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

"""Language modeling preprocessing
"""

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

# Your preprocessing, features construction, and word2vec code.

START = '<s>'
END = '</s>'
UNKNOWN = '<unk>'

FILE_PATHS = {"PTB": ("data/train.txt",
              "data/valid.txt",
              "data/test_blanks.txt",
              "data/words.dict"),
              "PTB1000": ("data/train.1000.txt",
              "data/valid.1000.txt",
              "data/test_blanks.txt",
              "data/words.1000.dict")}
args = {}
word_to_idx = {}
word_freq = {}

def build_ngrams(file_list, ngram):
  input_ngrams = {}
  output = {}
  for filename in file_list:
    if filename:
      input_ngrams[filename] = []
      output[filename] = []
      with codecs.open(filename, "r", encoding="latin-1") as f:
        print('Building ngrams from ' + filename + '...')

        iterlines = iter(f)
        next(iterlines) # Skip first line because it's utter nonsense
        for line in iterlines:
          words = [word_to_idx[str(w)] for w in line.split()]

          # Padding
          start = [word_to_idx[START]] * (ngram - 1)
          end = [word_to_idx[END]]
          words = start + words + end

          for i in xrange(ngram, len(words)+1):
            context = words[i-ngram:i]
            inp = context[:-1]
            out = context[-1]
            input_ngrams[filename].append(inp)
            output[filename].append(out)
  return input_ngrams, output

def build_test_ngrams(filename, ngram):
  input_ngrams = []
  output = []
  if filename:
    with codecs.open(filename, "r", encoding="latin-1") as f:
      print('Building ngrams from ' + filename + '...')
      while True:
        dist = str(f.readline()).split()[1:]
        words = str(f.readline()).split()[1:-1]
        if len(dist) == 0: break # eof

        # Replace unseen words with unknown tag
        for i in range(len(dist)):
          if dist[i] not in word_to_idx:
            dist[i] = UNKNOWN
        for i in range(len(words)):
          if words[i] not in word_to_idx:
            words[i] = UNKNOWN

        # Convert to indexes
        dist = [word_to_idx[w] for w in dist]
        words = [word_to_idx[w] for w in words]

        # Padding
        start = [word_to_idx[START]] * (ngram - 1)
        words = start + words
        inp = words[-ngram+1:]

        input_ngrams.append(inp)
        output.append(dist)
  return input_ngrams, output

all_idx = []
def build_word_dict(filename):
  last_idx = -1
  with codecs.open(filename, "r", encoding="latin-1") as f:
    for line in f:
      l = line.split()
      idx = int(l[0])
      word = str(l[1])
      freq = int(l[2])
      word_to_idx[word] = idx
      word_freq[idx] = freq
      last_idx = idx
      all_idx.append(idx)
  word_to_idx[START] = last_idx + 1; all_idx.append(last_idx + 1)
  word_to_idx[END] = last_idx + 2; all_idx.append(last_idx + 2)

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

  build_word_dict(words)
  input_dict, output_dict = build_ngrams([train, valid], ngram)
  train_input = np.array(input_dict[train], dtype=np.int32)
  train_output = np.array(output_dict[train], dtype=np.int32)
  valid_input = np.array(input_dict[valid], dtype=np.int32)
  valid_output = np.array(output_dict[valid], dtype=np.int32)

  test_input, test_output = build_test_ngrams(test, ngram)
  test_input = np.array(test_input, dtype=np.int32)
  test_output = np.array(test_output, dtype=np.int32)

  V = max(all_idx)
  # V = len(word_to_idx)
  C = len(word_to_idx)

  filename = args.dataset + '_' + str(ngram) + 'gram.hdf5'
  with h5py.File(filename, "w") as f:
    f['train_input'] = train_input
    f['train_output'] = train_output
    if valid:
      f['valid_input'] = valid_input
      f['valid_output'] = valid_output
    if test:
      f['test_input'] = test_input
      f['test_output'] = test_output

    f['nwords'] = np.array([V], dtype=np.int32)
    f['nclasses'] = np.array([C], dtype=np.int32)
    f['ngram'] = np.array([ngram], dtype=np.int32)

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