: Folder containing the models and the ensembling script for the final submission
    - **ensemble_models/**: Folder containing pretrained pytorch neural networks for each ensemble model
    - **model*.py**: Training scripts to create, train and save all models
    - **ensemble.py**: Script combining all models and generating the **final submission**
- **./models/**: Folder where pretrained word2vec word vectors should be placed
- **./sentiment/**: Folder containing all models created in the iterative improvement process
    - **sentiment_models/**: Folder containing pretrained pytorch neural networks for each sentiment model
    - **w2v_trainer.py**: Script to generate the word2vec word vectors
    - **sentiment*.py**: Training scripts for the models that show our progress
- **./twitter-datasets/**: Folder where twitter training and testing data should be placed

Requirements

All code was only tested on macOS High Sierra (using only the CPU) and Ubuntu 14.04 (using a NVIDIA GTX 970).

Please run all files using the following dependencies:

- Python 3.5.4
- numpy (1.13.3)
- scipy (0.19.1)
- scikit-learn (0.19.0)
- gensim (3.1.0)
- Cython (0.27.3)

- PyTorch (0.3.0.post4)

If you have an NVIDIA GPU, it is recommended to install the following libraries to severely speed up the training:

- PyTorch with CUDA enabled
- PyTorch with CUDNN enabled

How to compute the final submission

1. Download the **train_neg_full.txt**, **train_pos_full.txt** and **test_data.txt** files from the *epfml17-text* Kaggle competition and place them in the **./twitter-datasets/** folder.
2. From Google Drive (186MB), download the word2vec word vectors and place the unziped files **twitter_word_vectors.bin** and **twitter_word_vectors.bin.syn0.npy** inside the **./models/** folder. To check the *last modified date* on Google Drive, log in with your google account and at the top right open the "Details" menu in the "More actions" menu.
3. Install all the above mentioned required libraries.
4. `cd ensemble`
5. `python ensemble.py`

Description of ./ensemble/ensemble.py

ensemble.py loads the twitter datasets, the word2vec word vectors and computes the tf-idf scores. It then loads all trained models and computes predictions on the train and test data. It uses the predictions of the train data to train a random forest. In the end, it outputs the ensembling of all models as a Kaggle csv submission file.

Since ensemble.py does not do any training of neural networks itself and merely loads the pretrained models, it does not need a GPU to run efficiently.

Approximate runtime on CPU: 20 minutes

Description of ./ensemble/model*.py

All model*.py files train a model used in the ensemble model. We trained model 2 ten times with different seeds.

- **model1.py**: 2 convolutional layers, kernel size 3, 256 convolutional channels, only with word2vec (seed=1)
- **model2.py**: 2 convolutional layers, kernel size 5, 256 convolutional channels, only with word2vec (seed=20+i for model2.i)
- **model3.py**: 2 convolutional layers, kernel size 7, 256 convolutional channels, only with word2vec

  (seed=3)
- **model4.py**: 2 convolutional layers, kernel size 3, 256 convolutional channels, with word2vec and tf-idf (seed=4)
- **model5.py**: 2 convolutional layers, kernel size 5, 128 convolutional channels, only with word2vec (seed=5)

Attention: CUDNN uses non-deterministic kernels. To make this code deterministic, uncomment *line 29* to disable CUDNN. We used CUDNN to train all our models, because it would take far too long to train them without!

These files can both be run on a CPU and a GPU.

- Approximate train time on CPU: Too high, please use a GPU
- Approximate train time on NVIDIA GTX 970 using CUDA and CUDNN: 1.6h for 20 epochs

Description of ./sentiment/sentiment*.py

The sentiment*.py files train the models that show our iterative progress.

- **sentiment1.py**: 1 convolutional layer, kernel size 3, 265 convolutional channels (seed=1)
- **sentiment2.py**: 2 convolutional layers, kernel size 3, 265 convolutional channels (seed=2)
- **sentiment3.py**: 2 convolutional layers, kernel size 3, 265 convolutional channels, using Dropout (seed=3)
- **sentiment4.py**: 2 convolutional layers, kernel size 3, 265 convolutional channels, using Dropout and Batch-normalization (seed=4)
- **sentiment5.py**: 2 convolutional layers, kernel size 5, 265 convolutional channels, using Dropout and Batch-normalization (seed=5)
- **sentiment6.py**: 2 convolutional layers, kernel size 7, 265 convolutional channels, using Dropout and Batch-normalization (seed=6)

Attention: CUDNN uses non-deterministic kernels. To make this code deterministic, uncomment *line 29* to disable CUDNN. We used CUDNN to train all our models, because it would take far too long to train them without!

These files can both be run on a CPU and a GPU.

- Approximate train time on CPU: Too high, please use a GPU
- Approximate train time on NVIDIA GTX 970 using CUDA and CUDNN: Depending on the complexity of the model, 1-3h for 20 epochs

Description of ./sentiment/w2v_trainer.py

w2v_trainer.py creates the word2vec word vectors from all the twitter datasets. It uses Cython to speed things up using multiple CPU threads. You can specify the number of workers in the *Word2Vec* training function on *line 32*. **Attention:** Using multiple workers yields non-deterministic results, since threads may not all run the same way. Also different machine architectures might observe different results because of different hash function behaviour. On a given machine, the word2vec training can be made deterministic by using only one worker, but this is very slow!

That's why for the ensemble.py file, please use the word2vec word vectors we provided on the Drive.

Approximate runtime on CPU: 10 minutes