توضیحات پروژه دوم درس هوش مصنوعی

کد پروژه شامل چندین بخش می شود که در ادامه آورده شده است:

- Problem parameters: در این بخش، می آییم و با خواندن خط به خط فایل آموزشی، تیترها و متن های آن را جدا می کنیم و در لیست میریزیم.

```
#problem parameters
f = open("HAM-Train.txt",'r', encoding='utf-8')
subjects = []
subjects_counts = [] #number of texts from every subject
total_subjects_count = 0
subjects_probabilites = []
texts = []
uni_models = []
bi_models = []
for line in f:
    parts = line.split('@@@@@@@@@@')
    subject = parts[0].replace('\ufeff', '')
    text = parts[1]
    total_subjects_count = total_subjects_count + 1
    if not subject in subjects:
        subjects.append(subject)
        subjects_counts.append(1)
        texts.append(text)
    else:
        texts[subjects.index(subject)] += text
        subjects_counts[subjects.index(subject)] = subjects_counts[subjects.index(subject)] + 1
```

- Subjects probabilities: در این بخش، می آییم و احتمال های هر کدام از تیترها(p(I)) را با توجه به تعداد دفعات تکرار در فایل آموزشی، محاسبه می کنیم:

```
#subjects probabilites
for subject_count in subjects_counts:
    subjects_probabilites.append(math.log(subject_count / total_subjects_count, math.e))

print('subjects are', subjects)
print('subject counts are', subjects_counts)
print('total subjects count is', total_subjects_count)
print('subject probabilites are', subjects_probabilites)
```

- Unigram: در اینجا، مدل unigram را برای هر کدام از تیترها به دست می آوریم(از لگاریتم احتمال استفاده شده است):

```
#Unigram
for subject in subjects:
    uni_model = {}
    count = 0
    for word in texts[subjects.index(subject)].split(' '):
        count = count + 1
        if word in uni_model:
            uni_model[word] = uni_model[word] + 1
        else:
            uni_model[word] = 1
    for word in uni_model:
        uni_model[word] /= count
        uni_model[word] = math.log(uni_model[word], math.e) # using ln as a substitution
    uni_models.append(uni_model)
```

- Bigram: مشابه unigram، اینجا نیز مدل bigram را برای هر کدام از تیترها به دست می آوریم:

```
#Bigram
for subject in subjects:
    bi_uni_model = {}
    bi_model = {}
    words = []
    bi_words = []
    count = 0
    for word in texts[subjects.index(subject)].split(' '):
        count = count + 1
        words.append(word)
        if len(words) != 1 :
            bi_word = words[-2] + ' ' + words[-1]
            bi_words.append(bi_word)
            if bi_word in bi_model:
                bi_model[bi_word] = bi_model[bi_word] + 1
            else:
                bi_model[bi_word] = 1
        if word in bi_uni_model:
            bi_uni_model[word] = bi_uni_model[word] + 1
        else:
            bi_uni_model[word] = 1
    for bi_word in bi_model:
        bi_model[bi_word] /= bi_uni_model[bi_word.split()[0]]
        bi_model[bi_word] = math.log(bi_model[bi_word], math.e) # using ln as a substitution
    bi_models.append(bi_model)
```

- Test: در اینجا، ابتدا با توجه به فایل تست، تیترها و مشخصات آن را جدا می کنیم:

```
#test
f = open("HAM-Test.txt",'r', encoding='utf-8')
test_subjects = []
test_distinct_subjects = []
test_subjects_counts = [] #number of texts from every subject
total_test_subjects_count = 0
test_texts = []
results = []
results_subjects = []
for line in f:
    test_parts = line.split(          )
    test_subject = test_parts[0].replace('\ufeff', '')
    test_subjects.append(test_subject)
    if not test_subject in test_distinct_subjects:
        test_distinct_subjects.append(test_subject)
        test_subjects_counts.append(1)
    else:
        test_subjects_counts[test_distinct_subjects.index(test_subject)] = test_subjects_counts[test_distinct_subjects.inde
    test_texts.append(test_parts[1])
    total_test_subjects_count = total_test_subjects_count + 1

print('test distinct subjects are', test_distinct_subjects)
print('test subject counts are', test_subjects_counts)
print('total test subjects count is', total_test_subjects_count)
```

سپس، طبق مدل های unigram و bigram و با در نظر گرفتن backoff(که در ادامه توضیح داده شده است)، سعی می کنیم بهترین تیتر را پیدا کنیم:

```
for test_text in test_texts:
    words = test_text.split()
    words_2 = []
    bi_words = []
    #testing
    result = None
    result_subject = None
    for uni_model in uni_models:
        current_result = 0
        for word in words:
            words_2.append(word)
            if len(words_2) != 1:
                bi_word = words_2[-2] + ' ' + words_2[-1]
                if bi_word in bi_models[uni_models.index(uni_model)]:
                    if word in uni_model:
                        backoff_1 = 0.33 * (math.e ** uni_model[word]) + 0.66 * (math.e ** bi_models[uni_models.index(uni_
                        backoff_2 = 0.98 * (math.e ** uni_model[word]) + 0.01 * (math.e ** bi_models[uni_models.index(uni_
                        backoff_3 = 0.01 * (math.e ** uni_model[word]) + 0.98 * (math.e ** bi_models[uni_models.index(uni_
                        backoff = backoff_3
                    else:
                        backoff = (math.e ** bi_models[uni_models.index(uni_model)][bi_word]) + 0.01
                    current_result += math.log(backoff)
                else:
                    if word in uni_model:
                        backoff = (math.e **uni_model[word]) + 0.01
```

- بررسی صحت تیترها: در اینجا با iterate کردن روی تمام متن های فایل تست، بررسی می کنیم که چه تعداد از تیتر ها مطابق با تیتراصلی خود متن است:

```
true_count = 0
for i in range(len(test_subjects)):
    if results_subjects[i] == test_subjects[i]:#checks whether we predicted correct or not!
        true_count = true_count + 1

print('number of correct predictions are', true_count)
```

- ارزیابی سیستم بازیابی اطلاعات: نهایتا نیز، جدول بازیابی اطلاعات را به دست آورده و مقادیر precision، recall و f-measure برای هر کدام از تیترها به دست می آید و نمایش داده می شود:

```
#true-false table
#initializing 2d array
tfp = []
for i in range(len(test_distinct_subjects)):
    row = []
    for i in range(len(test_distinct_subjects)):
        row.append(0)
    tfp.append(row)

for k in range(len(results_subjects)):
    for i in range(len(test_distinct_subjects)):
        if test_subjects[k] == test_distinct_subjects[i]: #where the actual subject should be
            for j in range(len(test_distinct_subjects)):
                if results_subjects[k] == test_distinct_subjects[j]:
                    tfp[i][j] = tfp[i][j] + 1

print('true false table is', tfp)
precisions = []
recalls = []
f_measures = []
for i in range(len(test_distinct_subjects)):
    precision_denom = 0
    recall_denom = 0
    for j in range(len(test_distinct_subjects)):
        precision_denom += tfp[j][i]
        recall_denom += tfp[i][j]

    precision = tfp[i][i] / precision_denom
    recall = tfp[i][i] / recall_denom
    f_measure = (2 * precision * recall) / (precision + recall)
```

```
print('true false table is', tfp)
precisions = []
recalls = []
f_measures = []
for i in range(len(test_distinct_subjects)):
    precision_denom = 0
    recall_denom = 0
    for j in range(len(test_distinct_subjects)):
        precision_denom += tfp[j][i]
        recall_denom += tfp[i][j]

    precision = tfp[i][i] / precision_denom
    recall = tfp[i][i] / recall_denom
    f_measure = (2 * precision * recall) / (precision + recall)

    precisions.append(precision)
    recalls.append(recall)
    f_measures.append(f_measure)

print('classes are', test_distinct_subjects)
print('precisions are', precisions)
print('recalls are', recalls)
print('fmeasures are', f_measures)
```

نتیجه backoff: دو مورد نتیجه گیری شد:

- با تست هایی که بر روی ضرایب موجود در backoff انجام شد، اینطور به نظر می رسید که هر چه ضریب bigram در فرمول بیشتر باشد، تعداد تطابق تیترها بیشتر می شود.

- هر چه مقدار small_prob(احتمال کوچکی که در صورت صفر بودن unigram و bigram به آن کلمه نسبت داده می شود) کمتر باشد، بُرد تغییرات تطابق تیترها به ازای ضرایب مختلف، کمتر می باشد(یعنی مثلا تغییرات تعداد تطابق ها در small_prob = 0.00014 و backoff 3 موجود در کد، کمتر از small_prob = 0.01 می باشد)

همچنین توجه شود که برخی از این پارامترها و نتایجشان، در فایل test.txt موجود می باشد.