Методы сглаживания

Лебедев Андрей, группа 210

Подключения:

1. Модуль math для вычисления логарифма
2. Библиотека nltk для работы с униграммами и биграммами
3. Модуль collections создания и работы с классом defaultdict() с функцией по умолчанию int
4. Анализатор pymorphy2 для лемматизации слов

Создаем список токенов обучающего текста, очищенного от знаков препинания:

```
tokens = clear_text('poem.txt')
```

Файл poem.txt содержит обучающий текст.

Функция clear_text:

```
def clear_text(name):
   # создаем список токенов, очищая текст от знаков препинания с помощью replace и проводя лемматизацию слов с помощью pymorphy
   f = open(name, 'r')
   tokens = []
   for s in f:
       s = s.replace(',', '').replace('\n', ' ').replace('-', ' ').replace('.', '')
       s = s.split()
       for i in s:
           tokens.append(morph.parse(i)[0].normal_form)
   f.close()
   return tokens
```

Используем словарь с функцией int по умолчанию для хранения частоты вхождения в текст каждой униграммы и биграммы:

```
c_unigrams = defaultdict(int)
for unigram in tokens:
   c_unigrams[unigram] += 1

bigr = list(nltk.bigrams(tokens))
c_bigrams = defaultdict(int)
for bigram in bigr:
   c_bigrams[bigram] += 1
```

Далее работаем с текстом для тестирования, используя функцию clear_text:

```
tokens = clear_text('test.txt')
```

Таким образом, корпус:
['вот', 'дом', 'который', 'построить', 'джек', 'а', 'это', 'пшеница', 'который', 'в', 'тёмный', 'чулан', 'храниться', 'в', 'дом', 'который', 'построить', 'джек', 'а', 'это', 'весёлый', 'птица', 'синица', 'который', 'часто', 'воровать', 'пшеница', 'который', 'в', 'тёмный', 'чулан', 'храниться', 'в', 'дом', 'который', 'построить', 'джек']

Тестирующее множество:
['вот', 'кот', 'который', 'пугать', 'и', 'ловить', 'синица', 'который', 'часто', 'воровать', 'пшеница', 'который', 'в', 'тёмный', 'чулан', 'храниться', 'в', 'дом', 'который', 'построить', 'джек']

Сглаживание Лапласа

Считаем вероятность каждой униграммы и биграммы со сглаживанием Лапласа:

```
unigrams_probability_LP = defaultdict(int)
for unigram in tokens:
   unigrams_probability_LP[unigram] = (c_unigrams.get(unigram, 0) + 1) / (N + V)

bigr = list(nltk.bigrams(tokens))
bigrams_probability_LP = defaultdict(int)
for bigram in bigr:
   bigrams_probability_LP[bigram] = (c_bigrams.get(bigram, 0) + 1) /
(c_unigrams[bigram[0]] + len(c_bigrams) ** 2)
```

Где tokens - тестируемый текст, N - число слов в обучающем множестве, V - число уникальных слов.

Вероятность наиболее популярных униграмм:

который	0.12963
в	0.09259
дом	0.07407
построить	0.07407
джек	0.07407
пшеница	0.05556

Вероятность наиболее популярных биграмм:

дом | который	0.00901
построить | джек	0.00901
который | построить	0.00895
пшеница | который	0.00677
тёмный | чулан	0.00677
чулан | храниться	0.00677
храниться | в	0.00677
в | тёмный	0.00674

Считаем вероятность тестируемого множества по униграммам и биграммам:

```
P_unigrams = 1
for w, cnt in unigrams_probability_LP.items():
   P_unigrams *= cnt
P_bigrams = 1
for w, cnt in bigrams_probability_LP.items():
   P_bigrams *= cnt
```

Результаты:
По униграммам: $1.028393 * 10^{-23}$
По биграммам: $2.584487 * 10^{-47}$

Перплексия со сглаживанием Лагранжа:
По униграммам: 18.647743
По биграммам: 213.491841

Функция для вычисления перплексии:

```
def perplexy(lst, dict):
   summ = 0
   N = len(lst)
   for w in lst:
       summ += math.log2(dict[w])
   summ *= (- 1 / N)
   return 2 ** summ
```

Сглаживание Линдстоуна

Подберем коэффициент λ перебором с шагом 0.05:

```
unigrams_probability_Lid = defaultdict(int)
bigrams_probability_Lid = defaultdict(int)
for i in range(1, 21):
   k = 0.05 * i
   for unigram in tokens:
       unigrams_probability_Lid[unigram] = (c_unigrams.get(unigram, 0) + k) / (N +
V * k)
   for bigram in bigr:
       bigrams_probability_Lid[bigram] = (c_bigrams.get(bigram, 0) + k) /
(c_unigrams[bigram[0]] + len(c_bigrams) * k)
   PP_u = perplexy(tokens, unigrams_probability_Lid)
   PP_b = perplexy(bigr, bigrams_probability_Lid)
```

Вычислим перплексию:

λ	Униграммы	Биграммы
0.05	30.656251	4.660939
0.1	26.964117	5.266237
0.15	25.054602	5.836715
0.2	23.810027	6.359132
0.25	22.908237	6.83841
0.3	22.213394	7.280387
0.35	21.655993	7.690053
0.4	21.195919	8.071493
0.45	20.808025	8.428052
0.5	20.47556	8.762498
0.55	20.186836	9.077146
0.6	19.933391	9.373956
0.65	19.708916	9.654606
0.7	19.508589	9.920544
0.75	19.328647	10.17303
0.8	19.166102	10.413169

0.85	19.018545	10.641935
0.9	18.884006	10.860194
0.95	18.760858	11.068718
1.0	18.647743	11.268196

Выводы

Составили две языковые модели: используя сглаживание Лапласа и сглаживание Линдстоуна. Опираясь на значения перплексии, как на метрику для оценки языковых моделей, можем сделать вывод, что при вычислении вероятности текста с помощью униграмм сглаживание Лапласа не уступает сглаживанию Линдстоуна. Что касается биграмм, то можем заметить, что при сглаживании Линдстоуна перплексия возрастала с ростом λ, оставаясь значительно меньше перплексии при сглаживании Лапласа. Следовательно, лучший результат достигается при использовании сглаживания Линдстоуна и выбором минимального λ.