Mô hình ngôn ngữ và ứng dụng cho kiểm lỗi chính tả

Mục tiêu: xây dựng chương trình kiểm lỗi cú pháp tiếng Anh đơn giản

Bài 2:

a) Cải tiến mô hình bằng cách sử dụng smoothing là phương pháp interpolation theo “Stupid backoff” (Brants et al. 2007)

b) So sánh với kết quả bài 1

c) Dùng mô hình vừa xây dựng để sinh các từ tiếp theo với một chuỗi từ cho trước.

d) Kết hợp với hàm tính khoảng cách giữa các từ để dự đoán từ đúng cho vị trí từ sai chính tả. (from difflib import get_close_matches)

Đọc input

```
# đọc file
filename='data/tedtalk.txt'
lines=[]
count=0
#Max=-1
Max=10000
with open(filename,'r') as f:
    for s in f:
        count+=1
        if count>Max and Max!=-1:
            break
        lines.append(s.strip())
print(len(lines))
print(lines[:5])
```

```
10000
["Here are two reasons companies fail: they only do more of the same, or they only do what's new"
```

Thực hiện tokenize tập input

Thêm '<s>' vào trước mỗi câu, thêm '</s>' vào sau mỗi câu.

```
# tokenize sentences
import nltk
sentences=[]
all_tokens_count=0
for line in lines:
    tokens = nltk.word_tokenize(line.lower())
    all_tokens_count+=len(tokens)
    #sentences.append(tokens)
    sentences.append(['<s>']+tokens+['</s>'])
print('all_tokens_count=',all_tokens_count)
```

```
print('all_tokens_count=',all_tokens_count)
print(len(sentences))
print(sentences[:5])
```

```
all_tokens_count= 186299
10000
[['<s>', 'here', 'are', 'two', 'reasons', 'companies', 'fail', ':', 'they', 'only', 'do', 'more',
```

Counting 1-gram

```
# counting 1-gram
from collections import Counter
counter_unigram=Counter()
for sent in sentences:
    counter_unigram.update(sent)
V=len(counter_unigram)
print('V=',V)
n=0
for gram in counter_unigram:
    n+=counter_unigram[gram]
n=n-counter_unigram['<s>']-counter_unigram['</s>']
print('n=',n)
print(counter_unigram['the'])
print(counter_unigram['he'])
```

```
V= 12653
n= 186299
8059
399
```

Counting 2-gram

```
from nltk import ngrams
bi_grams=[]
for sent in sentences:
    gram2=ngrams(sent,2)
    bi_grams.extend(gram2)
print(len(bi_grams))

for i in range(3):
    print(bi_grams[i])

counter_bigram = Counter(bi_grams)
print('V=',len(counter_bigram))
print(counter_bigram[('here','are')])
```

```
196299
('<s>', 'here')
('here', 'are')
('are', 'two')
V= 79306
9
```

Counting 3-gram

```
tri_grams=[]
for sent in sentences:
    gram3=ngrams(sent,3)
    tri_grams.extend(gram3)
print(len(tri_grams))

for i in range(3):
    print(tri_grams[i])

counter_trigram = Counter(tri_grams)
print('V=',len(counter_trigram))
print(counter_trigram[('here','are','two')])
```

```
186299
('<s>', 'here', 'are')
('here', 'are', 'two')
('are', 'two', 'reasons')
V= 141749
2
```

Xây dựng dict từ 3-gram

```
tri_dict={}
set_tri_grams = set(tri_grams)
for gram in set_tri_grams:
    key=(gram[0],gram[1])
    if key in tri_dict.keys():
        tri_dict[key].append(gram[2])
    else:
        tri_dict[key]=[gram[2]]


print(len(tri_dict))
print(list(tri_dict.keys())[:3])
print('here are: ?')
print(tri_dict[('here','are')])
```

```
75672
[('they', 'did'), ('in', 'north'), ('becoming', 'so')]
here are: ?
['some', 'all', 'two', 'wondering', 'the', 'worth']
```

Dùng mô hình vừa xây dựng để sinh các từ tiếp theo với một chuỗi từ cho trước

```
# dự đoán từ tiếp theo
seq='here are the most'
tokens=nltk.word_tokenize(seq.lower())
tokens=['<s>']+tokens
print(tokens)
```

```
i=len(tokens)-1
key=(tokens[i-1],tokens[i])
#print(tri_dict[key])
dict_nextword={}

for next_word in tri_dict[key]:
    #print(counter_trigram[(key[0],key[1],next_word)])
    dict_nextword[next_word]=counter_trigram[(key[0],key[1],next_word)]
sorted_dict=sorted(dict_nextword,key=dict_nextword.__getitem__,reverse=True)

#print(dict_nextword)
print(sorted_dict[0])
```

```
['<s>', 'here', 'are', 'the', 'most']
important
```

Dự đoán chuỗi K từ tiếp theo

```
# dự đoán chuỗi K từ tiếp theo
seq='here are the'
tokens=nltk.word_tokenize(seq.lower())
tokens=['<s>']+tokens
print(tokens)
N=10
count=0
while count<N:
    count+=1
    i=len(tokens)-1
    key=(tokens[i-1],tokens[i])
    #print(tri_dict[key])
    dict_nextword={}
    if not (key in tri_dict.keys()):
        break
    for next_word in tri_dict[key]:
        dict_nextword[next_word]=counter_trigram[(key[0],key[1],next_word)]
    sorted_dict=sorted(dict_nextword,key=dict_nextword.__getitem__,reverse=True)
    #print(dict_nextword)
    #print(sorted_dict[0])
    tokens.append(sorted_dict[0])
    seq+= ' '+sorted_dict[0]
print(seq)
```

```
['<s>', 'here', 'are', 'the']
here are the ones that need to do it </s>
```

Kết hợp với hàm tính khoảng cách giữa các từ để dự đoán từ đúng cho vị trí từ sai chính tả. (from difflib import get_close_matches)

```
# for spelling
from difflib import get_close_matches
#get_close_matches('appel', ['ape', 'apple', 'peach', 'puppy'])
```

```
# dự đoán từ tiếp theo
seq='here are the most'
#errword='impotant'
errword='beutimport'
tokens=nltk.word_tokenize(seq.lower())
tokens=['<s>']+tokens
print(tokens)
i=len(tokens)-1
key=(tokens[i-1],tokens[i])
#print(tri_dict[key])
dict_nextword={}
for next_word in tri_dict[key]:
    #print(counter_trigram[(key[0],key[1],next_word)])
    dict_nextword[next_word]=counter_trigram[(key[0],key[1],next_word)]
sorted_dict=sorted(dict_nextword,key=dict_nextword.__getitem__,reverse=True)
#print(dict_nextword)
print(sorted_dict[:10])
closed_words=get_close_matches(errword,sorted_dict,n=10,cutoff=0.4)
print(closed_words[:10])
```

```
['<s>', 'here', 'are', 'the', 'most']
['important', 'powerful', 'pressing', '</s>', 'for', 'beautiful', 'dangerous', 'famous', 'extraor
['important', 'beautiful', 'revolutionary', 'impoverished', 'intimate', 'part', 'ambitious', 'pri
```

Double-click (or enter) to edit