第三次大作业报告

孙博一　2020K8009970015

1.题目

目标：编写程序，实现 BLEU 值的计算。可以从网络上选择高质量的平行句对，利用所写的 BLEU 值计算程序和平行句对，计算三个不同翻译引擎（如百度、搜狗、微软“必应”等）的译文 BLEU 值。完成一份对比实验和分析报告。

2.设计思路

使用 google，搜狗和金山词霸三个翻译软件进行测试。（百度翻译被我用超限了，其他或多或少收费）。翻译了 400 句中文，与标准译文进行对应，最终比较出一个翻译水平最高的翻译软件。

使用 bleu 进行度量翻译软件的翻译标准。Bleu 的计算未使用脚本，自主设计实现。

代码如下。

3.代码讲解

3.1 第一部分：文件的解析与本地保存

```
# 读取文件
from bs4 import BeautifulSoup
import math

def read_sgm_file(file_path):
    with open(file_path, 'r', encoding='utf-8') as file:
        content = file.read()
    return content

def parse_sgm_content(content):
    soup = BeautifulSoup(content, 'html.parser')
    english_text = [text.text for text in soup.find_all('seg',
id=lambda x: x != '0')]
    return english_text

file_path_en = 'WMT-18\\newstest2018-zhen-ref-ts.en.sgm'
file_path_zh = 'WMT-18\\newstest2018-zhen-src-ts.zh.sgm'
en_content = read_sgm_file(file_path_en)
zh_content = read_sgm_file(file_path_zh)
en_text = parse_sgm_content(en_content)
zh_text = parse_sgm_content(zh_content)
print(en_text[0])
print(zh_text[0])
#保存在本地
with open('WMT-18\\newstest2018-zhen-ref-ts.en.txt', 'w',
encoding='utf-8') as file:
```

```
    file.write('\n'.join(en_text))
with open('WMT-18\\newstest2018-zhen-src-ts.zh.txt', 'w',
encoding='utf-8') as file:
    file.write('\n'.join(zh_text))
```

使用 beautifulsoup 库进行文件的扫描，主要是使用 find_all 获取其中有用的信息，使用 get_text 完成内容的提取，提取出来的中文 1 和英文文件分别保存。

保存的内容几位翻译过后的中英文：

Last week, the broadcast of period drama “Beauty Private Kitchen” was temporarily halted, and accidentally triggered heated debate about faked ratings of locally produced dramas.

上周，古装剧《美人私房菜》临时停播，意外引发了关于国产剧收视率造假的热烈讨论。

3.2 第二部分：翻译

google，搜狗和金山词霸，翻译了 414 个句子（随便截的），使用文档翻译，因为有很多翻译软件的文档翻译存在字数和到出现的限制，所以选择了三个不收费的。

选取前 400 个句子作为参考，能够更加的准确。

翻译产生的文件都放在特定的文件夹下了。

```
with open('WMT-18\\trans_result\\google.txt', 'r', encoding='utf-8') as
file:
    google_text = file.read().split('\n')
    file.close()
with open('WMT-18\\trans_result\\sg.txt', 'r', encoding='utf-8') as
file:
    sg_text = file.read().split('\n')
    file.close()
with open('WMT-18\\trans_result\\jscb.txt', 'r', encoding='utf-8') as
file:
    jscb_text = file.read().split('\n')
    file.close()


with open('WMT-18\\newstest2018-zhen-ref-ts.en.txt', 'r',
encoding='utf-8') as file:
    ref_text = file.read().split('\n')
    file.close()
```

3.3 第三部分：

候选组的划分，我们以 google 翻译为例子：

```
    # google
    word = ''
    for j in range(len(google_text[i])):
        char = google_text[i][j]
        if 'A' <= char <= 'Z' or 'a' <= char <= 'z' or '0' <= char <=
'9':
            word = word + char
            continue
        else:
```

```
            if word != '':
                # print(word)
                google_p1.append(word)
                if char != ' ' and char != '\n':
                    google_p1.append(char)
                    # print(char)
                word = ''
    print(google_p1)
    google_p2 = []
    google_p3 = []
    google_p4 = []
    for j in range(len(google_p1)):
        if j< len(google_p1)-1:
            google_p2.append(google_p1[j]+google_p1[j+1])
        if j< len(google_p1)-2:
             google_p3.append(google_p1[j]+google_p1[j+1]+google_p1[j+2])
        if j< len(google_p1)-3:
            google_p4.append(google_p1[j]+google_p1[j+1]+google_p1[j+2]
+google_p1[j+3])
```

候选组的划分首先要进行的一元组的划分，这一步也有非常多需要注意的地方，其中我们需要把标点符号当成是一个一元组进行划分，所以在代码中，我对读取到的字符进行了区分，如果是“ ”或者是“\n”那么就到下一个元组的划分中，其余情况的化，如果是数字或者是单词的字母，我们把连续的一串当成是一个划分对象，其余的，当我们遇见不是字母和数字的时候，我们统一认成是标点符号进行划分，标点符号占一个元组。

词语划分结果如图所示：

二，三，四元组的划分就在一元组的划分中合并即可。

3.4 第四部分

计算 BLEU

代码如下：

```
    if len(google_set_p1) > 0:
        google_p1_score = len(google_set_p1 & ref_set_p1) / ref_num
        # print("google_p1_score = ",len(google_set_p1 & ref_set_p1))
```

```
        if google_p1_score > 0:
            google_p1_score = math.log(len(google_set_p1 & ref_set_p1)
/ ref_num,2)
        else:
            google_p1_score = 0
        if ref_num > 1:
            google_p2_score = len(google_set_p2 & ref_set_p2) /
(ref_num-1)
            if google_p2_score > 0:
                google_p2_score = math.log(len(google_set_p2 &
ref_set_p2) / (ref_num-1),2)
            else:
                google_p2_score = 0
        else:
            google_p2_score = google_p1_score
        if ref_num > 2:
            google_p3_score = len(google_set_p3 & ref_set_p3) /
(ref_num-2)
            if google_p3_score > 0:
                google_p3_score = math.log(len(google_set_p3 &
ref_set_p3) / (ref_num-2),2)
            else:
                google_p3_score = 0
        else:
            google_p3_score = google_p1_score
        if ref_num > 3:
            google_p4_score = len(google_set_p4 & ref_set_p4) /
(ref_num-3)
            if google_p4_score > 0:
                google_p4_score = math.log(len(google_set_p4 &
ref_set_p4) / (ref_num-3),2)
            else:
                google_p4_score = 0
        else:
            google_p4_score = google_p1_score
        #
print(google_p1_score,google_p2_score,google_p3_score,google_p4_score)
```

对于标准数据以及翻译得到的对应的句子的 1，2，3，4 元组进行重合度的比较，同时进行 BLEU 的计算：

$$BLEU = BP * \exp\left(\sum_{n=1}^{N} wn * \log(pn)\right)$$

其中 BP 为长度过短的句子的惩罚因子，在此例中，wn = 1/n，实际中是每一个不同长度的元组的权重值，pn 为重合度。

计算的结果如下图所示：

```
['Civil', 'rights', 'groups', 'issue', 'travel', 'warning', 'for']
['Civil', 'rights', 'groups', 'issued', 'travel', 'warnings', 'against', 'Missouri', '.']
['Civil', '-', 'rights', 'groups', 'have', 'issued', 'travel', 'warnings', 'against']
['Civil', 'rights', 'group', 'issues', 'travel', 'warning', 'for']
0.38603056430513105
0.386030564305131
0.7366826287887924
['Due', 'to', 'discriminatory', 'policies', 'and', 'racist', 'attacks', 'in', 'Missouri', ',', 'the', 'National
['Due', 'to', 'the', 'discriminatory', 'policies', 'and', 'racist', 'attacks', 'in', 'Missouri', ',', 'the', 'A
['Due', 'to', 'Missouri', "'", 's', 'discriminatory', 'policies', 'and', 'racist', 'attacks', ',', 'the', 'Amer
['The', 'National', 'Association', 'for', 'the', 'Advancement', 'of', 'Colored', 'People', 'has', 'put', 'out',
0.2392541523207821
0.17991482644459816
0.15455699193831238
['The', 'NAACP', 'Missouri', 'Travel', 'Advisory', ',', 'effective', 'August', '28', ',', '2017', ',', 'calls',
['The', 'NAACP', 'Missouri', 'Travel', 'Consultation', ',', 'which', 'came', 'into', 'effect', 'on', 'August',
['Effective', 'August', '28', ',', '2017', 'NAACP', 'Missouri', 'travel', 'consultation', ',', 'because', 'of',
['The', 'NAACP', 'Travel', 'Advisory', 'for', 'the', 'state', 'of', 'Missouri', ',', 'effective', 'through', 'A
0.05937284023793568
0.07574511470507032
0.02876114553625507
['The', 'NAACP', 'pointed', 'to', 'a', 'recently', 'passed', 'Missouri', 'law', 'that', 'makes', 'it', 'harder'
...
['At', 'present', ',', 'the', 'airport', 'in', 'Bali', 'is', 'still', 'temporarily', 'open', '.']
0.19653956110926118
0.19653956110926118
0.23322141549146388
```

再算完 400 个句子的权重值之后，我们将得到的结果做平均值。

4.结果分析

结果如下：

```
# 输出结果
print(sum(google_score)/len(google_score))
print(sum(sg_score)/len(sg_score))
print(sum(jscb_score)/len(jscb_score))
```

0.2787240329824863

0.2629586689606913

0.24164962511335086

可以看到，google 翻译的分数是最高的，搜狗翻译的分数第二，金山词霸第三。

看起来 transfromer 还是有比较好的效果的。