```
import nltk
nltk.download('punkt')
from nltk.tokenize import word_tokenize
from nltk.util import ngrams
from collections import Counter
import pickle
import numpy

# read in pickled dicts
with open('/ngram_meta_dict_2.pkl','rb') as file:
  ngram_meta_dict = pickle.load(file)

languages = ['English','French','Italian']
vocabulary_size = 0
for language in languages:
  vocabulary_size = vocabulary_size + len(ngram_meta_dict[language]['unigram'])
  # print(f"Vocabulary size after adding {language}:", vocabulary_size)

def line_probability(line, unigram_dict, bigram_dict):

  # get line's unigrams and bigrams
  tokens = word_tokenize(line)
  unigrams = tokens # unigrams are the tokens
  bigrams = list(ngrams(tokens,2))

  # print(f"Line has {len(unigrams)} unigrams")
  # print(f"Line has {len(bigrams)} bigrams")

  # check for unigram count
  unigram_counts_list = [0] * len(unigrams)
  for i, unigram in enumerate(unigrams):
    if unigram in unigram_dict:
      unigram_count = unigram_dict[unigram]
    else:
      unigram_count = 0
    unigram_counts_list[i] = unigram_count

  # check for bigram count
  bigram_counts_list = [0] * len(bigrams)
  for i, bigram in enumerate(bigrams):
    if bigram in bigram_dict:
      bigram_count = bigram_dict[bigram]
    else:
      bigram_count = 0
    bigram_counts_list[i] = bigram_count

  # calculate probability
  total_probability = 1
  for i, bigram in enumerate(bigrams):
    total_probability = total_probability * \
     (bigram_counts_list[i] + 1)/(unigram_counts_list[i] + vocabulary_size)

  # print(f"Total probability for {language}:{total_probability}")
  return total_probability


line_count = 0 #use same line count for test and solution file

# read in test file
with open('/LangId.test.txt','r') as file:
  for line in file:
    line_count += 1 # determine total line_count
  file.seek(0)
  line_probabilities = numpy.zeros((line_count,len(languages))) # stores probability of each language
  # stores the probability of most likely language, and which language that is:
  greatest_line_probability = numpy.zeros((2,line_count))
  iterations = 0

  # nested for loop to loop through each line and each language
  # nested for loop to loop through each line and each language
  for line_num, line in enumerate(file):
    for language_num, language in enumerate(languages):
      # calculate and store line probability for the current language:
      line_probabilities[line_num][language_num] = line_probability(line,ngram_meta_dict[language]['unigram'], \
      ngram_meta_dict[language]['bigram'])

      # update most likely language
      if greatest_line_probability[1][line_num] < line_probabilities[line_num][language_num]:
        greatest_line_probability[0][line_num] = language_num
        greatest_line_probability[1][line_num] = line_probabilities[line_num][language_num]
    # print(f"Processing line:'{line}'")
    # print(f'Most likely language:{languages[int(greatest_line_probability[0][line_num])]}')
    # iterations += 1
    # if iterations > 4:
    #  break

# write predicted languages to a file
with open('predicted_languages.txt', 'w') as file:
    for line in range(line_count):
        file.write(f"{languages[int(greatest_line_probability[0][line])]}\n")

total_correct = 0

# read in the correct answer file
with open('/LangId.sol.txt','r') as file:
  language_solutions = numpy.zeros(line_count)
  for line_num, line in enumerate(file):
    tokens = word_tokenize(line)
    # loop through languages to find which one matches the solution language for this line
    for language_num, language in enumerate(languages):
      if language == tokens[1]:
        language_solutions[line_num] = language_num

        #check if the solution matches the predicted (greatest_line_probability is the predicted)
        if int(language_solutions[line_num]) == int(greatest_line_probability[0][line_num]):
          total_correct += 1
        else:
          # print mess ups
          print(f"Messed up this line: {tokens[0]}") # tokens[0] is the line number
          print(f"""Thought it was {languages[int(greatest_line_probability[0][line_num])]}, but it was {languages[int(
          print(" ")

# check % correct
percent_correct = total_correct/line_count
print(f'PERCENT CORRECT aka ACCURACY: {percent_correct*100:.2f}%')
```

```
[nltk_data] Downloading package punkt to /root/nltk_data...
[nltk_data]   Package punkt is already up-to-date!
Messed up this line: 24
Thought it was English, but it was French

Messed up this line: 44
Thought it was French, but it was Italian

Messed up this line: 92
Thought it was French, but it was English

Messed up this line: 187
Thought it was English, but it was Italian

Messed up this line: 191
Thought it was English, but it was French

Messed up this line: 247
Thought it was English, but it was Italian

Messed up this line: 277
Thought it was English, but it was Italian

Messed up this line: 279
```

```
Thought it was English, but it was French

PERCENT CORRECT aka ACCURACY: 97.33%
```