Soutonglang_CS585_Homework01_Problem1-5

September 12, 2023

1 CS 585 - HW 1 - Getting Started

Tania Soutonglang A20439949

[ ]:
```
#NLTK setup - uncomment and run first time you import NLTK
import nltk
nltk.download('punkt')

import pandas as pd
from nltk.tokenize import word_tokenize
from csv import QUOTE_NONE

import numpy as np
from nltk import probability
import math
```

```
[nltk_data] Downloading package punkt to
[nltk_data]     C:\Users\tsout\AppData\Roaming\nltk_data…
[nltk_data]   Package punkt is already up-to-date!
```

[ ]:
```
# import sst dataset
df_sst = pd.read_csv("SST-2/train.tsv",delimiter="\t")
df_sst.head(3)
```

[ ]:
```
                                            sentence  label
0       hide new secretions from the parental units       0
1               contains no wit , only labored gags       0
2  that loves its characters and communicates som…       1
```

[ ]:
```
# import qnli dataset
df_qnli = pd.read_csv("QNLI/dev.tsv",delimiter="\t",quoting=QUOTE_NONE)
df_qnli.head(3)
```

[ ]:
```
   index                                           question  \
0      0  What came into force after the new constitutio…
1      1  What is the first major city in the stream of …
2      2  What is the minimum required if you want to te…

                                            sentence           label
```

```
0  As of that day, the new constitution heralding…      entailment
1  The most important tributaries in this area ar…  not_entailment
2  In most provinces a second Bachelor's Degree s…  not_entailment
```

1.1 Problem 1 - Representing English Text

[ ]:
```
# create sst series with only sentence column
df_sstData = df_sst[['sentence']].dropna()
df_sstData = df_sstData['sentence'].str.lower()
df_sstData.head(3)
```

[ ]:
```
0         hide new secretions from the parental units
1                 contains no wit , only labored gags
2    that loves its characters and communicates som…
Name: sentence, dtype: object
```

[ ]:
```
# create qnli series with only sentence column
df_qnliData = df_qnli[['sentence']].dropna()
df_qnliData = df_qnliData['sentence'].str.lower()
df_qnliData.head(3)
```

[ ]:
```
0    as of that day, the new constitution heralding…
1    the most important tributaries in this area ar…
2    in most provinces a second bachelor's degree s…
Name: sentence, dtype: object
```

[ ]:
```
df_qnliData.head(3)
```

[ ]:
```
0    as of that day, the new constitution heralding…
1    the most important tributaries in this area ar…
2    in most provinces a second bachelor's degree s…
Name: sentence, dtype: object
```

[ ]:
```
# create token list for sst data
sst_tokens = [word_tokenize(t) for t in df_sstData]
temp = []
for words in sst_tokens:
    temp.extend(words)
sst_vocab = np.unique(temp)

for token in range(10):
    print(sst_vocab[token])
```

```
!
#
$
&
'
```

```
''
'30s
'40s
'50s
'53
```

[ ]:
```python
# create token list for qnli data
qnli_tokens = [word_tokenize(t) for t in df_qnliData]
temp = []
for words in qnli_tokens:
    temp.extend(words)
qnli_vocab = np.unique(temp)

for token in range(10):
    print(qnli_vocab[token])
```

```
!
#
$
%
&
'
''
'aided
'apothecary
'bath
```

1.2 Problem 2 - Word Probability

[ ]:
```python
def create_probDist(tokenList):
    # get the total number of tokens
    count = len(tokenList)

    # create a frequency distribution of the token list
    freqDist = probability.FreqDist(tokenList)

    # calculate the probability of each token
    probDist = {token: freq / count for token, freq in freqDist.items()}

    return probDist
```

[ ]:
```python
sst_probDist = create_probDist(sst_vocab)
print(sum(sst_probDist.values()))
```

```
1.0000000000001843
```

[ ]:
```python
qnli_probDist = create_probDist(qnli_vocab)
print(sum(qnli_probDist.values()))
```

```
0.9999999999999619
```

1.3 Problem 3 - Entropy

[ ]:
```
def calc_entropy(probDist):
    entropy = 0.0

    # calculate the entropy of each word
    for prob in probDist.values():
        if prob > 0:
            entropy += -prob * math.log2(prob)

    return entropy
```

[ ]:
```
sst_entropy = calc_entropy(sst_probDist)
print(sst_entropy)
```

```
13.853699420295655
```

[ ]:
```
qnli_entropy = calc_entropy(qnli_probDist)
print(qnli_entropy)
```

```
13.946358033889211
```

1.4 Problem 4 - KL Divergence

[ ]:
```
def calc_KLDivergence(probDist_a, probDist_b):
    kl_divergence = 0.0

    for key, prob_a in probDist_a.items():
        # Get the corresponding probability from the 2nd list, otherwise␣
 ↪default to 0 if it doesn't exist
        prob_b = probDist_b.get(key, 0)

        # calculate the KL divergence
        if prob_b > 0:
            kl_divergence += prob_a * math.log2(prob_a / prob_b)

    return kl_divergence
```

[ ]:
```
sst_KLqnli = calc_KLDivergence(sst_probDist, qnli_probDist)
print(sst_KLqnli)
```

```
0.03225275843102831
```

[ ]:
```
qnli_KLsst = calc_KLDivergence(qnli_probDist, sst_probDist)
print(qnli_KLsst)
```

```
-0.03024641047846031
```

1.5 Problem 5 - Entropy Rate

[ ]:
```
def calc_perWordEntropyRate(doc, probDist):
    # split the inputted document into tokens
    tokens = doc.split()
    token_count = len(words)
    entropy_rate = 0.0

    for word in tokens:
        # get the probability of the word, otherwise it will be 0
        probability = probDist.get(word, 0)
        if probability > 0:
            entropy_rate += -probability * math.log2(probability)

        # find the per-word rate
        if token_count > 0:
            entropy_rate /= token_count

    return entropy_rate
```

[ ]:
```
# movie review of The Nun II from Rotten Tomatoes (https://www.rottentomatoes.
 ↪com/m/the_nun_ii/reviews)
review = "A narratively bland sequel that grants star Taissa Farmiga a bit more␣
 ↪agency (while wasting Storm Reid), The Nun II proves that while there's␣
 ↪plenty of box office in The Conjuring Universe, the storytelling techniques␣
 ↪are phoning it in."
```

[ ]:
```
# per-word entropy using sst probability distribution
sst_perWord = calc_perWordEntropyRate(review, sst_probDist)
print(sst_perWord)
```

```
4.836273055258784e-07
```

[ ]:
```
# per-word entropy using qnli probability distribution
qnli_perWord = calc_perWordEntropyRate(review, qnli_probDist)
print(qnli_perWord)
```

```
4.5657562774588297e-07
```