تمرین دوم NLP

یاسمین مدنی
۹۷۵۳۲۲۶۵

فهرست

سوال ۱

Cross validation

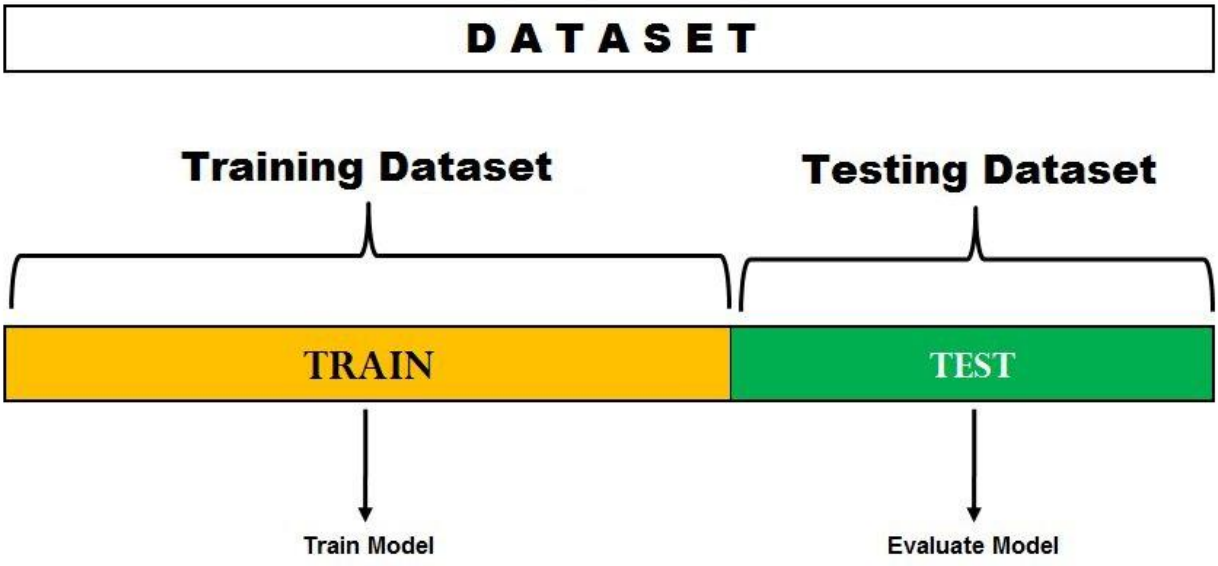
اعتبار سنجی متقاطع(cross validation) یک تکنیک نمونه گیری مجدد با ایده اساسی تقسیم مجموعه داده ها به ۲ بخش - داده های آموزشی و داده های آزمایشی است. داده‌های آموزش برای آموزش مدل و داده‌های آزمایش دیده نشده برای پیش‌بینی استفاده می‌شود. به عبارت دیگر تکنیکی برای ارزیابی نحوه تعمیم تحلیل آماری به یک مجموعه داده مستقل است. این تکنیک برای ارزیابی مدل های یادگیری ماشین با آموزش چندین مدل بر روی زیرمجموعه های داده های ورودی موجود و ارزیابی آنها بر روی زیر مجموعه داده های مکمل است. با استفاده از این روش، شانس زیادی وجود دارد که بتوانیم به راحتی over fittingرا تشخیص دهیم.

چندین تکنیک اعتبارسنجی متقاطع وجود دارد مانند

- **K-Fold Cross Validation**
- **Stratified k-fold cross-validation**
- **Leave P-out Cross Validation**
- **Leave One-out Cross Validation**
- **Repeated Random Sub-sampling Method**
- **Time Series cross-validation**
- **Nested cross-validation**
- **Holdout Method**

Holdout Method

یک روش اعتبار سنجی متقاطع جامع، که به طور تصادفی مجموعه داده را به داده های آموزش و آزمایش بسته به تجزیه و تحلیل داده ها تقسیم می کند. این ساده ترین روش ارزیابی است و به طور گسترده در پروژه های یادگیری ماشین استفاده می شود.بسته به مورد استفاده، داده ها را می توان به ۷۰-۳۰ یا ۶۰-۴۰، ۷۵-۲۵ یا ۸۰-۲۰ یا حتی ۵۰-۵۰ تقسیم کرد. به عنوان یک قاعده، نسبت داده های آموزشی باید بزرگتر از داده های آزمون باشد.



این روش برای زمانی که مجموعه داده بسیار بزرگی داریم، در تنگنای زمانی قرار داریم، یا در حال شروع به ساختن یک مدل اولیه در پروژه خود هستیم، مناسب است. از آنجایی که سایر روشهای اعتبار سنجی متقاطع از چندین تقسیم تست و آموزش استفاده می کند، اجرای آن به قدرت محاسباتی و زمان بیشتری نسبت به روش Holdout نیاز دارد.

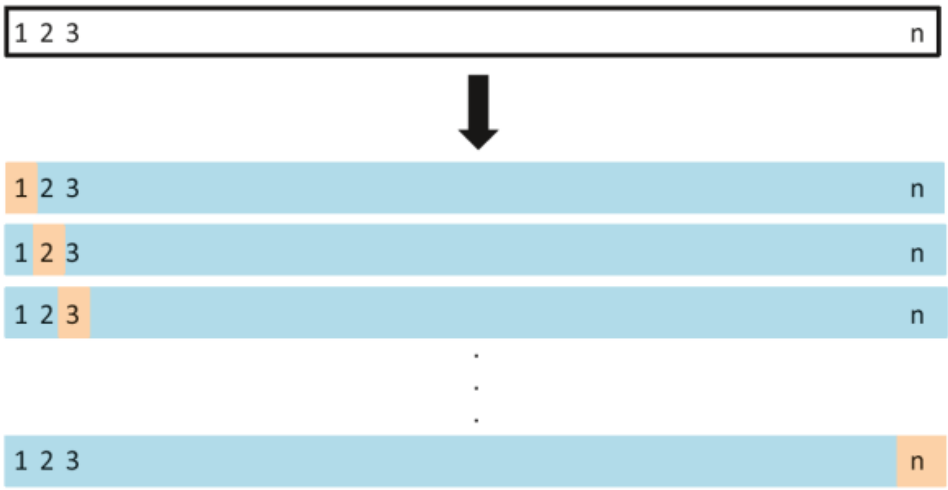
Leave P-out Cross Validation

(LpOCV)یک تکنیک اعتبار سنجی جامع است که شامل استفاده از p مشاهده به عنوان داده های اعتبار سنجی است و داده های باقی مانده برای آموزش مدل استفاده می شود.

Leave-one-out cross-validation

این روش یک دسته از LpOCV با p=1 است.

LOOCV یک نوع اعتبارسنجی متقاطع k-fold است که در آن k=n است

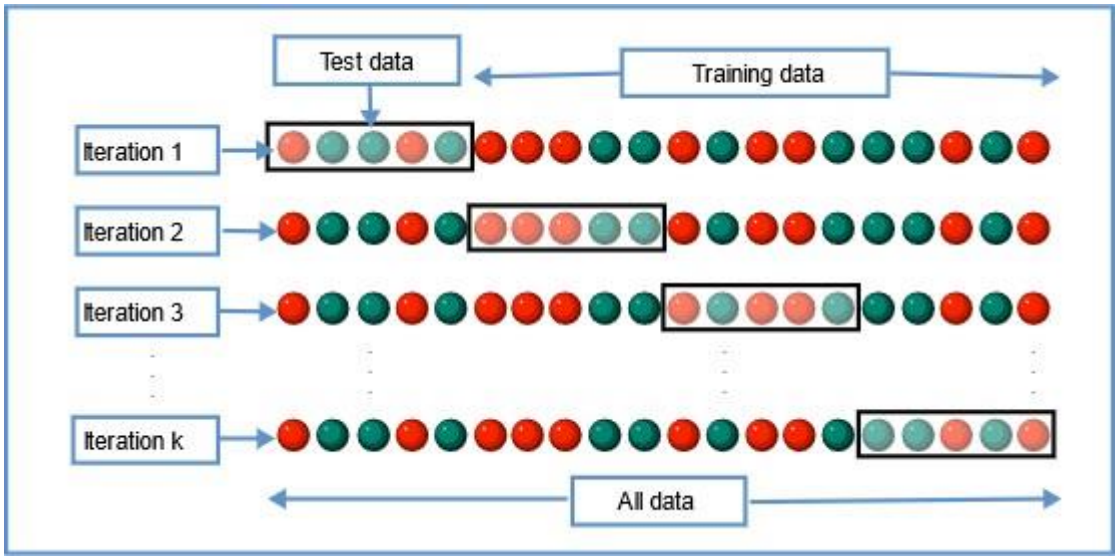


LOOCV برای تخمین عملکرد الگوریتم‌های یادگیری ماشین زمانی که برای پیش‌بینی داده‌هایی که برای آموزش مدل استفاده نمی‌شوند، استفاده می‌شود.برای مجموعه داده های کوچک یا زمانی که عملکرد مدل تخمین زده شده حیاتی است مناسب است.

انجام این روش از لحاظ محاسباتی پرهزینه است، اگرچه منجر به تخمین قابل اعتماد و بدون بایاس عملکرد مدل می‌شود. اگرچه استفاده از آن ساده است اما مواقعی وجود دارد که از این روش نباید استفاده شود، مانند زمانی که یک مجموعه داده بسیار بزرگ یا یک مدل محاسباتی گران برای ارزیابی داریم.

K-Fold Cross Validation

در این روش، مجموعه داده اصلی به طور مساوی به k زیربخش یا fold تقسیم می شود. از میان k-folds یا گروه ها، برای هر تکرار، یک گروه به عنوان داده های اعتبار سنجی و گروه های باقی مانده (k-1) به عنوان داده های آموزشی انتخاب می شوند. این فرآیند برای k بار تکرار می شود تا زمانی که هر گروه به عنوان اعتبار سنجی در نظر گرفته شود و به عنوان داده آموزشی باقی بماند.

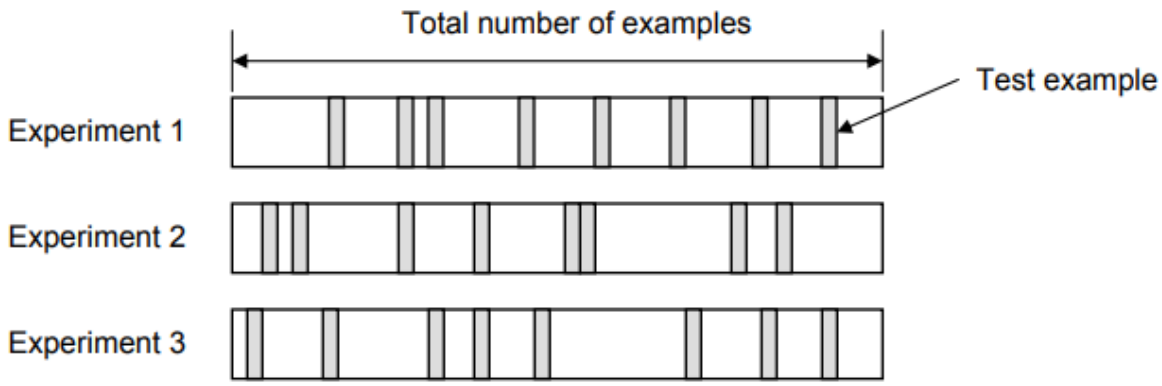


دقت نهایی مدل با در نظر گرفتن میانگین دقت داده های اعتبارسنجی k مدل محاسبه می شود.

از مزایای این روش میتوان به اینکه مدل دارای سوگیری کم است، پیچیدگی زمانی کمی دارد و کل مجموعه داده هم برای آموزش و هم برای اعتبار سنجی استفاده می شود.اما برای مجموعه داده نامتعادل مناسب نیست.

Repeated random subsampling validation

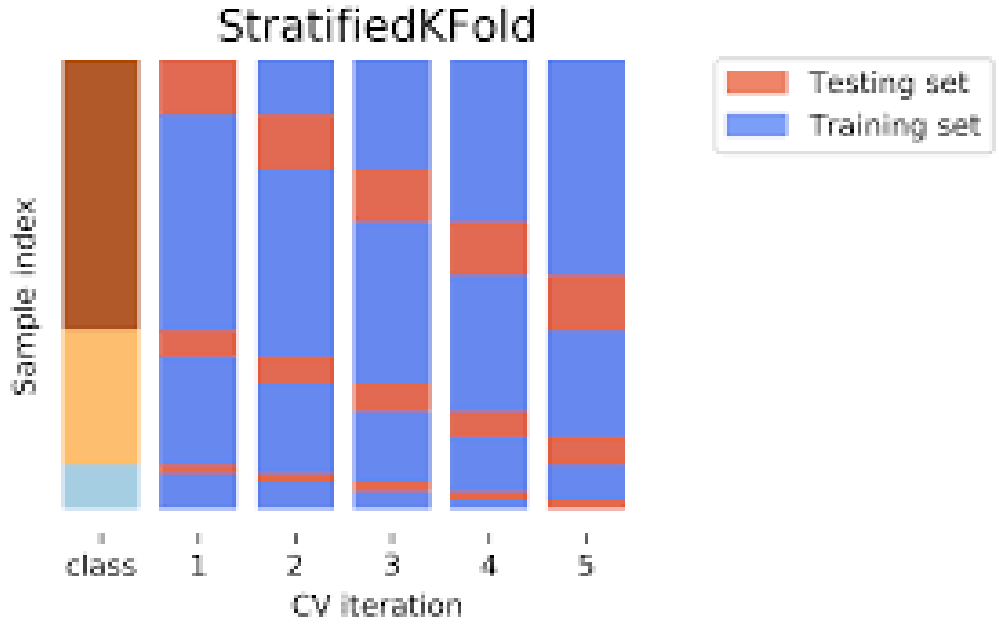
اعتبارسنجی زیرنمونه‌گیری تصادفی مکرر که به آن اعتبارسنجی متقابل مونت کارلو نیز گفته می‌شود، مجموعه داده را به‌طور تصادفی به آموزش و اعتبار سنجی تقسیم می‌کند. برخلاف k-fold تقسیم مجموعه داده به گروه ها یا چین ها نیست، اما به صورت تصادفی تقسیم می شود.



Stratified k-fold cross-validation

تکنیک های قبلی که توضیح داده شد،ممکن است با یک مجموعه داده نامتعادل به خوبی کار نکنند.Stratified k-fold مشکل مجموعه داده نامتعادل را حل کرد.

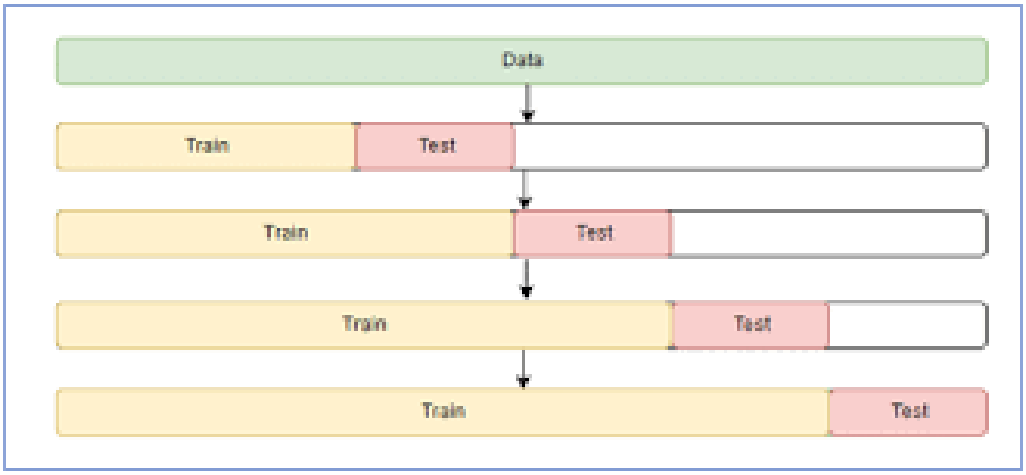
در این روش مجموعه داده به k گروه یا چین تقسیم می شود به طوری که داده های اعتبارسنجی دارای تعداد مساوی از نمونه های برچسب کلاس هدف هستند. به این صورت تضمین میشود که یک کلاس خاص در اعتبارسنجی یا داده‌های آموزش بیش از اندازه وجود ندارد، به‌خصوص زمانی که مجموعه داده نامتعادل است.



Time Series cross-validation

در این روش ترتیب داده ها برای مسائل مربوط به سری زمانی بسیار مهم است. برای مجموعه داده های مرتبط با زمان، تقسیم تصادفی یا k-fold داده ها به آموزش و اعتبارسنجی ممکن است نتایج خوبی به همراه نداشته باشد.

برای مجموعه‌داده سری زمانی، تقسیم داده‌ها به آموزش و اعتبارسنجی براساس زمان است که به آن روش forward chaining یا rolling cross-validation نیز گفته می‌شود.

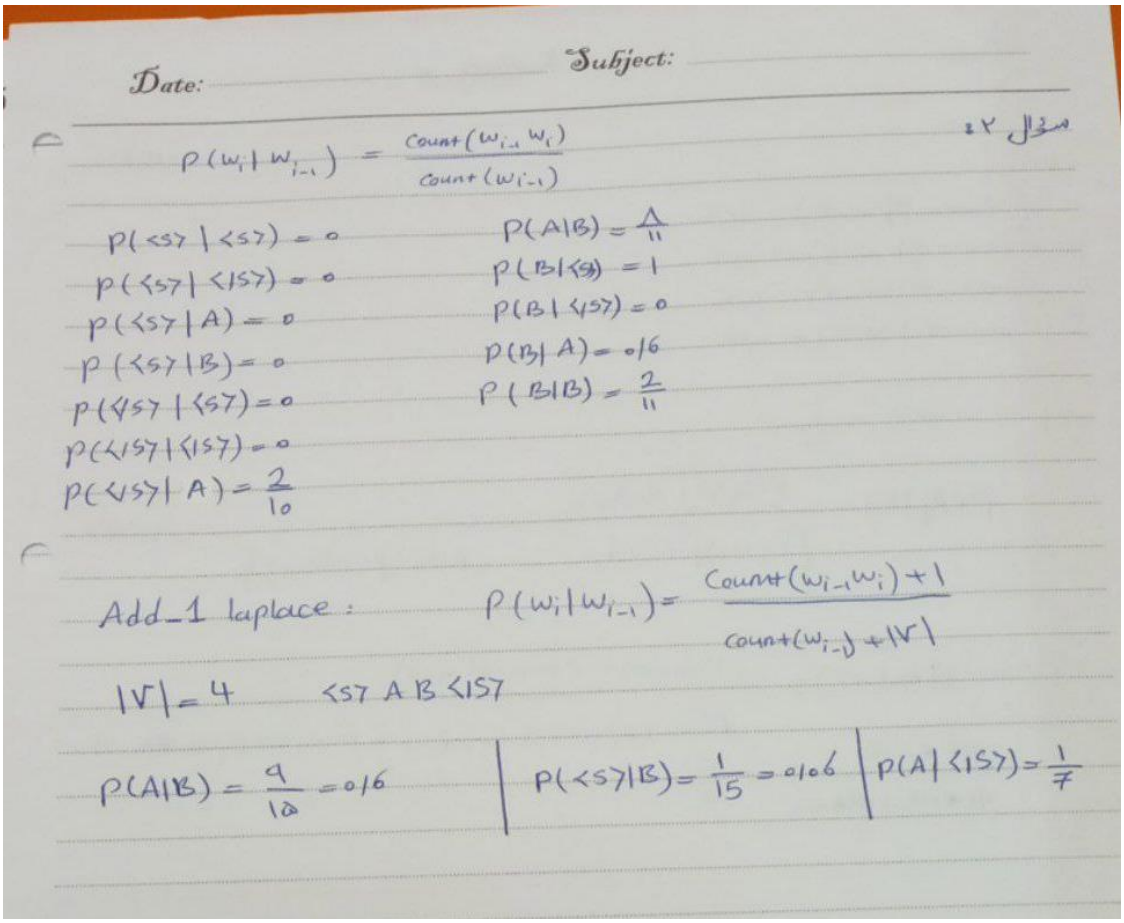


Nested cross-validation

در مورد اعتبارسنجی k-fold و Stratified k-fold ، ما تخمین ضعیفی از خطا در داده های آموزشی و آزمایشی دریافت می کنیم. تنظیم هایپرپارامتر در روش های قبلی به طور جداگانه انجام می شود. هنگامی که از اعتبارسنجی متقاطع به طور همزمان برای تنظیم فراپارامترها و تعمیم تخمین خطا استفاده می شود، اعتبارسنجی Nested cross-validationمورد نیاز است.

Nested Cross Validation می تواند در هر دو نوع k-fold و k-fold طبقه بندی شده قابل اجرا باشد.

سوال ۲



Date: Subject:

سوال ۲:

$$P(w_i \mid w_{i-1}) = \frac{Count(w_{i-1} w_i)}{Count(w_{i-1})}$$

$P(\langle s \rangle \mid \langle s \rangle) = 0$
$P(\langle s \rangle \mid \langle /s \rangle) = 0$
$P(\langle s \rangle \mid A) = 0$
$P(\langle s \rangle \mid B) = 0$
$P(\langle /s \rangle \mid \langle s \rangle) = 0$
$P(\langle /s \rangle \mid \langle /s \rangle) = 0$
$P(\langle /s \rangle \mid A) = \frac{2}{10}$

$P(A \mid B) = \frac{4}{11}$
$P(B \mid \langle s \rangle) = 1$
$P(B \mid \langle /s \rangle) = 0$
$P(B \mid A) = 0.16$
$P(B \mid B) = \frac{2}{11}$

Add_1 laplace : $P(w_i \mid w_{i-1}) = \frac{Count(w_{i-1} w_i) + 1}{Count(w_{i-1}) + |V|}$

$|V| = 4$ $\langle s \rangle$ A B $\langle /s \rangle$

$P(A \mid B) = \frac{9}{15} = 0.6$ | $P(\langle s \rangle \mid B) = \frac{1}{15} = 0.06$ | $P(A \mid \langle /s \rangle) = \frac{1}{7}$

سوال ۳

سوال ۳:

$P(out=7 \mid in=8) = 0/3$

$P(out=8 \mid in=8) = 0/7$

$P(out=8 \mid in=7) = 0/5$

$P(out=7 \mid in=7) = 0/5$

$P(in=k) = \frac{1}{10}$

قانون بیز:

$$P(A_k \mid B) = \frac{P(A_k)\,P(B \mid A_k)}{[P(A_1)P(B \mid A_1) + \dots + P(A_n)P(B \mid A_n)]}$$

$$P(in=7 \mid out=7) = \frac{P(in=7) * P(out=7 \mid in=7)}{P(in=0) * P(out=7 \mid in=0) + \dots + P(in=9) * P(out=7 \mid in=9)}$$

$$= \frac{0/1 * 0/5}{0/1 * 0/5 + 0/1 * 0/3} = \frac{5}{8}$$

$$P(in=8 \mid out=8) = \frac{P(in=8) * P(out=8 \mid in=8)}{P(in=0) * P(out=8 \mid in=0) + \dots + P(in=9) * P(out=8 \mid in=9)}$$

$$= \frac{0/1 * 0/7}{0/1 * 0/5 + 0/1 * 0/7} = \frac{7}{12}$$

Kian

سوال عملی

با لود کردن دیتا به روش زیر کراس فرض می کند اندیس ها از ۱ شروع شده و عملیات UKN را نیز در این مورد پیاده سازی می کند .

```
from keras.datasets import imdb
INDEX_FROM=3
(x_train, y_train), (x_test, y_test) = imdb.load_data(index_from=INDEX_FROM)
```

سپس با لود کردن استاپ وردهای انگلیسی در یک آرایه برای استفاده از آنها زمینه را فراهم می کنیم.

```
import nltk
nltk.download('stopwords')
from nltk.corpus import stopwords
stop_words=stopwords.words('english')
```

در بخش پیش پردازش علاوه برآن که آفست مناسب را برای به دست آوردن کلمات مناسب در هر کامنت تعین می کنیم هر توکن بازگردانی شده را از لحاظ علامت نگارشی نبودن،عدد نبودن و استاپ ورد نبودن برسی و در صورت برقرار نبودن شرط آنها را حذف می کنیم.

تگ های br که از دیتای کراس پاک نشده بودن و به صورت خودکار هم unkدر نظر گرفته نمیشوند را نیز از توکن ها حذف کرده ایم.

```
import string
train_sents=[]
test_sents=[]
word_to_id = imdb.get_word_index()
word_to_id = {k:(v+INDEX_FROM) for k,v in word_to_id.items()}
word_to_id["<PAD>"] = 0
word_to_id["<START>"] = 1
word_to_id["<UNK>"] = 2
word_to_id["<UNUSED>"] = 3
id_to_word = {value:key for key,value in word_to_id.items()}
for j in range(0,len(x_train)):
  decoded =  [ id_to_word[id] for id in x_train[j] if id_to_word[id] != "br" and  not id_to_word[id] in stop_words and not (id_to_word[id].isdigit()) and not id_to_word[id] in stri
  train_sents.append(decoded)

for j in range(0,len(x_test)):
  decoded = [id_to_word[id] for id in x_test[j] if id_to_word[id] != "br" and  not id_to_word[id] in stop_words  and not (id_to_word[id].isdigit()) and not id_to_word[id] in string
  test_sents.append(decoded)
```

قدم بعدی یافتن ریشه کلمات و ساده سازی آنها یا همان stemmingاست البته که روش lemmatizing مناسب تر عمل می کند و دقت بالاتری دارد اما به دلیل نیاز داشتن به pos taggingزمان بر بوده که در این تمرین از آن صرف نظر شده است و اینجا از استمر snowball استفاده کرده ایم.

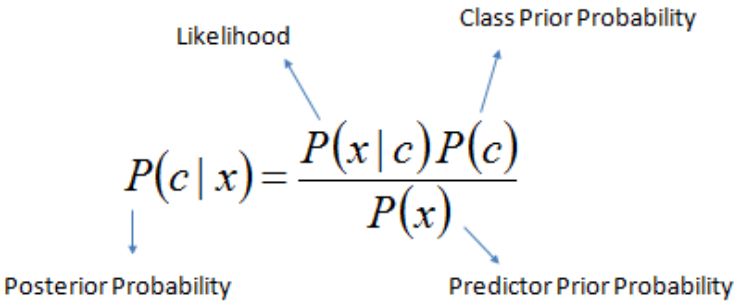
```
import nltk
from nltk.stem.snowball import SnowballStemmer
snow_stemmer = SnowballStemmer(language='english')
train_sttemed_sents=[]
test_sttemed_sents=[]
for sent in train_sents:
    tmp = [snow_stemmer.stem(w) for w in sent]
    train_sttemed_sents.append(tmp)

for sent in test_sents:
    tmp = [snow_stemmer.stem(w) for w in sent]
    test_sttemed_sents.append(tmp)
```

در مرحله ساخت ngramها ابتدا تعداد توکن های هر کلاس را محاسبه می کنیم و پس از آن با iterationروی داده های ترین تعداد تکرار یا همان فریکوعنسی کلمات را محاسبه می کنیم و از آنجا که باید لاپلاس اسموتینک و کلمات ناشناخته را در محاسبه دخیل کنیم از فرمول مناسب آن ها که در تصویر زیر آورده شده استفاده می کنیم.

```
unigram_prop['neg'] = {k : (v + 1) / (count_neg + len(unigram['neg']) + 1) for k, v in unigram['neg'].items()}
unigram_prop['neg'][-1] = 1 / (count_neg + len(unigram['neg']) + 1)
unigram_prop['pos'] = {k : (v + 1) / (count_pos + len(unigram['pos']) + 1) for k, v in unigram['pos'].items()}
unigram_prop['pos'][-1] = 1 / (count_pos + len(unigram['pos']) + 1)
```

سایر مدل ها به طریق مشابه به دست می آیند.

برای پیاده سازی نایبو بیز می دانیم فرمول آن از رابطه زیر به دست می آیدکه احتمال شرطی پیش از این محاسبه شده نیازی به دانست مخرج نیست و تنها با به دست آوردن ترم دیگر می توان به جواب رسید.



$$P(c \mid \mathrm{X}) = P(x_1 \mid c) \times P(x_2 \mid c) \times \cdots \times P(x_n \mid c) \times P(c)$$

در تابع NB که همان پیاده سازی نایبو بیز است این احتمال را برای هر دو کلاس + و – محاسبه و پس از آن پاسخ ها را تخمین میزنیم.

```python
#unigram
preds_uni = []
for sample in test_sttemed_sents:
        neg_pred, pos_pred = NB(1, unigram_prop, sample)
        if neg_pred > pos_pred:
            preds_uni.append(0)
        else:
            preds_uni.append(1)
y_pred_unigram=np.array(preds_uni)

#bigram
preds_bi = []
for sample in test_sttemed_sents:
        neg_pred, pos_pred = NB(2, bigram_prop, sample)
        if neg_pred > pos_pred:
            preds_bi.append(0)
        else:
            preds_bi.append(1)
y_pred_bigram=np.array(preds_bi)
#trigram
preds_tri = []
for sample in test_sttemed_sents:
        neg_pred, pos_pred = NB(3, trigram_prop, sample)
        if neg_pred > pos_pred:
            preds_tri.append(0)
        else:
```

با استفاده از توابع آماده کتابخانه sklearn معیار هی خواسته شده سوال را نیز در مورد تخمین هایمان چاپ کرده که کد و نتایج در زیر قابل مشاهده است.

```
print('Precision For unigram:', "{:.3f}".format(precision_score(y_test, y_pred_unigram)))
print('Precision For bigram:', "{:.3f}".format(precision_score(y_test, y_pred_bigram)))
print('Precision For trigram', "{:.3f}".format(precision_score(y_test, y_pred_trigram)))
print("////////////////////////////")
print('Recall For unigram:', "{:.3f}".format(recall_score(y_test, y_pred_unigram)))
print('Recall For bigram:',"{:.3f}".format(recall_score(y_test, y_pred_bigram)))
print('Recall For trigram', "{:.3f}".format(recall_score(y_test, y_pred_trigram)))
print("////////////////////////////")
print('F1 Score For unigram:', "{:.3f}".format(f1_score(y_ Loading... ed_unigram)))
print('F1 Score For bigram:',"{:.3f}".format(f1_score(y_test, y_pred_bigram)))
print('F1 Score For trigram', "{:.3f}".format(f1_score(y_test, y_pred_trigram)))
print("////////////////////////////")
print('Accuracy For unigram:', "{:.3f}".format(accuracy_score(y_test, y_pred_unigram)))
print('Accuracy For bigram:',"{:.3f}".format(accuracy_score(y_test, y_pred_bigram)))
print('Accuracy For trigram', "{:.3f}".format(accuracy_score(y_test, y_pred_trigram)))
```

Precision For unigram: 0.753

Precision For bigram: 0.873

Precision For trigram 0.792

////////////////////////////

Recall For unigram: 0.922

Recall For bigram: 0.831

Recall For trigram 0.721

////////////////////////////

F1 Score For unigram: 0.829

F1 Score For bigram: 0.852

F1 Score For trigram 0.755

////////////////////////////

Accuracy For unigram: 0.810

Accuracy For bigram: 0.855

Accuracy For trigram 0.766

در پایان با سه نمونه مختلف می بینیم که در هر کدام پاسخ با داشتن ngramهای مختلف متفاوت عمل می کند چرا که برای یک متن ممکنه است در نظر گرفتن کلمات به صورت مستقل و unigram نتیجه مناسبی نداشته باشد و تنها ترکیب های چندتایی ما را به نتایج درستی در تصمیم گیری برساند مانند مثال زیر که خروجی برنامه است.

['<start>', 'emperor', 'richard', 'haydn', 'dog', 'betroth', 'johanna', 'joan', 'fontain', 'dog', 'howev', 'virgil', 'bing', 'crosbi', 'arriv', 'town', 'sell', 'record', 'player', 'emperor', 'dog', 'attack', 'johanna', 'dog', 'reveng', 'attack', 'virgil', 'banish', 'town', 'psychoanalyst', 'insist', 'johanna', 'dog', 'must', 'confront', 'dog', 'overcom', 'doggi', 'fear', 'arrang', 'dog', 'fall', 'love', 'virgil', 'johanna', 'rest', 'film', 'pass', 'romanc', 'end', 'johanna', 'dog', 'give', 'birth', 'father', 'dog', 'stori', 'weak', 'vehicl', 'use', 'tri', 'creat', 'stori', 'human', 'terribl', 'storylin', 'main', 'music', 'piec', 'rubbish', 'bad', 'song', 'dread', 'choreographi', 'extrem', 'bore', 'film', 'bing', 'mani', 'word', 'sentenc', 'deliv', 'almost', 'irrit', 'manner', 'funni', 'ever', 'meant', 'bing', 'joan', 'done', 'much', 'better']

y_test: False

unigram: 0.00000000018930282923 0.00000002048303390717 True

bigram: 2365.02400839971369350678 0.00109233333028247847 False

trigram: 0.00000694624858749255 0.00000080109583952223 False

از طرف دیگر در برخی متون تنها بررسی مستقل کلمات راهبرد بهینه و بهتر بوده و ترکیب چندتایی کلمات خیلی مناسب نیست . مانند مثال زیر که اصلا پاسخ مناسبی از ۳گرام ها نمیگیریم.

['<start>', 'film', 'requir', 'lot', 'patienc', 'focus', 'mood', 'charact', 'develop', 'plot', 'simpl', 'mani', 'scene', 'take', 'place', 'set', 'franc', 'austen', 'sandi', 'denni', 'charact', 'apart', 'film', 'build', 'disturb', 'climax', 'charact', 'creat', 'atmospher', 'rife', 'sexual', 'tension', 'psycholog', 'trickeri', 'interest', 'robert', 'altman', 'direct', 'consid', 'style', 'structur', 'film', 'still', 'trademark', 'altman', 'audio', 'style', 'evid', 'think', 'realli', 'make', 'film', 'work', 'brilliant', 'perform', 'sandi', 'denni', 'definit', 'one', 'darker', 'charact', 'play', 'perfect', 'convinc', 'scari', 'michael', 'burn', 'good', 'job', 'mute', 'young', 'man', 'regular', 'altman', 'player', 'michael', 'murphi', 'small', 'part', 'solemn', 'moodi', 'set', 'fit', 'content', 'stori', 'well', 'short', 'movi', 'power', 'studi', 'loneli', 'sexual', 'repress', 'desper', 'patient', 'soak', 'atmospher', 'pay', 'attent', 'wonder', 'written', 'script', 'prais', 'robert', 'altman', 'one', 'mani', 'film', 'deal', 'unconvent', 'fascin', 'subject', 'matter', 'film', 'disturb', 'sincer', 'sure', 'elicit', 'strong', 'emot', 'respons', 'viewer', 'want', 'see', 'unusu', 'film', 'might', 'even', 'say', 'bizarr', 'worth', 'time', 'unfortun', 'difficult', 'find', 'video', 'store', 'may', 'buy', 'internet']

y_test: True

unigram: 0.00000000037924195660 555.02757185687346463965 True

bigram: 0.00000009151660757814 10.617294233257169366688 True

trigram: 0.01090983298296508558 0.00017707702542392575 False

البته مشکلات مطرح شده می تواند ناشی از این باشد که ما علاوه بر حذف کردن استاپ ورد ها ریشه کلمات را نیز به دست آورده ایم(stemming).

با وجود اینکه کلمات استاپ ورد به خودی خود بار معنایی نداشته و خیلی اهمیت خاصی به متن اضافه نکرده و به اصطلاح فیچر محسوب نمیشوند اما وقتی ترکیب کلمات مطرح میشود اهمیت آنا را نمی توان نادیده گرفت چراکه ممکن است وجود یا عدم وجود این کلمات معنای متفاوتی به ترکیب های چندتایی ببخشد.