```
#!/usr/bin/env python

"""Space Prediction Preprocessing
"""

import numpy as np
import h5py
import argparse
import sys
import re
import codecs

SPACE = '<space>'
START = '<s>'
FILE_PATHS = {"PTB": ("data/train_chars.txt",
              "data/valid_chars.txt",
              "data/valid_chars_kaggle.txt",
              "data/test_chars.txt")}
args = {}
char_to_idx = {}

def build_indices(file_list):
  inp = {}
  out = {}
  for filename in file_list:
    if filename:
      with codecs.open(filename, "r", encoding="latin-1") as f:
        print('Converting ' + filename + ' to indices...')
        lines = f.readlines()
        if len(lines) == 1: # Train or valid
          chars = [char_to_idx[str(c)] for c in lines[0].split()]
          spaces = [int(c == 1) for c in chars] # 0 or 1 depending on
presence of space or not
          inp[filename] = chars
          out[filename] = spaces[1:] + [0] # Offset by 1 as _next_
char is label
        else: # Valid kaggle or test
          inp_arr = []
          out_arr = []

          # Find longest example and standardize length to max
          longest = 0
          for line in lines:
            l = len(line.split())
            if l > longest: longest = l

          for line in lines:
            chars = [char_to_idx[str(c)] for c in line.split()]
            nspaces = chars.count(1)
            chars = [c for c in chars if c != 1] # Remove spaces
```

```
            # Standardize length with padding
            chars = chars + [char_to_idx['</s>']] * (longest -
len(chars))
            inp_arr.append(chars)
            out_arr.append(nspaces)

          inp[filename] = inp_arr
          out[filename] = out_arr
  return inp, out

def build_batches(vals, l, b):
  n = len(vals)
  batches = []
  for i in range(n / (b * l)):
    batch = []
    for j in range(b):
      batch.append(vals[(n / b * j) + (l * i): (n / b * j) + (l * (i +
1))])

    batches.append(batch)
  return batches

def build_char_dict(file_list):
  last_idx = 3
  char_to_idx[SPACE] = 1
  char_to_idx[START] = 2
  for filename in file_list:
    if filename:
      with codecs.open(filename, "r", encoding="latin-1") as f:
        count = 0
        for line in f:
          count = count + 1
          letters = line.split(' ')
          for l in letters:
            l = l.rstrip() # Remove pesky line feed from </s>
            if l not in char_to_idx:
              char_to_idx[l] = last_idx
              last_idx = last_idx + 1

def main(arguments):
  global args
  parser = argparse.ArgumentParser(
    description=__doc__,
    formatter_class=argparse.RawDescriptionHelpFormatter)
  parser.add_argument('dataset', help="Data set",
            type=str)
  args = parser.parse_args(arguments)
  dataset = args.dataset
  train, valid, valid_kaggle, test = FILE_PATHS[dataset]
```

```
  seq = 50
  batch = 32

  build_char_dict([train, valid, valid_kaggle, test])
  input_dict, output_dict = build_indices([train, valid, valid_kaggle,
test])
  train_input_batch = build_batches(input_dict[train], seq, batch)
  train_output_batch = build_batches(output_dict[train], seq, batch)
  valid_input_batch = build_batches(input_dict[valid], seq, batch)
  valid_output_batch = build_batches(output_dict[valid], seq, batch)

  train_input_cb = np.array(input_dict[train], dtype=np.int32)
  train_output_cb = np.array(output_dict[train], dtype=np.int32)
  valid_input_cb = np.array(input_dict[valid], dtype=np.int32)
  valid_output_cb = np.array(output_dict[valid], dtype=np.int32)

  valid_kaggle_input = np.array(input_dict[valid_kaggle],
dtype=np.int32)
  valid_kaggle_output = np.array(output_dict[valid_kaggle],
dtype=np.int32)
  test_input = np.array(input_dict[test], dtype=np.int32)

  filename = args.dataset + '.hdf5'
  print('Writing to ' + filename)
  with h5py.File(filename, "w") as f:
    f['train_input_cb'] = train_input_cb
    f['train_output_cb'] = train_output_cb
    f['valid_input_cb'] = valid_input_cb
    f['valid_output_cb'] = valid_output_cb

    f['train_input'] = train_input_batch
    f['train_output'] = train_output_batch
    f['valid_input'] = valid_input_batch
    f['valid_output'] = valid_output_batch
    f['valid_kaggle_input'] = valid_kaggle_input
    f['valid_kaggle_output'] = valid_kaggle_output
    f['test_input'] = test_input

    f['nclasses'] = np.array([2], dtype=np.int32) # space or not
    f['nletters'] = np.array([len(char_to_idx)], dtype=np.int32)
    f['seq'] = np.array([seq], dtype=np.int32)
    f['batch'] = np.array([batch], dtype=np.int32)


if __name__ == '__main__':
  sys.exit(main(sys.argv[1:]))
```