```python
import nltk
import pickle
import argparse
from collections import Counter
from pycocotools.coco import COCO


class Vocabulary(object):
    """Simple vocabulary wrapper."""
    def __init__(self):
        self.word2idx = {}
        self.idx2word = {}
        self.idx = 0

    def add_word(self, word):
        if not word in self.word2idx:
            self.word2idx[word] = self.idx
            self.idx2word[self.idx] = word
            self.idx += 1

    def __call__(self, word):
        if not word in self.word2idx:
            return self.word2idx['<unk>']
        return self.word2idx[word]

    def __len__(self):
        return len(self.word2idx)

def build_vocab(json, threshold):
    """Build a simple vocabulary wrapper."""
    coco = COCO(json)
    counter = Counter()
    ids = coco.anns.keys()
    for i, id in enumerate(ids):
        caption = str(coco.anns[id]['caption'])
        tokens = nltk.tokenize.word_tokenize(caption.lower())
        counter.update(tokens)

        if (i+1) % 1000 == 0:
            print("[{}/{}] Tokenized the captions.".format(i+1, len(ids)))

    # If the word frequency is less than 'threshold', then the word is discarded.
    words = [word for word, cnt in counter.items() if cnt >= threshold]

    # Create a vocab wrapper and add some special tokens.
    vocab = Vocabulary()
    vocab.add_word('<pad>')
    vocab.add_word('<start>')
    vocab.add_word('<end>')
    vocab.add_word('<unk>')

    # Add the words to the vocabulary.
    for i, word in enumerate(words):
        vocab.add_word(word)
    return vocab

def main(args):
    vocab = build_vocab(json=args.caption_path, threshold=args.threshold)
    vocab_path = args.vocab_path
```

```
60      with open(vocab_path, 'wb') as f:
61          pickle.dump(vocab, f)
62      print("Total vocabulary size: {}".format(len(vocab)))
63      print("Saved the vocabulary wrapper to '{}'".format(vocab_path))
64
65
66  if __name__ == '__main__':
67      parser = argparse.ArgumentParser()
68      parser.add_argument('--caption_path', type=str,
69
    default='data/annotations/captions_train2014.json',
70                          help='path for train annotation file')
71      parser.add_argument('--vocab_path', type=str,
    default='./data/vocab.pkl',
72                          help='path for saving vocabulary wrapper')
73      parser.add_argument('--threshold', type=int, default=4,
74                          help='minimum word count threshold')
75      args = parser.parse_args()
76      main(args)
```