


Text Classification with QRNN

Python notebook using data from Toxic Comment Classification Challenge · 574 views ·
deep learning, text data, text mining

Version 4
4 commits

forked from Text Classification with Atrous Convolutions

Submission
✔ Ran successfully
Submitted by Kevin Mader 9 months ago

Private Score	Public Score
0.50000	0.50000

Overview

Here we show how well a QRNN works when compared with standard CNN and LSTM models. The parent is a 1D Convolution and other models show the stacked LSTM performance

In [1]:
```
import numpy as np # linear algebra
import pandas as pd # data processing, CSV file I/O (e.g. pd.re
d_csv)
from keras.models import Model
from keras.layers import Dense, Embedding, Input
from keras.layers import Conv1D, GlobalMaxPool1D, Dropout, con
atenate
from keras.preprocessing import text, sequence
from keras.callbacks import EarlyStopping, ModelCheckpoint
```

```
/opt/conda/lib/python3.6/site-packages/h5py/__init__.py:36: Fu
tureWarning: Conversion of the second argument of issubdtype f
rom `float` to `np.floating` is deprecated. In future, it will
be treated as `np.float64 == np.dtype(float).type`.
  from ._conv import register_converters as _register_converte
rs
Using TensorFlow backend.
```

QRNN Implementation

We use a reference implementation of the QRNN layer from https://github.com/DingKe/nn_playground/blob/master/qrnn/qrnn.py (https://github.com/DingKe/nn_playground/blob/master/qrnn/qrnn.py)

In [2]:
```
from keras import backend as K
from keras import activations, initializers, regularizers, con
straints
from keras.layers import Layer, InputSpec
from keras.utils.conv_utils import conv_output_length

def _dropout(x, level, noise_shape=None, seed=None):
    x = K.dropout(x, level, noise_shape, seed)
    x *= (1. - level) # compensate for the scaling by the dropo
t
    return x


class QRNN(Layer):
    '''Quasi RNN
    # Arguments
        units: dimension of the internal projections and the fir
```

```
    al output.
    # References
        - [Quasi-recurrent Neural Networks](http://arxiv.org/ab
s/1611.01576)
    '''
    def __init__(self, units, window_size=2, stride=1,
                 return_sequences=False, go_backwards=False,
                 stateful=False, unroll=False, activation='tan
h',
                 kernel_initializer='uniform', bias_initialize
='zero',
                 kernel_regularizer=None, bias_regularizer=Non
,
                 activity_regularizer=None,
                 kernel_constraint=None, bias_constraint=None,
                 dropout=0, use_bias=True, input_dim=None, inpu
t_length=None,
                 **kwargs):
        self.return_sequences = return_sequences
        self.go_backwards = go_backwards
        self.stateful = stateful
        self.unroll = unroll

        self.units = units
        self.window_size = window_size
        self.strides = (stride, 1)

        self.use_bias = use_bias
        self.activation = activations.get(activation)
        self.kernel_initializer = initializers.get(kernel_init
alizer)
        self.bias_initializer = initializers.get(bias_initiali
er)
        self.kernel_regularizer = regularizers.get(kernel_regu
arizer)
        self.bias_regularizer = regularizers.get(bias_regulari
er)
        self.activity_regularizer = regularizers.get(activity_
egularizer)
        self.kernel_constraint = constraints.get(kernel_constr
int)
        self.bias_constraint = constraints.get(bias_constraint

        self.dropout = dropout
        self.supports_masking = True
        self.input_spec = [InputSpec(ndim=3)]
        self.input_dim = input_dim
        self.input_length = input_length
        if self.input_dim:
            kwargs['input_shape'] = (self.input_length, self.i
put_dim)
        super(QRNN, self).__init__(**kwargs)

    def build(self, input_shape):
```

```
        if isinstance(input_shape, list):
            input_shape = input_shape[0]

        batch_size = input_shape[0] if self.stateful else None
        self.input_dim = input_shape[2]
        self.input_spec = InputSpec(shape=(batch_size, None, s
lf.input_dim))
        self.state_spec = InputSpec(shape=(batch_size, self.un
ts))

        self.states = [None]
        if self.stateful:
            self.reset_states()

        kernel_shape = (self.window_size, 1, self.input_dim, s
lf.units * 3)
        self.kernel = self.add_weight(name='kernel',
                                      shape=kernel_shape,
                                      initializer=self.kernel_
nitializer,
                                      regularizer=self.kernel_
egularizer,
                                      constraint=self.kernel_c
nstraint)
        if self.use_bias:
            self.bias = self.add_weight(name='bias',
                                        shape=(self.units * 3
,),
                                        initializer=self.bias_
nitializer,
                                        regularizer=self.bias_
egularizer,
                                        constraint=self.bias_c
nstraint)

        self.built = True

    def compute_output_shape(self, input_shape):
        if isinstance(input_shape, list):
            input_shape = input_shape[0]

        length = input_shape[1]
        if length:
            length = conv_output_length(length + self.window_s
ze - 1,
                                        self.window_size, 'val
d',
                                        self.strides[0])
        if self.return_sequences:
            return (input_shape[0], length, self.units)
        else:
            return (input_shape[0], self.units)

    def compute_mask(self, inputs, mask):
```

```
        if self.return_sequences:
            return mask
        else:
            return None

    def get_initial_states(self, inputs):
        # build an all-zero tensor of shape (samples, units)
        initial_state = K.zeros_like(inputs)  # (samples, times
teps, input_dim)
        initial_state = K.sum(initial_state, axis=(1, 2))  # (s
amples,)
        initial_state = K.expand_dims(initial_state)  # (sample
s, 1)
        initial_state = K.tile(initial_state, [1, self.units])
# (samples, units)
        initial_states = [initial_state for _ in range(len(sel
.states))]
        return initial_states

    def reset_states(self, states=None):
        if not self.stateful:
            raise AttributeError('Layer must be stateful.')
        if not self.input_spec:
            raise RuntimeError('Layer has never been called '
                               'and thus has no states.')

        batch_size = self.input_spec.shape[0]
        if not batch_size:
            raise ValueError('If a QRNN is stateful, it needs
o know '
                             'its batch size. Specify the batc
size '
                             'of your input tensors: \n'
                             '- If using a Sequential model, '
                             'specify the batch size by passin
'
                             'a `batch_input_shape` '
                             'argument to your first layer.\n'
                             '- If using the functional API, s
ecify '
                             'the time dimension by passing a
                             '`batch_shape` argument to your I
put layer.')

        if self.states[0] is None:
            self.states = [K.zeros((batch_size, self.units))
                           for _ in self.states]
        elif states is None:
            for state in self.states:
                K.set_value(state, np.zeros((batch_size, self.
nits)))
        else:
            if not isinstance(states, (list, tuple)):
                states = [states]
```

```
            if len(states) != len(self.states):
                raise ValueError('Layer ' + self.name + ' expe
cts ' +
                                 str(len(self.states)) + ' sta
es, '
                                 'but it received ' + str(len(
tates)) +
                                 'state values. Input received
' +
                                 str(states))
            for index, (value, state) in enumerate(zip(states,
self.states)):
                if value.shape != (batch_size, self.units):
                    raise ValueError('State ' + str(index) +
                                     ' is incompatible with la
er ' +
                                     self.name + ': expected s
ape=' +
                                     str((batch_size, self.uni
s)) +
                                     ', found shape=' + str(va
ue.shape))
                K.set_value(state, value)

    def __call__(self, inputs, initial_state=None, **kwargs):
        # If `initial_state` is specified,
        # and if it a Keras tensor,
        # then add it to the inputs and temporarily
        # modify the input spec to include the state.
        if initial_state is not None:
            if hasattr(initial_state, '_keras_history'):
                # Compute the full input spec, including state
                input_spec = self.input_spec
                state_spec = self.state_spec
                if not isinstance(state_spec, list):
                    state_spec = [state_spec]
                self.input_spec = [input_spec] + state_spec

                # Compute the full inputs, including state
                if not isinstance(initial_state, (list, tuple
)):
                    initial_state = [initial_state]
                inputs = [inputs] + list(initial_state)

                # Perform the call
                output = super(QRNN, self).__call__(inputs, **
wargs)

                # Restore original input spec
                self.input_spec = input_spec
                return output
            else:
                kwargs['initial_state'] = initial_state
        return super(QRNN, self).__call__(inputs, **kwargs)
```

```
    def call(self, inputs, mask=None, initial_state=None, trai
ing=None):
        # input shape: `(samples, time (padded with zeros), inpu
t_dim)`
        # note that the .build() method of subclasses MUST defi
e
        # self.input_spec and self.state_spec with complete inpu
t shapes.
        if isinstance(inputs, list):
            initial_states = inputs[1:]
            inputs = inputs[0]
        elif initial_state is not None:
            pass
        elif self.stateful:
            initial_states = self.states
        else:
            initial_states = self.get_initial_states(inputs)

        if len(initial_states) != len(self.states):
            raise ValueError('Layer has ' + str(len(self.state
)) +
                             ' states but was passed ' +
                             str(len(initial_states)) +
                             ' initial states.')
        input_shape = K.int_shape(inputs)
        if self.unroll and input_shape[1] is None:
            raise ValueError('Cannot unroll a RNN if the '
                             'time dimension is undefined. \n'
                             '- If using a Sequential model, '
                             'specify the time dimension by pa
sing '
                             'an `input_shape` or `batch_input
shape` '
                             'argument to your first layer. If
 your '
                             'first layer is an Embedding, you
 can '
                             'also use the `input_length` argu
ent.\n'
                             '- If using the functional API, s
ecify '
                             'the time dimension by passing a
shape` '
                             'or `batch_shape` argument to you
Input layer.')
        constants = self.get_constants(inputs, training=None)
        preprocessed_input = self.preprocess_input(inputs, tra
ning=None)

        last_output, outputs, states = K.rnn(self.step, prepro
essed_input,
                                             initial_states,
                                             go_backwards=self.
```

```
o_backwards,
                                                mask=mask,
                                                constants=constant
,
                                                unroll=self.unroll
                                                input_length=input_
shape[1])
        if self.stateful:
            updates = []
            for i in range(len(states)):
                updates.append((self.states[i], states[i]))
            self.add_update(updates, inputs)

        # Properly set learning phase
        if 0 < self.dropout < 1:
            last_output._uses_learning_phase = True
            outputs._uses_learning_phase = True

        if self.return_sequences:
            return outputs
        else:
            return last_output

    def preprocess_input(self, inputs, training=None):
        if self.window_size > 1:
            inputs = K.temporal_padding(inputs, (self.window_si
ze-1, 0))
        inputs = K.expand_dims(inputs, 2)  # add a dummy dimens
ion

        output = K.conv2d(inputs, self.kernel, strides=self.st
ides,
                          padding='valid',
                          data_format='channels_last')
        output = K.squeeze(output, 2)  # remove the dummy dimen
sion
        if self.use_bias:
            output = K.bias_add(output, self.bias, data_format=
'channels_last')

        if self.dropout is not None and 0. < self.dropout < 1.
            z = output[:, :, :self.units]
            f = output[:, :, self.units:2 * self.units]
            o = output[:, :, 2 * self.units:]
            f = K.in_train_phase(1 - _dropout(1 - f, self.dropo
ut), f, training=training)
            return K.concatenate([z, f, o], -1)
        else:
            return output

    def step(self, inputs, states):
        prev_output = states[0]

        z = inputs[:, :self.units]
```

```
        f = inputs[:, self.units:2 * self.units]
        o = inputs[:, 2 * self.units:]

        z = self.activation(z)
        f = f if self.dropout is not None and 0. < self.dropou
< 1. else K.sigmoid(f)
        o = K.sigmoid(o)

        output = f * prev_output + (1 - f) * z
        output = o * output

        return output, [output]

    def get_constants(self, inputs, training=None):
        return []

    def get_config(self):
        config = {'units': self.units,
                  'window_size': self.window_size,
                  'stride': self.strides[0],
                  'return_sequences': self.return_sequences,
                  'go_backwards': self.go_backwards,
                  'stateful': self.stateful,
                  'unroll': self.unroll,
                  'use_bias': self.use_bias,
                  'dropout': self.dropout,
                  'activation': activations.serialize(self.act
vation),
                  'kernel_initializer': initializers.serialize
self.kernel_initializer),
                  'bias_initializer': initializers.serialize(s
lf.bias_initializer),
                  'kernel_regularizer': regularizers.serialize
self.kernel_regularizer),
                  'bias_regularizer': regularizers.serialize(s
lf.bias_regularizer),
                  'activity_regularizer': regularizers.seriali
e(self.activity_regularizer),
                  'kernel_constraint': constraints.serialize(s
lf.kernel_constraint),
                  'bias_constraint': constraints.serialize(sel
.bias_constraint),
                  'input_dim': self.input_dim,
                  'input_length': self.input_length}
        base_config = super(QRNN, self).get_config()
        return dict(list(base_config.items()) + list(config.it
ms()))
```

In [3]:

```
# define network parameters
max_features = 20000
maxlen = 100
```

Load and Preprocessing Steps

Here we load the data and fill in the misisng values

In [4]:
```
train = pd.read_csv("../input/train.csv")
test = pd.read_csv("../input/test.csv")
train = train.sample(frac=1)

list_sentences_train = train["comment_text"].fillna("Invalid")
values
list_classes = ["toxic", "severe_toxic", "obscene", "threat",
"insult", "identity_hate"]
y = train[list_classes].values
list_sentences_test = test["comment_text"].fillna("Invalid").v
lues
```

Sequence Generation

Here we take the data and generate sequences from the data

In [5]:
```
tokenizer = text.Tokenizer(num_words=max_features)
tokenizer.fit_on_texts(list(list_sentences_train))
# train data
list_tokenized_train = tokenizer.texts_to_sequences(list_sente
ces_train)
X_t = sequence.pad_sequences(list_tokenized_train, maxlen=maxl
n)
# test data
list_tokenized_test = tokenizer.texts_to_sequences(list_senten
es_test)
X_te = sequence.pad_sequences(list_tokenized_test, maxlen=maxl
n)
```

In [6]:
```
def build_model(conv_layers = 2, max_dilation_rate = 3):
    embed_size = 128
    inp = Input(shape=(maxlen, ))
    x = Embedding(max_features, embed_size)(inp)
    x = Dropout(0.25)(x)
    x = Conv1D(2*embed_size,
                   kernel_size = 3)(x)
    prefilt_x = Conv1D(2*embed_size,
                   kernel_size = 3)(x)
    out_conv = []
    x = prefilt_x
    # strides rate we use here for skip
    for strides in [1, 1, 2]:
        x = QRNN(128*2**(strides),
                 return_sequences = True
```

```
                return_sequences = True,
                stride = strides,
                dropout = 0.2)(x)
    x_f = QRNN(512)(x)
    x_b = QRNN(512, go_backwards=True)(x)
    x = concatenate([x_f, x_b])
    x = Dropout(0.5)(x)
    x = Dense(64, activation="relu")(x)
    x = Dropout(0.1)(x)
    x = Dense(6, activation="sigmoid")(x)
    model = Model(inputs=inp, outputs=x)
    model.compile(loss='binary_crossentropy',
                  optimizer='adam',
                  metrics=['binary_accuracy'])

    return model

model = build_model()
model.summary()
```

```
_________________________________________________________________
_____________________________________
Layer (type)                 Output Shape              Param #
Connected to
=================================================================
=====================================
input_1 (InputLayer)         (None, 100)               0
_________________________________________________________________
_____________________________________
embedding_1 (Embedding)      (None, 100, 128)          2560000
input_1[0][0]
_________________________________________________________________
_____________________________________
dropout_1 (Dropout)          (None, 100, 128)          0
embedding_1[0][0]
_________________________________________________________________
_____________________________________
conv1d_1 (Conv1D)            (None, 98, 256)           98560
dropout_1[0][0]
_________________________________________________________________
_____________________________________
conv1d_2 (Conv1D)            (None, 96, 256)           196864
conv1d_1[0][0]
_________________________________________________________________
_____________________________________
qrnn_1 (QRNN)                (None, 96, 256)           393984
conv1d_2[0][0]
_________________________________________________________________
_____________________________________
qrnn_2 (QRNN)                (None, 96, 256)           393984
qrnn_1[0][0]
_________________________________________________________________
_____________________________________
qrnn_3 (QRNN)                (None, 48, 512)           787968
```

```
qrnn_2[0][0]
__________________________________________________________________
___________________________________
qrnn_4 (QRNN)                   (None, 512)          1574400
qrnn_3[0][0]
__________________________________________________________________
___________________________________
qrnn_5 (QRNN)                   (None, 512)          1574400
qrnn_3[0][0]
__________________________________________________________________
___________________________________
concatenate_1 (Concatenate)     (None, 1024)         0
qrnn_4[0][0]

qrnn_5[0][0]
__________________________________________________________________
___________________________________
dropout_2 (Dropout)             (None, 1024)         0
concatenate_1[0][0]
__________________________________________________________________
___________________________________
dense_1 (Dense)                 (None, 64)           65600
dropout_2[0][0]
__________________________________________________________________
___________________________________
dropout_3 (Dropout)             (None, 64)           0
dense_1[0][0]
__________________________________________________________________
___________________________________
dense_2 (Dense)                 (None, 6)            390
dropout_3[0][0]
==================================================================
===================================
Total params: 7,646,150
Trainable params: 7,646,150
Non-trainable params: 0
__________________________________________________________________
___________________________________
```

Train the Model

Here we train the model and use model checkpointing and early stopping to keep only the best version of the model

In [7]:

```
batch_size = 1024 # big beefy GPUs like large batches
epochs = 12

file_path="weights.hdf5"
checkpoint = ModelCheckpoint(file_path, monitor='val_loss', ve
bose=1, save_best_only=True, mode='min')
```

```
early = EarlyStopping(monitor="val_loss", mode="min", patience=
5)

callbacks_list = [checkpoint, early] #early
model.fit(X_t, y,
          batch_size=batch_size,
          epochs=epochs,
          shuffle = True,
          validation_split=0.25,
          callbacks=callbacks_list)
```

```
Train on 119678 samples, validate on 39893 samples
Epoch 1/12
119678/119678 [==============================] - 244s 2ms/step
- loss: 0.6415 - binary_accuracy: 0.9596 - val_loss: 0.6409 -
val_binary_accuracy: 0.9632

Epoch 00001: val_loss improved from inf to 0.64090, saving mod
el to weights.hdf5
Epoch 2/12
119678/119678 [==============================] - 232s 2ms/step
- loss: 0.6166 - binary_accuracy: 0.9634 - val_loss: 0.5929 -
val_binary_accuracy: 0.9632

Epoch 00002: val_loss improved from 0.64090 to 0.59286, saving
model to weights.hdf5
Epoch 3/12
119678/119678 [==============================] - 232s 2ms/step
- loss: 0.5711 - binary_accuracy: 0.9634 - val_loss: 0.5498 -
val_binary_accuracy: 0.9632

Epoch 00003: val_loss improved from 0.59286 to 0.54985, saving
model to weights.hdf5
Epoch 4/12
119678/119678 [==============================] - 232s 2ms/step
- loss: 0.5303 - binary_accuracy: 0.9634 - val_loss: 0.5112 -
val_binary_accuracy: 0.9632

Epoch 00004: val_loss improved from 0.54985 to 0.51116, saving
model to weights.hdf5
Epoch 5/12
119678/119678 [==============================] - 232s 2ms/step
- loss: 0.4935 - binary_accuracy: 0.9634 - val_loss: 0.4763 -
val_binary_accuracy: 0.9632

Epoch 00005: val_loss improved from 0.51116 to 0.47631, saving
model to weights.hdf5
Epoch 6/12
119678/119678 [==============================] - 232s 2ms/step
- loss: 0.4604 - binary_accuracy: 0.9634 - val_loss: 0.4449 -
val_binary_accuracy: 0.9632

Epoch 00006: val_loss improved from 0.47631 to 0.44494, saving
model to weights.hdf5
```

```
Epoch 7/12
119678/119678 [==============================] - 232s 2ms/step
- loss: 0.4305 - binary_accuracy: 0.9634 - val_loss: 0.4167 -
val_binary_accuracy: 0.9632

Epoch 00007: val_loss improved from 0.44494 to 0.41669, saving
model to weights.hdf5
Epoch 8/12
119678/119678 [==============================] - 232s 2ms/step
- loss: 0.4037 - binary_accuracy: 0.9634 - val_loss: 0.3912 -
val_binary_accuracy: 0.9632

Epoch 00008: val_loss improved from 0.41669 to 0.39124, saving
model to weights.hdf5
Epoch 9/12
119678/119678 [==============================] - 232s 2ms/step
- loss: 0.3795 - binary_accuracy: 0.9634 - val_loss: 0.3683 -
val_binary_accuracy: 0.9632

Epoch 00009: val_loss improved from 0.39124 to 0.36830, saving
model to weights.hdf5
Epoch 10/12
119678/119678 [==============================] - 232s 2ms/step
- loss: 0.3577 - binary_accuracy: 0.9634 - val_loss: 0.3476 -
val_binary_accuracy: 0.9632

Epoch 00010: val_loss improved from 0.36830 to 0.34764, saving
model to weights.hdf5
Epoch 11/12
119678/119678 [==============================] - 232s 2ms/step
- loss: 0.3380 - binary_accuracy: 0.9634 - val_loss: 0.3290 -
val_binary_accuracy: 0.9632

Epoch 00011: val_loss improved from 0.34764 to 0.32900, saving
model to weights.hdf5
Epoch 12/12
119678/119678 [==============================] - 232s 2ms/step
- loss: 0.3203 - binary_accuracy: 0.9634 - val_loss: 0.3122 -
val_binary_accuracy: 0.9632

Epoch 00012: val_loss improved from 0.32900 to 0.31217, saving
model to weights.hdf5
```

Out[7]:

```
<keras.callbacks.History at 0x7f6574d93278>
```

Test on a few sentences

Show what the model actually predicts for a few test sentences

In [8]:

```
from IPython.display import Markdown, display
```

```
dmd = lambda x: display(Markdown(x))
def show_sentence(sent_idx):
    dmd('# Input Sentence:\n `{}`'.format(list_sentences_train
sent_idx]))
    c_pred = model.predict(X_t[sent_idx:sent_idx+1])[0]
    dmd('## Positive Categories')
    for k, v, p in zip(list_classes, y[sent_idx], c_pred):
        if v>0:
            dmd('- {}, Prediction: {:2.2f}%'.format(k, 100*v,
00*p))
    dmd('## Negative Categories')
    for k, v, p in zip(list_classes, y[sent_idx], c_pred):
        if v<1:
            dmd('- {}, Prediction: {:2.2f}%'.format(k, 100*p))
show_sentence(0)
show_sentence(50)
```

Input Sentence:

```
Sure, that couldn't hurt. I've actually tested it on Mac/Firefo
and Mac/Safari (albeit an earlier version of Firefox) and it's
worked for me. So I think it's less likely a straight-up
browser/OS issue, and more likely an interaction with either the
monobook.js or perhaps another piece of Javascript code that may
be running on your client. But that's just a guess. If it behave
differently for you with Safari, please let me know. '''''' Talk
```

Positive Categories

Negative Categories

- toxic, Prediction: 24.37%
- severe_toxic, Prediction: 23.33%
- obscene, Prediction: 23.79%
- threat, Prediction: 23.26%
- insult, Prediction: 23.75%
- identity_hate, Prediction: 23.32%

Input Sentence:

```
I don't believe it is an attack, but a reminder not to make
```

```
I don t believe it is an attack, but a reminder not to make
personal attacks.  If one were to misconstrue it, as you choose
to, as an attack, it would still only be one attack, not multipl
I'm sorry, but you'll need to try again.
```

Positive Categories

Negative Categories

- toxic, Prediction: 24.37%
- severe_toxic, Prediction: 23.33%
- obscene, Prediction: 23.79%
- threat, Prediction: 23.26%
- insult, Prediction: 23.75%
- identity_hate, Prediction: 23.32%

Make Predictions

Load the model and make predictions on the test dataset

In [9]:
```
model.load_weights(file_path)
y_test = model.predict(X_te, verbose = True, batch_size = 1024
sample_submission = pd.read_csv("../input/sample_submission.cs
v")
sample_submission[list_classes] = y_test
sample_submission.to_csv("predictions.csv",
                         index=False)
```

```
153164/153164 [==============================] - 92s 603us/ste
p
```


Data

Data Sources

Toxic Comment Clas...

Toxic Comment Classification Challenge

samp... 153k x 7
test.c... 153k x 2
test_l... 153k x 7
train.... 160k x 8

Identify and classify toxic online comments
Last Updated: a year ago

About this Competition

You are provided with a large number of Wikipedia commer which have been labeled by human raters for toxic behavior The types of toxicity are:

- `toxic`
- `severe_toxic`
- `obscene`
- `threat`
- `insult`
- `identity_hate`

You must create a model which predicts a probability of eac type of toxicity for each comment.

File descriptions

- **train.csv** - the training set, contains comments with their binary labels
- **test.csv** - the test set, you must predict the toxicity probabilities for these comments. To deter hand labeling, the test set contains some comments which are not included in scoring.
- **sample_submission.csv** - a sample submission file in the correct format
- **test_labels.csv** - labels for the test data; value of `-1` indicates it was not used for scoring; (**Note:** file added aft competition close!)

Output Files

New Kernel Download All

Output Files

predictions.csv
weights.hdf5

About this file

This file was created from a Kernel, it does not have a description.

predictions.csv

1	id	toxic	severe_toxic	obscene	threat	insult	identity_hate
2	00001cee341fdb12	0.2437363266944885	0.2333270013332367	0.2378795444965362	0.2325991988182068	0.2374608963727951	0.233263242187
3	0000247867823ef7	0.2437363266944885	0.2333270013332367	0.2378795444965362	0.2325991988182068	0.2374608963727951	0.233263242187

4	00013b17ad220c46	0.24373632669448853	0.2333270013332367	0.23787954449653625	0.2325991988182068	0.2374608963727951	0.2332 3242187
5	00017563c3f7919a	0.24373632669448853	0.2333270013332367	0.23787954449653625	0.2325991988182068	0.2374608963727951	0.2332 3242187
6	00017695ad8997eb	0.24373632669448853	0.2333270013332367	0.23787954449653625	0.2325991988182068	0.2374608963727951	0.2332 3242187
7	0001ea8717f6de06	0.24373632669448853	0.2333270013332367	0.23787954449653625	0.2325991988182068	0.2374608963727951	0.2332 3242187
8	00024115d4cbde0f	0.24373632669448853	0.2333270013332367	0.23787954449653625	0.2325991988182068	0.2374608963727951	0.2332 3242187
9	000247e83dcc1211	0.24373632669448853	0.2333270013332367	0.23787954449653625	0.2325991988182068	0.2374608963727951	0.2332 3242187
10	00025358d4737918	0.24373632669448853	0.2333270013332367	0.23787954449653625	0.2325991988182068	0.2374608963727951	0.2332 3242187
11	00026d1092fe71cc	0.24373632669448853	0.2333270013332367	0.23787954449653625	0.2325991988182068	0.2374608963727951	0.2332 3242187
12	0002eadc3b301559	0.24373632669448853	0.2333270013332367	0.23787954449653625	0.2325991988182068	0.2374608963727951	0.2332 3242187
13	0002f87b16116a7f	0.24373632669448853	0.2333270013332367	0.23787954449653625	0.2325991988182068	0.2374608963727951	0.2332 3242187
14	0003806b11932181	0.24373632669448853	0.2333270013332367	0.23787954449653625	0.2325991988182068	0.2374608963727951	0.2332 3242187
15	0003e1cccfd5a40a	0.24373632669448853	0.2333270013332367	0.23787954449653625	0.2325991988182068	0.2374608963727951	0.2332 3242187
16	00059ace3e3e9a53	0.24373632669448853	0.2333270013332367	0.23787954449653625	0.2325991988182068	0.2374608963727951	0.2332 3242187
17	000634272d0d44eb	0.24373632669448853	0.2333270013332367	0.23787954449653625	0.2325991988182068	0.2374608963727951	0.2332 3242187
18	000663aff0fffc80	0.24373632669448853	0.2333270013332367	0.23787954449653625	0.2325991988182068	0.2374608963727951	0.2332 3242187
19	000689dd34e20979	0.24373632669448853	0.2333270013332367	0.23787954449653625	0.2325991988182068	0.2374608963727951	0.2332 3242187

Run Info

Succeeded	True	Run Time	2940.2 seco
Exit Code	0	Queue Time	0 seconds
Docker Image Name	/python(Dockerfile)	Output Size	0
Timeout Exceeded	False	Used All Space	False
Failure Message			

Log

Download

```
Time   Line #  Log Message
3.4s         1  [NbConvertApp] Converting notebook script.ipynb to html
3.4s         2  [NbConvertApp] Executing notebook with kernel: python3
44.8s        3  2018-07-12 22:51:42.838822: I
                tensorflow/stream_executor/cuda/cuda_gpu_executor.cc:898]
                successful NUMA node read from SysFS had negative value (-1),
                but there must be at least one NUMA node, so returning NUMA nod
                zero
44.9s        4  2018-07-12 22:51:42.839549: I
                tensorflow/core/common_runtime/gpu/gpu_device.cc:1355] Found
                device 0 with properties:
                name: Tesla K80 major: 3 minor: 7 memoryClockRate(GHz): 0.8235
```

```
                pciBusID: 0000:00:04.0
                totalMemory: 11.17GiB freeMemory: 11.10GiB
                2018-07-12 22:51:42.839611: I
                tensorflow/core/common_runtime/gpu/gpu_device.cc:1434] Adding
                visible gpu devices: 0
47.3s       5   2018-07-12 22:51:45.350638: I
                tensorflow/core/common_runtime/gpu/gpu_device.cc:922] Device
                interconnect StreamExecutor with strength 1 edge matrix:
                2018-07-12 22:51:45.350695: I
                tensorflow/core/common_runtime/gpu/gpu_device.cc:928]      0
                2018-07-12 22:51:45.350707: I
                tensorflow/core/common_runtime/gpu/gpu_device.cc:941] 0:   N
47.4s       6   2018-07-12 22:51:45.351066: I
                tensorflow/core/common_runtime/gpu/gpu_device.cc:1052] Created
                TensorFlow device (/job:localhost/replica:0/task:0/device:GPU:0
                with 10629 MB memory) -> physical GPU (device: 0, name: Tesla
                K80, pci bus id: 0000:00:04.0, compute capability: 3.7)
2939.7s     7   [NbConvertApp] Writing 345963 bytes to __results__.html
2939.7s     8
2939.7s     10  Complete. Exited with code 0.
```

Comments (2)

Sort by: All Comments | Hotness

Please sign in to leave a comment.

Kevin Mader Kernel Author • Posted on Version 1 • 9 months ago 0

@datacanary this early enough to get any Kaggle swag?

DataCana... • Posted on Latest Version • 9 months ago 0

Nice! I'll be in touch directly. Incidentally sorry about the slow reply, I didn't get a notification from this at-mention (I've filed a bug report).

Similar Kernels