Text Classification Competition - Twitter Sarcasm Detection

CS410 Fall 2020

Neeraj Aggarwal (noa2@illinois.edu), Samarth Keshari (keshari2@illinois.edu), Rishi Wadhwa (rishiw2@illinois.edu)

Background

The goal of this project is to build a model that can perform **Sarcasm Detection** on twitter data. The trained model should have a **F1-score above 0.723** on test data.

Solution Approaches

Machine Learning

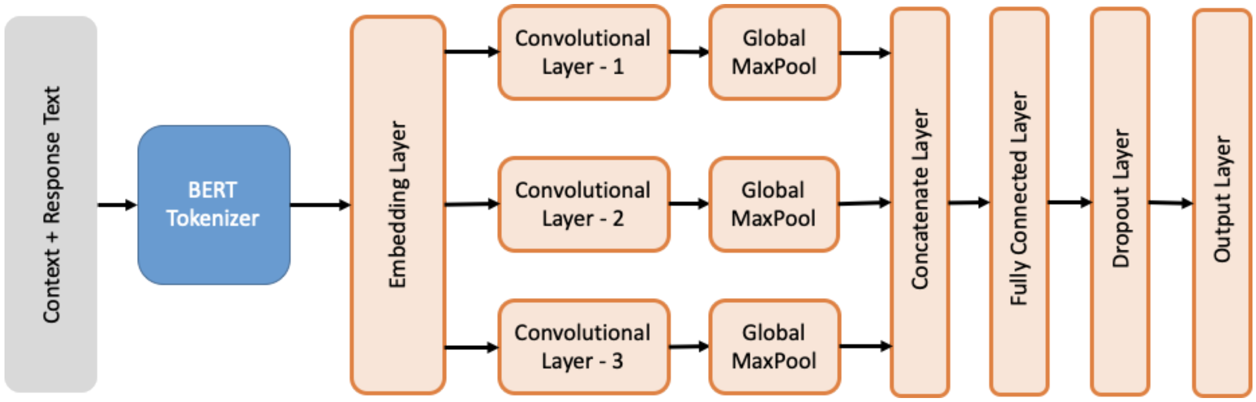
Standard machine learning requires us to manually select a set of relevant features and then train a machine learning model based on the features. As part of this project we trained a **Random Forest Classifier** using **Bag-of-Words of bigrams** as the feature representation. Training on many different hyperparameters, we eventually settled on 1000 trees, and a 46% confidence threshold to define as sarcasm.

Deep Learning

In contrast to the standard machine learning approach in deep learning we skip the manual step of feature extraction and directly feed the data to the deep learning algorithm which automatically learns features. The trained deep learning model is then used to perform the prediction task. In this project we used **Convolutional Neural Network(CNN)** based architecture along with a **pre-trained BERT Tokenizer** to generate the token ids. Following are the different elements of the software built as part of this approach

Model Architecture

The figure below shows the architecture of the deep neural network model implemented. The input text to the neural network is first tokenized using a pre-trained BERT tokenizer. The tokenized text is then fed to an embedding layer, followed by three parallel convolutional layers. Finally, the outputs from the convolutional layer become input to a fully connected layer followed by the softmax output layer.



Implementation Comparison

Below table compares the F1-scores of the tuned models based on both machine learning and deep learning approaches

Approach	Algorithm	Precision	Recall	F1
Deep Learning	Convolutional Neural Networks	0.6227867590	0.89888888	0.7357889949977261
Machine Learning	Random Forest Classifier	0.6824825174825175	0.7265952491849093	0.7265952491849093

Both the tuned models are able to achieve the goal to get F1 scores above the baseline of 0.723.

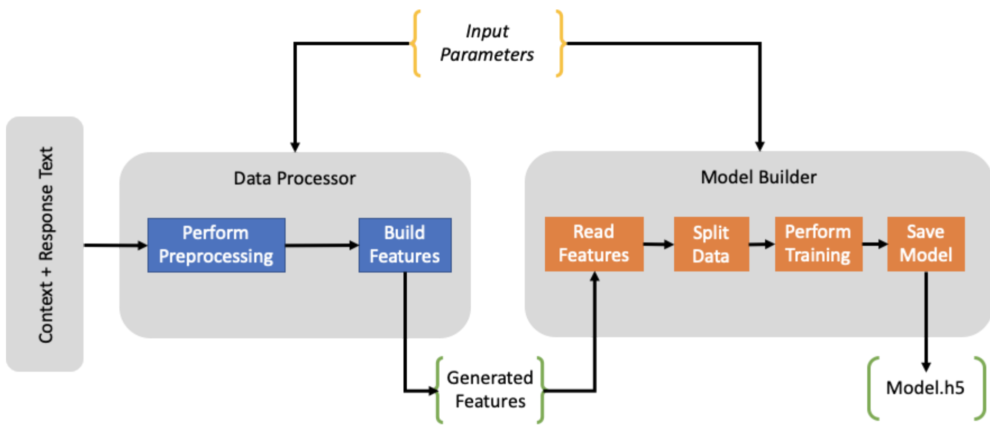
Software Implementation

Machine Learning - Random Forest

Deep Learning - CNN

Model Training Flow

The figure below shows the flow of the training process that is followed to build the deep learning model from data - Response + Context.

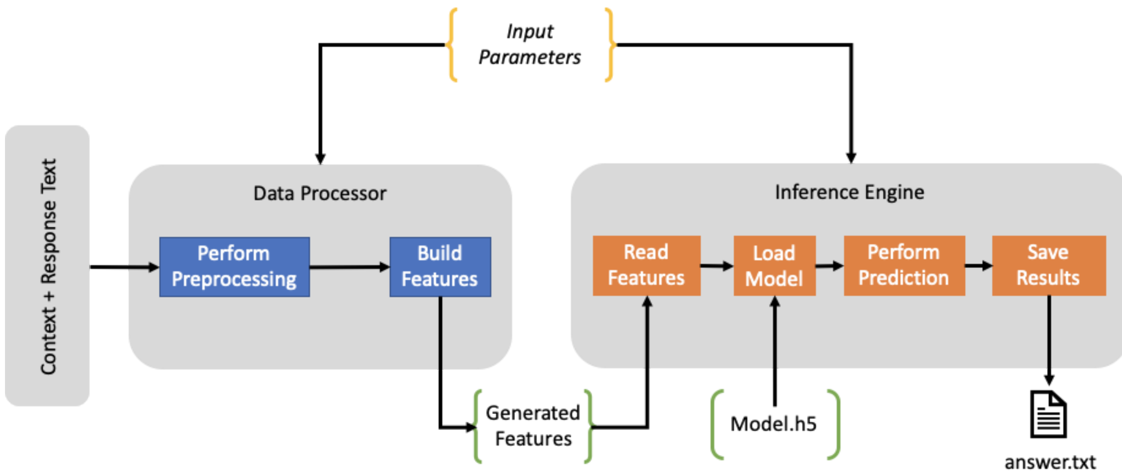


Model Training Flow

The training process consists of two components - 1) Data Processor 2) Model Builder. The Data Processor takes the context and response text from training data as input and generates required features as output. The Model Builder then takes the generated features as input and performs a sequence of steps to train and save the trained model. Both the components can be controlled based on the input parameters.

Model Inference Flow

The figure below shows the flow of the inference engine that is used to predict the results of test data.



Model Inference Flow

The inference process makes use of the same Data Processor from training to generate the features from input test data - context + response text. The generated features are then fed to the inference engine which loads the trained model and performs the prediction and saves the results.

Software Usage

Machine Learning

There are 4 machine learning models that are available for usage:

- Random Forest Classifier `random_forest.py`
- MLP Classifier `mlp_classifier.py`
- SGD Classifier `sgd_classifier.py`
- Logistic Regression `logistic_regression.py`

To run the machine learning models, `python [file.py]`. It will generate an `answer.txt` in `./src/machinelearning`.

Deep learning - CNN

Installation Requirements

The entire software is built using python(**version 3.7.7**) programming language and following are the packages required to run the Model Train and Inference programs. These packages can be installed using command “pip install”

- pip install regex
- pip install emoji

- pip install demoji
- pip install pandas
- pip install nltk
- pip install --upgrade tensorflow
- pip install Keras
- pip install bert-for-tf2
- pip install sentencepiece

After the installation of required packages is complete, clone or download the code from https://github.com/n3a9/CourseProject.git.

APIs

As part of this project, the user can call two apis

- **Model Training**
  - In order to call this api, in the command terminal the user need to
    - Navigate into the cloned repository to the directory ***../src/deeplearning***
    - If required, change the parameters file '***params_config.json***' at ***../src/deeplearning/parameters***. Refer to the *Parameters section* below for details about the different parameters used during the model training
    - Run command - ***python modelTrain.py***
    - The trained model weights will be saved at ***../src/deeplearning/trained-models*** in '***cnn_model_weight.h5***' file

Note: For the project verification purpose, model training can be performed by changing different parameters(refer to Parameters section below). Currently by default any new trained model weights will not be saved, however, caution should be taken that any new trained model weights if saved can vary the final results.

- **Model Inference**
  - In order to call this api, in the command terminal the user need to
    - Navigate into the cloned repository to the directory ***../src/deeplearning***
    - If required, change the parameters file '***params_config.json***' at ***../src/deeplearning/parameters***. Refer to the *Parameters section* below for details about the different parameters used during the model inference.
    - Run command - ***python modelInference.py***
    - The final predictions will be saved at ***../src*** in '***answer.txt***' file

Note: For project verification purpose run only the Model Inference

Parameters

Below is the list of parameters that are used during the model training and inference process. Refer to '***params_config.json***' file in the cloned repository at ***twitter-sarcasm-detection/src/deeplearning/parameters***

Name	Description	Used In
n_last_context	Number of last entries from in the context list	Training + Inference
data-path	Path to folder storing the train and test data files	Training + Inference
train-data-filename	Name of the train file in .jsonl format	Training
test-data-filename	Name of the test file in .jsonl format	Inference
processed-data-path	Path to folder storing the processed train and test data files	Training + Inference
processed-train-data-filename	Name of the processed train file in .csv format	Training
processed-test-data-filename	Name of the processed test file in .csv format	Inference
features-path	Path to folder storing the train and test features files	Training + Inference
features-train-filename	Name of the train features file in .json format	Training
features-test-filename	Name of the test features file in .json format	Inference
trained-model-save	Flag to indicate that the weights of the trained model should be saved. By default the model will not be saved. If required, set the flag to "X".	Training
trained-model-path	Path to the folder storing the trained model weights	Training
trained-model-weight-filename	Name of the file storing the trained model weights in .h5 format	Training
train_test_split	% of records that are needed for validation during model training. The value of this parameter should be between (0,1)	Training
embedding_dimensions	Number of dimensions in the embedding layer of the model	Training

cnn_filters	Number of CNN filters in the CNN layers of the model	Training
dnn_units	Number of neurons in the fully connected layer of the model	Training
dropout_rate	Dropout rate for the fully connected layer of the model	Training
verbose		Training
n_epochs	Number of epochs for model training	Training
batch_size	Batch size for model training	Training
prediction-threshold	Model predictions for test data are in terms of probabilities. For a particular test sample, if the prediction probability is above this threshold value, then the test sample is flagged as SARCASM otherwise NON-SARCASM	Inference
answer-file	Path + filename of the final results file in .txt format	Inference