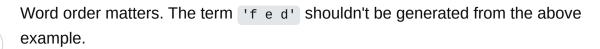
How do I generate multi-word terms recursively?

Asked 15 years, 8 months ago Modified 15 years ago Viewed 731 times



Say I have a string of words: $\ 'a\ b\ c\ d\ e\ f'\$. I want to generate a list of multi-word terms from this string.







Edit: Also, words should not be skipped. 'a c', or 'b d f' shouldn't be generated.

1

What I have right now:

```
doc = 'a b c d e f'
terms= []
one_before = None
two_before = None
for word in doc.split(None):
    terms.append(word)
    if one_before:
        terms.append(' '.join([one_before, word]))
    if two_before:
        terms.append(' '.join([two_before, one_before, word]))
    two_before = one_before
    one_before = word

for term in terms:
    print term
```

Prints:

```
a b c c b c a b c d c d b c d e d e c d e f e f d e f
```

How would I make this a recursive function so that I can pass it a variable maximum number of words per term?

Application:

I'll be using this to generate multi-word terms from readable text in HTML documents. The overall goal is a latent semantic analysis of a large corpus (about two million documents). This is why keeping word order matters (Natural Language Processing and whatnot).

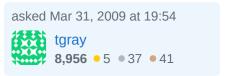
python recursion

Share

edited Apr 1, 2009 at 12:41

Improve this question

Follow



For simplicity I substituted single letters for words. – tgray Mar 31, 2009 at 19:59

did you mean "variable maximum number of terms per words"? because it doesn't make sense to me in current form. - SilentGhost Mar 31, 2009 at 20:00

I think the real question here is, does it need to be recursive to do the job? Is there a requirement for recursion here? - Dan Coates Mar 31, 2009 at 20:03

@SilentGhost, If I pass the function 'max words=4', I want to get back terms where len(term.split(None)) <= 4. - tgray Mar 31, 2009 at 20:11

@Dan Coates, No it's not a requirement, I was under the impression that a recursive function could perform the analysis faster than a loop, but after further research I find that may not be the case. - tgray Mar 31, 2009 at 20:51

4 Answers

Sorted by:

Highest score (default)





This isn't recursive, but I think it does what you want.

if index + n < len(words):</pre>

print ' '.join(words[index:index+n+1])













doc = 'a b c d e f'

max = 3

words = doc.split(None)

for index in xrange(len(words)): for n in xrange(max):

```
def find_terms(words, max_words_per_term):
    if len(words) == 0: return []
```

```
return [" ".join(words[:i+1]) for i in xrange(min(len(words),
max_words_per_term))] + find_terms(words[1:], max_words_per_term)

doc = 'a b c d e f'
words = doc.split(None)
for term in find_terms(words, 3):
    print term
```

Here's the recursive function again, with some explaining variables and comments.

```
def find_terms(words, max_words_per_term):
    # If there are no words, you've reached the end. Stop.
    if len(words) == 0:
        return []

# What's the max term length you could generate from the remaining
# words? It's the lesser of max_words_per_term and how many words
# you have left.
max_term_len = min(len(words), max_words_per_term)

# Find all the terms that start with the first word.
initial_terms = [" ".join(words[:i+1]) for i in xrange(max_term_len)]

# Here's the recursion. Find all of the terms in the list
# of all but the first word.
other_terms = find_terms(words[1:], max_words_per_term)

# Now put the two lists of terms together to get the answer.
return initial_terms + other_terms
```

Share

edited Apr 1, 2009 at 12:55

answered Mar 31, 2009 at 20:07

Patrick McElhaney **59.1k** • 41 • 137 • 169

Improve this answer

Follow

It looks like I'll have to use the first solution you provided. Python won't let a function recurs more than 999 times. My test document had about 1750 words and it is on the small side.

```
    tgray Apr 1, 2009 at 12:45
```

That makes sense. The recursive solution was fun to work out, but not really practical.

```
- Patrick McElhaney Apr 1, 2009 at 13:14
```

If you really want to have deep recursion, you can increase the recursion limit with sys.setrecursionlimit. But the iterative solution is probably better here anyway. – Kiv Dec 19, 2009 at 15:21



I would suggest that you should make your function a generator and then generate required number of terms. You would need to change print to yield (and make the whole block function, obviously).



You might have a look at <u>itertools</u> module as well, it's fairly useful for kind of work you do.



9

Share Improve this answer Follow

answered Mar 31, 2009 at 20:00





Why are you doing this? You can instead just use a for loop and <u>itertools.combinations()</u>.



Share Improve this answer Follow

answered Mar 31, 2009 at 20:01







Good suggestion, but I need the order to be preserved. Example: 'a b c' creates ['a', 'b', 'a b', 'c', 'b c', 'a b c'], but not 'b a' or 'c b a'. — tgray Mar 31, 2009 at 20:18

Sorry for the confusion, it also shouldn't skip words. The doc "The quick brown fox jumped over the fence" shouldn't have "brown fence" as a term. Is there a way to use itertools to do this? - tgray Apr 1, 2009 at 12:36

Not that I can think of at the moment. I did notice that after I replied, but yeah. Still, itertools is nice. It should always be the first place to look. – Devin Jeanpierre Apr 1, 2009 at 16:48



What you are looking for is N-gram algorithm. That will give you [a,ab,b,bc,c,cd,...].



Share Improve this answer Follow

answered Dec 19, 2009 at 13:59





