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 1:

a) Xây dựng mô hình ngôn ngữ dựa trên n-gram sử dụng phương pháp smoothing là Laplace, cho các mô hình:

- 1-gram
- 2-gram
- 3-gram

b) Tính xác suất của một câu và tính Perplexity của một câu dựa theo 1-gram, 2-gram, 3-gram

c) Phân tích kết quả

Đọc file 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"
```

Cài đặt thư viện xử lý ngôn ngữ nltk

```
!pip install nltk
```

```
Collecting nltk
  Downloading nltk-3.5.zip (1.4 MB)
Collecting click
  Downloading click-7.1.2-py2.py3-none-any.whl (82 kB)
Requirement already satisfied: joblib in c:\programdata\anaconda3\envs\dl\lib\site-packages (from
Collecting regex
  Downloading regex-2020.11.13-cp36-cp36m-win_amd64.whl (269 kB)
Collecting tqdm
  Downloading tqdm-4.54.1-py2.py3-none-any.whl (69 kB)
Building wheels for collected packages: nltk
```

```
  Building wheel for nltk (setup.py): started
  Building wheel for nltk (setup.py): finished with status 'done'
  Created wheel for nltk: filename=nltk-3.5-py3-none-any.whl size=1434680 sha256=1c2e91d1deba89e8
  Stored in directory: c:\users\administrator\appdata\local\pip\cache\wheels\de\5e\42\64abaeca668
Successfully built nltk
Installing collected packages: click, regex, tqdm, nltk
Successfully installed click-7.1.2 nltk-3.5 regex-2020.11.13 tqdm-4.54.1
```

```
import nltk
nltk.download('punkt')
```

```
[nltk_data] Downloading package punkt to
[nltk_data]     C:\Users\Administrator\AppData\Roaming\nltk_data...
[nltk_data]   Unzipping tokenizers\punkt.zip.
True
```

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(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
```

Viết hàm tính xác suất cho từng loại: 1-gram, 2-gram, 3-gram

```
# tính prob theo từng loại: 1-gram, 2-gram, 3-gram
def uni_prob(word):
    return max(1,counter_unigram[word])/all_tokens_count

def bi_prob(word1,word2):
    if counter_bigram[(word1,word2)]>0:
        return counter_bigram[(word1,word2)]/counter_unigram[word1]
    else:
        return 0.4*uni_prob(word2)

def tri_prob(word1,word2,word3):
    if counter_trigram[(word1,word2,word3)]>0:
        return counter_trigram[(word1,word2,word3)]/counter_bigram[(word1,word2)]
    else:
        return 0.4*bi_prob(word1,word2)
```

Viết hàm tính xác xuất cho một câu, normalize theo 1 từ

```
# tính xác suất của một câu, normalize theo 1 từ
def probLM(sent,n):
    if n>3 or n<1: # không xét trường hợp này
        return 0
    tokens=nltk.word_tokenize(sent.lower())
    tokens += ['<s>']+tokens
    prob=1
    for i in range(1,len(tokens)):
        if n==1:
            prob*=uni_prob(tokens[i])
        elif n==2:
            prob*=bi_prob(tokens[i-1],tokens[i])
        elif n==3:
            if i>=2:
                prob*=tri_prob(tokens[i-2],tokens[i-1],tokens[i])
            else:
                prob*=bi_prob(tokens[i-1],tokens[i])
    l=len(tokens)-1
    return prob**(1/l)
```

Kiểm tra xác xuất của 1 câu

```
sent='the human body with new abilities is no longer a question'
#sent='the human body with new from abilities is no longer a question'
#sent='A few years back from'
print('n=1')
pr=probLM(sent,1)
print('prob=',pr)
print('perplexity=',1/pr)
```

```
print('perplexity=',1/pr)

print('n=2')
pr=probLM(sent,2)
print('prob=',pr)
print('perplexity=',1/pr)

print('n=3')
pr=probLM(sent,3)
print('prob=',pr)
print('perplexity=',1/pr)
```

```
n=1
prob= 0.001493428059077296
perplexity= 669.6003827715966
n=2
prob= 0.018964321328103315
perplexity= 52.73059777351987
n=3
prob= 0.139268032225464
perplexity= 7.180398717640231
```

So sánh xác suất của 2 câu

```
# kiểm tra xem 2 xâu có xác suất hơn nhau thế nào, ví dụ cho bài toán speech to t
sent1='the human body with new abilities is no longer a question'
sent2='the human body with knew abilities is know longer a question'
pr=probLM(sent1,3)
print('prob=',pr)
print('perplexity=',1/pr)

pr=probLM(sent2,3)
print('prob=',pr)
print('perplexity=',1/pr)
```

```
prob= 0.139268032225464
perplexity= 7.180398717640231
prob= 0.003432760555032788
perplexity= 291.31073489349404
```