تمرین دوم درس پردازش زبان طبیعی

نام تهیه‌کننده: ملیکا نوبختیان

نسخه: ۱

۱- بخش تئوری

۱-۱- سوال اول

۱-۱-۱- Leave p-out cross validation

در این روش ابتدا p نمونه از دیتاست را به عنوان test set انتخاب می‌کنیم. بقیه n-p داده training set ما خواهند بود. مدل را روی داده‌های train آموزش می‌دهیم و validation را روی داده‌های تست انجام می‌دهیم. نتیجه مدل روی داده‌های validation را ذخیره می‌کنیم. مراحل گفته‌شده را حالا با یک مجموعه p دیگر انجام می‌دهیم و این کار تا امتحان کردن تمام مجموعه‌های p ممکن انجام می‌دهیم. برای محاسبه عملکرد کلی مدل از نتایجی که به دست آوردیم میانگین می‌گیریم.

۱-۱-۲- Leave one-out cross validation

این روش در واقع همان روش دوم است که در آن p = 1 است و عملیات تکرار در آن n بار انجام خواهد شد. این روش و روش قبل هر دو از نوع exhaustive cross validation هستند. این روش‌ها از نظر محاسباتی گران هستند ولی تعداد داده کمتری را نسبت به روش‌های دیگر هدر می‌دهند.

۱-۱-۳- Hold-out cross validation

این روش ساده‌ترین و رایج ترین نوع cross validation است. در این روش دیتاست رو به دو بخش training set و test set تقسیم می‌کنیم که معمولا نسبت آن‌ها ۸۰ درصد به ۲۰ درصد است. مدل را روی training set آموزش می‌دهیم و validation آن را روی test set انجام می‌دهیم و نتایج مدل روی validation را ذخیره می‌کنیم. این روش معمولا روی دیتاست‌های بزرگ انجام می‌شود که تنها یک بار نیاز به train مدل داریم. معایب این روش این است که مجموعه test set ممکن است نماینده خوبی نباشد و تفاوت بسیاری با training داشته باشد و هم چنین این نکته که تنها یک بار انجام می‌شود دقیق بودن آن را کم می‌کند.

۱-۱-۴- K-fold cross validation

این روش در تلاش است تا معایب hold-out cross validation را به حداقل برساند و برای این کار یک روش جدید برای split کردن دیتاست معرفی می‌کند تا bottleneck تنها یک بار تست کردن را حل کند.
در این روش ابتدا یک عدد به نام k انتخاب می‌کنیم که تعداد fold ها را نشان خواهد داد. سپس دیتاست را به k بخش مساوی تقسیم می‌کنیم. تعداد k-1 fold را به عنوان training set و fold باقیمانده را به عنوان test set انتخاب می‌کنیم. مدل را روی training set آموزش می‌دهیم و validation را روی test set انجام می‌دهیم و نتیجه validation را ذخیره می‌کنیم. این عمل را k بار انجام می‌دهیم تا هر بار یک بخش به عنوان test set انتخاب شود.

در نهایت از تمام نتیجه‌ها میانگین می‌گیریم. این روش معایب کمتری نسبت به hold-out دارد ولی اگر تعداد k زیاد بشود هزینه زیادی خواهد داشت.

۱-۱-۵- Repeated random subsampling validation

این روش به monte-carlo cross validation هم شناخته می‌شود که دیتاست را به طور رندوم به training set و test set تقسیم می‌کند. بر خلاف k-fold که این تقسیم به چند گروه یا بخش انجام می‌شود ولی اینجا این کار به صورت رندوم انجام می‌شود. در نهایت نتیجه کلی ببین تمام split ها میانگین گرفته می‌شود.
در این روش ممکن است صرفا بعضی از داده‌ها به train یا test محدود شوند و در مجموعه دیگر حضور پیدا نکنند.

۱-۱-۶- Startified k-fold cross validation

این روش هنگامی یک دیتاست imbalanced از کلاس‌های متفاوت داریم به کمک ما می‌آید. این روش مانند همان k-fold است با این تفاوت که در هر fold باید نسبت کلاس‌های متفاوت به یکدیگر مشابه با مجموعه کلی باشد. در نهایت هم نتیجه‌ها میانگین گرفته خواهند شد.

۱-۲- سوال دوم

الف)

فرکانس تمام unigram های موجود:

<s>	A	B	</s>
3	10	11	3

تمام Bigram های قابل ساخت با توجه به جملات:

(<s>, A), (<s>, B), (A , A), (A , B), (B , B) , (B , A), (A, </s>) , (B, </s>)

فرکانس bigram ها:

	<s>	A	B	</s>
<s>	۰	۰	۳	۰
A	۰	۲	۶	۲
B	۰	۸	۲	۱
</s>	۰	۰	۰	۰

ب)

$$|V| = 4$$

P(w2=A|w1=B)=$\frac{c(B,A)}{c(B)} = \frac{8}{11} = 0.72$

Psmooth(w2=A|w1=B)= $\frac{c(B,A)+1}{c(B)+|V|} = \frac{8+1}{11+4} = 0.6$

P(w2=B|w1=<s>)=$\frac{c(<s>,B)}{c(<s>)} = \frac{3}{3} = 1$

Psmooth(w2=B|w1=<s>)=$\frac{c(<s>,B)+1}{c(<s>)+|V|} = \frac{3+1}{3+4} = 0.57$

P(w2=</s>|w1=A)= $\frac{2}{10} = 0.2$

Psmooth(w2=</s>|w1=A)= $\frac{2+1}{10+4} = 0.21$

۳-۱- سوال سوم

Sub:

③ احتمال رخداد تمام کلاس ها یکسان است پس خواهیم داشت:

$P(7) = P(8) = \frac{1}{10}$

$P(d=7 \mid c=7) = 0.5 \quad P(d=8 \mid c=8) = 0.7$

$P(d=8 \mid c=7) = 0.5 \quad P(d=7 \mid c=8) = 0.3$

برای اینکه احتمال شخص درست یک بود را باید احتمال اول سر و کار خواهیم داشت به عبارت دیگر خواهیم داشت

$P(\text{شخص درست}) = P(d=7, c=7) + P(d=8, c=8)$

$= P(7)\,P(d=7 \mid c=7) + P(8)\,P(d=8 \mid c=8)$

$= \frac{1}{10} \times 0.5 + \frac{1}{10} \times 0.7 = 0.12$

۲- بخش عملی

پس از دانلود دیتاست imdb، word_index آن را که کلمات را به ایندکس‌های آن‌ها مرتبط می‌کند نیز دانلود کردم. از این word_index و عکس آن استفاده کردم تا کلمات موجود را شناسایی کنم. در این word_index تنها کلمات پرتکرارتر و مهم‌تر وجود دارند. پس یک index را نیز برای کلماتی که در آن وجود ندارند در نظر گرفتم و آن را همان unknown در نظر گرفتم و مجموعه train و test را به این شکل تغییر دادم:

```
indexed_tokens_count = len(inverted_word_index.keys())
## We will use "indexed_tokens_count + 1" as an index for unknown words
inverted_word_index[indexed_tokens_count + 1] = '<UNK>'
```

```
## with adding <UNK> we should clean data and substitute unknown tokens with index <UNK>
def clean_unknowns(samples, idx_word_dict):
  known_idx = idx_word_dict.keys()
  for i, sample in enumerate(samples):
    for j, tok in enumerate(sample):
      if not tok in known_idx:
        samples[i][j] = indexed_tokens_count + 1
```

```
## Clean train data
clean_unknowns(x_train, inverted_word_index)
## Clean Test data
clean_unknowns(x_test, inverted_word_index)
```

در مرحله بعد به سراغ tokenization رفتم و با استفاده از دیکشنری موجود و index ها متن اصلی نمونه‌های train و test را به دست آوردم:

```
def text_tokenization(samples, inverted_word_index):
  tokenized_samples = [None] * len(samples)
  for i, sample in enumerate(samples):
    tokenized_samples[i] = []
    for tok in sample:
      tokenized_samples[i].append(inverted_word_index[tok])

  return tokenized_samples
```

```
tokenized_train = text_tokenization(x_train, inverted_word_index)
tokenized_test = text_tokenization(x_test, inverted_word_index)
```

در مرحله بعد به سراغ stemming رفتم و برای این کار از Snowball استفاده کردم:

```
def text_stemmer(samples):
  stemmer = SnowballStemmer("english")
  stemmed_samples = [None] * len(samples)
  for i, sample in enumerate(samples):
    stemmed_samples[i] = []
    for tok in sample:
      stemmed_samples[i].append(stemmer.stem(tok))

  return stemmed_samples
```

برای مدل‌های unigram, bigram و trigram من یک مدل کلی در نظر گرفتم که با ورودی دادن یک پارامتر به نام n_gram مشخص می‌کنیم که از کدام یک از این سه دسته است. داده‌های ورودی و کلاس‌های آن‌ها و هم چنین تعداد کلاس‌ها موارد دیگری هستند که برای ساختن یک نمونه از این کلاس باید داشته باشیم. پس از داشتن این موارد باید مشخص کنیم که کدام یک از نمونه‌ها به کدام یک از کلاس‌ها تعلق دارند. سپس باید با توجه به n_gram ای که به مدل داده‌ایم تمام آن‌ها را به دست آوریم. سپس n_gram هایی که در هر کلاس هستند را به دست می‌آوریم. باید احتمال هر کلاس را نیز داشته باشیم. حالا نوبت به شمارش vocabulary می‌رسد و هم چنین اینکه هر کلاس شامل چه مقدار از آن‌ها است:

```
class Ngram_Model:

  def __init__(self, x, y, class_count, n_gram):
    self.n_gram = n_gram
    ## we give each class an index
    self.classes = [i for i in range(class_count)]
    ## split samples based on the class they belong to it
    self.class_samples_idx = [np.where(y == c)[0] for c in self.classes]
    ## find n_grams
    x_n_grams = self.find_n_grams(x)
    ## find n_grams that belong to each class
    self.x_grams = [[x_n_grams[idx] for idx in list(c)] for c in self.class_samples_idx]
    ## find probablity of each class
    self.class_prob = [len(c) / len(x_n_grams) for c in self.class_samples_idx]
    self.class_text = [[l for idx in list(c) for l in x_n_grams[idx]]for c in self.class_samples_idx]
    ## find vocabs of classes and freq
    self.vocab = [set(t) for t in self.class_text]
    self.vocab = {t for v in self.vocab for t in v}
    self.vocab_count = len(self.vocab)
    self.class_all_words_count = [len(c) for c in self.class_text]
    self.class_word_count = []
    for c in self.classes:
      self.class_word_count.append(dict())
      counter = Counter(self.class_text[c])
      for w in self.vocab:
        self.class_word_count[c][w] = counter[w]
```

```
def find_n_grams(self, data):

  ## Unigram
  if self.n_gram == 1:
    return data

  ## Bigram
  elif self.n_gram == 2:
    new_data = []
    for l in data:
      new_data.append(list(nltk.bigrams(["<s>"] + l + ["</s>"])))
    return new_data

  ## Trigram
  elif self.n_gram == 3:
    new_data = []
    for l in data:
      new_data.append(list(ngrams(["<s>"] + l + ["</s>"], 3)))
    return new_data
```

در نهایت برای پیش‌بینی test set متد predict را داریم. در اینجا ابتدا n_gram های آن مجموعه را محاسبه می‌کنیم. حالا که برای هر نمونه n_gram هایش را محاسبه کرده‌ایم نوبت به محاسبه احتمال خود آن با توجه به کلاس‌های مختلف و انتخاب کلاس موردنظر با توجه به بیشترین احتمال خواهد بود:

```
#add-1 smmothing / unknown word
def compute_word_prob(self, word, c):
  word_count = 0
  if word in self.class_word_count[c]:
    word_count = self.class_word_count[c][word]
  prob = (word_count + 1) / (self.class_all_words_count[c] + self.vocab_count + 1)
  return prob

def find_probs(self, test_data):
  probs = []
  for c in self.classes:
    prob_log = math.log(self.class_prob[c])
    for word in test_data:
      word_prob = self.compute_word_prob(word, c)
      prob_log += math.log(word_prob)
    probs.append(prob_log)
  return probs


def predict(self, test_set):
  test_n_gram = self.find_n_grams(test_set)
  pred = []
  for test_data in test_n_gram:
    pred.append(np.argmax(self.find_probs(test_data)))
  return pred
```

در نهایت مدل‌ها را با معیارهایی که گفته شده ارزیابی می‌کنیم نتیجه برای مدل‌های مختلف به شکل زیر خواهد بود:

Unigram

```
Accuracy: 0.79256
Precision: 0.8371128318584071
Recall: 0.72648
f1-score: 0.7778824738735652
```

Bigram

```
Accuracy: 0.86716
Precision: 0.8984286830454032
Recall: 0.82792
f1-score: 0.8617344602189934
```

Trigram

```
Accuracy: 0.87816
Precision: 0.9157431838170624
Recall: 0.83296
f1-score: 0.8723921240050272
```

همان طور که واضح است نتیجه trigram بهتر از bigram و نتیجه bigram بهتر از unigram است که نشان می‌دهد هر چه بیشتر به در رابطه میان کلمات دقت کنیم نتیجه بهتری خواهیم داشت. در نتیجه اثر bag of words که در مدل unigram وجود دارد دقت پایین تری را به ما ارائه می‌دهد.

موارد زیر چند نمونه از نقدهایی هستند که دسته‌بندی شده‌اند و نتایج برای مدل‌های مختلف در آن‌ها متفاوت است. همان طور که مشاهده می‌کنید در اکثر موارد trigram نتیجه را به درستی پیش‌بینی کرده‌است و بعد از آن bigram و سپس unigram پیش‌بینی‌های بهتری داشته‌اند:

```
Text:
the watching boy couch as on interesting never aunt an like did as on real reception badly to shiny of purchased but that eyed average one in exploitati
True class: 1
Unigram class: 0
Bigram class: 0
Trigram model: 1

Text:
the of unger animation underproduced male it pressured in miracles' explanation feat male take no commodity damsel psyche risk this kill in exploitatior
True class: 0
Unigram class: 1
Bigram class: 1
Trigram model: 0

Text:
the or decisions how was plays as on was might best failure of here br only explosive was might for was pretty in character up to up friends years only
True class: 1
Unigram class: 0
Bigram class: 1
Trigram model: 1

Text:
the don't laughable in typical clearly comedies is appears is reckless murders br must to peter as by beautiful exercise know suitably effects that cana
True class: 1
Unigram class: 0
Bigram class: 1
Trigram model: 0

Text:
the bygone bites 95 brilliantly recognizing or historically this positively secondly it early be far but of black hate bang faith br auction but good sp
True class: 1
Unigram class: 1
Bigram class: 1
Trigram model: 0

Text:
the watching his woman such as you it 'tough her along are shame no if boy dvd have after your pretty this is fact you as it of survivor br never is fact you was recent except
True class: 1
Unigram class: 0
Bigram class: 0
Trigram model: 1
```