from \_\_future\_\_ import absolute\_import  
from \_\_future\_\_ import division, print\_function, unicode\_literals  
import itertools  
from copy import deepcopy  
class Ngrams(object):  
 def \_\_init\_\_(self, ngrams={}, exclusive=True):  
 if exclusive:  
 self.\_ngrams = set(ngrams)  
 else:  
 self.\_ngrams = list(ngrams)  
 self.exclusive = exclusive  
 def add(self, o):  
 if self.exclusive:  
 self.\_ngrams.add(o)  
 else:  
 self.\_ngrams.append(o)  
 def \_\_len\_\_(self):  
 return len(self.\_ngrams)  
 def intersection(self, o):  
 if self.exclusive:  
 inter\_set = self.\_ngrams.intersection(o.\_ngrams)  
 return Ngrams(inter\_set, exclusive=True)  
 else:  
 other\_list = deepcopy(o.\_ngrams)  
 inter\_list = []  
 for e in self.\_ngrams:  
 try:  
 i = other\_list.index(e)  
 except ValueError:  
 continue  
 other\_list.pop(i)  
 inter\_list.append(e)  
 return Ngrams(inter\_list, exclusive=False)  
 def union(self, \*ngrams):  
 if self.exclusive:  
 union\_set = self.\_ngrams  
 for o in ngrams:  
 union\_set = union\_set.union(o.\_ngrams)  
 return Ngrams(union\_set, exclusive=True)  
 else:  
 union\_list = deepcopy(self.\_ngrams)  
 for o in ngrams:  
 union\_list.extend(o.\_ngrams)  
 return Ngrams(union\_list, exclusive=False)  
def \_get\_ngrams(n, text, exclusive=True):  
 ngram\_set = Ngrams(exclusive=exclusive)  
 text\_length = len(text)  
 max\_index\_ngram\_start = text\_length - n  
 for i in range(max\_index\_ngram\_start + 1):  
 ngram\_set.add(tuple(text[i:i + n]))  
 return ngram\_set  
def \_split\_into\_words(sentences):  
 return list(itertools.chain(\*[\_.split(" ") for \_ in sentences]))  
def \_get\_word\_ngrams(n, sentences, exclusive=True):  
 assert len(sentences) > 0  
 assert n > 0  
 words = \_split\_into\_words(sentences)  
 return \_get\_ngrams(n, words, exclusive=exclusive)  
def \_len\_lcs(x, y):  
 table = \_lcs(x, y)  
 n, m = len(x), len(y)  
 return table[n, m]