COMP-8730_Assignment-2 Spell Correction Using LM

Student Information

- Name: Jiajie Yang
- UWin Acc: yang4q
- Student ID: 110115897


```
!python --version
!pip install "pytrec-eval-terrier"
!pip install --user -U nltk
```

```
import pytrec_eval
import json
import nltk
from nltk.corpus import brown
nltk.download('brown')
import random
```

```
    [nltk_data] Downloading package brown to /root/nltk_data...
    [nltk_data]   Unzipping corpora/brown.zip.
```

N-gram Language Model with NLTK News Genre of Brown's Corpus

Prepare Data

```
sent_lst = []
for sent in brown.sents(categories=['news', 'adventure']):
    sent_lst.append([w.lower() for w in sent])
#print(sent_lst[1])
#sent_lst = brown.sents()
#brown.words(categories='news')
#print(len([s for s in sent_lst if len(s) > 0]))
```

Train Models

```
from nltk.lm.preprocessing import padded_everygram_pipeline
from nltk.lm import MLE

ngram_order_lst = [1,2,3,5,10]
lm_dict = {}
for order in ngram_order_lst:
    train_data, vocab_data = padded_everygram_pipeline(order, sent_lst)
    lm = MLE(order)
    lm.fit(train_data, vocab_data)
    lm_dict[order] = lm
```

Use Max Heap to Store Top-K Probable Tokens

[ ] ↳ *1 cell hidden*

Find Success at K on the Testing Corpus

```
path = '/data/APPLING1DAT.643'
file_ = open(path, 'r')
lines = file_.readlines()

qrel_dict = {}
run_dict = {}
# init qrel's and run's
for order in ngram_order_lst:
    qrel_dict[order] = {}
    run_dict[order] = {}

lst_k = [1, 5, 10]
max_k = lst_k[-1]

for l in lines:
    # create correct spell and its context list
    if '$' in l: continue
    l_comp = l.split("  ")
    word_inc = l_comp[0].lower()
    word_cor = l_comp[1].lower()
    token_lst = l_comp[2].split("\n")[0].lower().split(" ")
    end_idx = token_lst.index('*')
    ctx_w_lst = token_lst[:end_idx]

    # predict using trained models - lm_dict
    for order in ngram_order_lst:
        lm = lm_dict[order]
        cur_ctx_w_lst = []
        if not order == 1:
            cur_ctx_w_lst = ctx_w_lst[0 - order + 1:]
        pred_heap = MaxHeap_Max_Score_Word(max_k)
        pred_lst = []
        qrel_dict[order][word_inc] = {word_cor : 1}
        run_dict[order][word_inc] = {}
        # find top-k largest score words => top-k smallest minus scores
        for w in lm.vocab:
            if not (ord('a') <= ord(w[0]) and ord(w[0]) <= ord('z')): continue
            score = lm.logscore(w, cur_ctx_w_lst)
            minus_score_word_pair = (0 - score, w)
            if not pred_heap.is_full():
                pred_heap.insert(minus_score_word_pair)
            elif minus_score_word_pair[0] < pred_heap.topKey():
                pred_heap.extractMax()
                pred_heap.insert(minus_score_word_pair)
        for i in range(1, len(lst_k) + 1):
            cut_off = lst_k[len(lst_k) - i]
            key_val = 1 / cut_off
            interval = cut_off
            if not i == len(lst_k):
                interval -= lst_k[len(lst_k) - i - 1]
            for j in range(interval):
                predict = pred_heap.extractMax()[1]
                run_dict[order][word_inc][predict] = key_val
```

Apply evaluation measures for each ngram order

```
for ngram_oder in ngram_order_lst:
    evaluator = pytrec_eval.RelevanceEvaluator(qrel_dict[ngram_oder], {'success'})
    res = evaluator.evaluate(run_dict[ngram_oder])
```

```
print(ngram_oder, 'Gram Averages:')
for measure in list(res[list(res.keys())[0]].keys()):
    q = [query_measures[measure] for query_measures in res.values()]
    val = pytrec_eval.compute_aggregated_measure(measure, q)
    print("  ", measure, val)
```

```
 1 Gram Averages:
    success_1 0.0
    success_5 0.0
    success_10 0.0
 2 Gram Averages:
    success_1 0.005235602094240838
    success_5 0.010471204188481676
    success_10 0.02617801047120419
 3 Gram Averages:
    success_1 0.015706806282722512
    success_5 0.020942408376963352
    success_10 0.03664921465968586
 5 Gram Averages:
    success_1 0.015706806282722512
    success_5 0.020942408376963352
    success_10 0.03664921465968586
 10 Gram Averages:
    success_1 0.015706806282722512
    success_5 0.020942408376963352
    success_10 0.03664921465968586
```