```python
# Author: Shashi Narayan
# Date: September 2016
# Project: Document Summarization
# H2020 Summa Project
# Comments: Jan 2017
# Improved for Reinforcement Learning
####################################

"""
Document Summarization Final Model
"""

from __future__ import absolute_import
from __future__ import division
from __future__ import print_function

import math
import os
import random
import sys
import time

import numpy as np
import tensorflow as tf

import model_docsum
from my_flags import FLAGS
import model_utils
```

```
###################### Define Final Network ############################

class MY_Model:
  def __init__(self, sess, vocab_size):
    dtype = tf.float16 if FLAGS.use_fp16 else tf.float32

    ### Few variables that has been initianlised here
    # Word embedding variable
    self.vocab_embed_variable = model_utils.get_vocab_embed_variable(vocab_size)

    ### Define Place Holders
    self.document_placeholder = tf.placeholder("int32", [None,
                                                    (FLAGS.max_doc_length + FLAGS.max_title_length + FLAGS.max_image_length),
                                                    FLAGS.max_sent_length], name='doc-ph')
    self.label_placeholder = tf.placeholder(dtype, [None, FLAGS.max_doc_length, FLAGS.target_label_size], name='label-ph')
    self.weight_placeholder = tf.placeholder(dtype, [None, FLAGS.max_doc_length], name='weight-ph')

    # Reward related place holders: Pass both rewards as place holders to make them constant for rl optimizer
    self.actual_reward_multisample_placeholder = tf.placeholder(dtype, [None, 1], name='actual-reward-multisample-ph') # [FLAGS.batch_size, Single
Sample]

    # Self predicted label placeholder
    self.predicted_multisample_label_placeholder = tf.placeholder(dtype, [None, 1, FLAGS.max_doc_length, FLAGS.target_label_size], name='pred-
multisample-label-ph')

    # Only used for test/validation corpus
    self.logits_placeholder = tf.placeholder(dtype, [None, FLAGS.max_doc_length, FLAGS.target_label_size], name='logits-ph')

    ### Define Policy Core Network: Consists of Encoder, Decoder and Convolution.
    self.extractor_output, self.logits = model_docsum.policy_network(self.vocab_embed_variable, self.document_placeholder, self.label_placeholder)

    ### Define Reward-Weighted Cross Entropy Loss
    self.rewardweighted_cross_entropy_loss_multisample = model_docsum.reward_weighted_cross_entropy_loss_multisample(self.logits,
self.predicted_multisample_label_placeholder,
                                                                                                          self.actual_reward_multisample_placeholder,
self.weight_placeholder)
```

```
    ### Define training operators
    self.train_op_policynet_expreward = model_docsum.train_neg_expectedreward(self.rewardweighted_cross_entropy_loss_multisample)

    # accuracy operation : exact match
    self.accuracy = model_docsum.accuracy(self.logits, self.label_placeholder, self.weight_placeholder)
    # final accuracy operation
    self.final_accuracy = model_docsum.accuracy(self.logits_placeholder, self.label_placeholder, self.weight_placeholder)

    # Create a saver.
    self.saver = tf.train.Saver(tf.all_variables(), max_to_keep=None)

    # Scalar Summary Operations
    self.rewardweighted_ce_multisample_loss_summary = tf.summary.scalar("rewardweighted-cross-entropy-multisample-loss",
self.rewardweighted_cross_entropy_loss_multisample)
    self.taccuracy_summary = tf.summary.scalar("training_accuracy", self.accuracy)
    self.vaccuracy_summary = tf.summary.scalar("validation_accuracy", self.final_accuracy)

    # # Build the summary operation based on the TF collection of Summaries.
    # # self.summary_op = tf.merge_all_summaries()

    # Build an initialization operation to run below.
    init = tf.initialize_all_variables()

    # Start running operations on the Graph.
    sess.run(init)

    # Create Summary Graph for Tensorboard
    self.summary_writer = tf.summary.FileWriter(FLAGS.train_dir, sess.graph)
```