SpellCorrectionWithKGrams

September 7, 2023

[184]:
```
import re
import os
from collections import defaultdict, Counter
from progressbar import progressbar
```

[185]:
```
DATA_PATH = "f:/Datasets/classics/"
```

[186]:
```
def tokenize_text(text: str, min_len: int=4, to_lower=True)->list:
    pattern = re.compile(r"[A-Za-z0-9]+[-|']{0,1}[A-Za-z0-9]+")
    if to_lower == True:
        return [w.lower() for w in re.findall(pattern, text) if len(w)>=min_len]
    else:
        return [w for w in re.findall(pattern, text) if len(w)>=min_len]
```

[187]:
```
def count_k_grams(token: str, kgram_len: int=2)->int:
    return len(token)-(kgram_len)
```

[188]:
```
def get_k_grams(token: str, kgram_len: int=2)->list:
    return [token[i:i+kgram_len] for i in range(len(token)-kgram_len+1)]
```

[190]:
```
get_k_grams("partuculars")
```

[190]:
```
['pa', 'ar', 'rt', 'tu', 'uc', 'cu', 'ul', 'la', 'ar', 'rs']
```

[191]:
```
vocabulary = defaultdict(dict)
inverted_index = defaultdict(list)
k_gram_index = defaultdict(list)
for doc_id, flname in enumerate(progressbar(os.listdir(DATA_PATH))):
    with open(os.path.join(DATA_PATH, flname), encoding="latin") as fp:
        text = fp.read()
        tokens = tokenize_text(text, min_len=4)
        for token in tokens:
            vocabulary[token]["n_kgrams"] = count_k_grams(token)
            if doc_id not in inverted_index[token]:
                inverted_index[token].append(doc_id)
            for k_gram in get_k_grams(token, 3):
                if token not in k_gram_index[k_gram]:
                    k_gram_index[k_gram].append(token)
```

```
100% (14 of 14) |########################| Elapsed Time: 0:00:22 Time:  0:00:22
```

[192]:
```
def intersect_postings(p1, p2):
    results = []
    i = 0
    j = 0
    while i<len(p1) and j<len(p2):
        if p1[i] == p2[j]:
            results.append(p1[i])
            i += 1
            j +=1
        elif p1[i] < p2[j]:
            i += 1
        else:
            j += 1

    return results
```

[193]:
```
intersect_postings(k_gram_index["par"], k_gram_index["tic"])
```

[193]:
```
[]
```

[211]:
```
query = "greede"
```

[212]:
```
def jaccard_coefficient(term1, term2):
    return count_overlap[term2]/
 ↪(count_k_grams(term1)+count_k_grams(term2)-count_overlap[term2])
```

[213]:
```
query_k_grams = get_k_grams(query, 3)
count_overlap = defaultdict(int)
jaccard_overlap = defaultdict(float)
for k_gram in query_k_grams:
    results = k_gram_index[k_gram]
    for term in results:
        if term[0] == query[0]:
            count_overlap[term] += 1
```

[214]:
```
sorted([(k, v) for k, v in count_overlap.items()], key=lambda x: x[1],␣
 ↪reverse=True)[:10]
```

[214]:
```
[('greedy', 3),
 ('greedily', 3),
 ('greediness', 3),
 ('greed', 3),
 ('green', 2),
 ('greeting', 2),
 ('greens', 2),
 ('greenish', 2),
```

```
 ('greek', 2),
 ('greeted', 2)]
```

[215]:
```
for term in count_overlap.keys():
    jaccard_overlap[term] = jaccard_coefficient(query, term)
```

[216]:
```
sorted([(k, v) for k, v in jaccard_overlap.items()], key=lambda x: x[1],␣
 ↪reverse=True)[:10]
```

[216]:
```
[('greed', 0.75),
 ('greedy', 0.6),
 ('greedily', 0.42857142857142855),
 ('green', 0.4),
 ('greek', 0.4),
 ('greet', 0.4),
 ('greens', 0.3333333333333333),
 ('greediness', 0.3333333333333333),
 ('greece', 0.3333333333333333),
 ('greeks', 0.3333333333333333)]
```

[ ]: