U3_GSV

March 12, 2023

Gustavo Santana Velázquez

1 Introducción

En este trabajo, se escribieron varios scripts en Python que permiten determinar la probabilidad de formar bigramas y frases a partir de un texto, utilizando el *Modelo de Máxima Versoimilitud* (MLE) y el *Modelo de Máxima Verosimilitud con suavizado de Laplace.*

El modelo MLE, permite estimar la probabilidad de ocurrencia de un brigrama o frase a partir de las frecuencias de los mismos corpus de texto. Por otro lado, el modelo de MLE con suavizado de Laplace permite corregir los problemas de estimación de probabilidades que se presentan cuando hay bigramas o frases que no aparecen en el corpus.

Para calcular la probabilidad de un bigrama xy en el corpus, donde cada variable representa una palabra, se utiliza la siguiente fórmula para el modelo MLE:

$$P(y|x) = \frac{C(xy)}{C(x)}$$

Es sencillo observar que utilizando este modelo, los bigramas que no ocurren en el texto tendrían una probabilidad de 0, por lo que al momento de medir la probabilidad de ocurrencia de una oración, se obtendría un resultado de 0 con que uno solo de los bigramas que conforman la oración no existan en el corpus.

Para corregir esto se utiliza la técnica de suavizado Laplace:

$$P_{Lap}(y|x) = \frac{C(xy) + 1}{C(x) + V}$$

donde v es el tamaño del vocabulario (el número total de palabras diferentes en el corpus).

Para este ejercicio el corpus fueron documentso de una conferencia del parlamento europeo en español. Sin embargo todas las funciones creadas funcionan para cualquier texto.

2 Desarrollo

Se comenzó por elaborar una función para el pre-procesamiento del texto (remover símbolos y cambiar todos los caracteres a minúsuclas). Posteriormente se hizo la carga del texto (*Apéndice 5.2*).

Posteriormente se creó una función para guardar las probabilidades de ocurrencia de cada bigrama en el texto en diccionarios. El reto en esta función era que los diccionarios no pueden tener listas como llaves, por lo que se tuvo que castear a *strings* cada bigrama previo a hacer el cálculo.

También se desarrolló la función para la técnica de suavizado de laplace, en este caso también se regresa el conteo de los unigramas para poderlos utilizar si se quiere estimar la probabilidad de bigramas que no ocurren en el texto (*Apéndice 5.3*).

Los objetivos del proyecto eran utilizar el modelo MLE y MLE con suavizado de Laplace para:

1. Calcular los modelos de probabilidad para bigramas sobre el corpus,
2. Obtener la probabilidad de oraciones, y
3. Predicción de palabras: generar un script que sugiera las 5 palabras con mayor probabilidad dada una palabra inicial.

Por lo que finalmente se escribieron tres funciones para cumplir con los puntos solicitados y se realizaron las pruebas (*Apéndice 5.4*).

2.1 Resultados

En el Apéndice 5.4 se pueden observar los resultados obtenidos.

En la sección 5.4.2 se observa la probabilidad de ocurrencia de 7 oraciones diferentes utilizando ambos modelos. Podemos observar que en la sexta, *el abismo de la cantera entre pobres y ricos*, utilizando el modelo MLE se obtiene un resultado de 0, mientras que con la técnica de suavizado obtenemos una probabilidad mayor a 0 (aunque bastante pequeña). Esto es debido a que ni el bigrama [la cantera] ni [cantera entre] existen en el corpus.

Finalmente, en la sección 5.4.3 se lleva a cabo una simulación de predicción de palabras utilizando la probabilidad de los bigramas. Se observa que ambos modelos regresan el mismo top 5.

3 Conclusiones

El modelo de MLE ha permitido estimar la probabilidad de ocurrencia de bigramas a partir de un corpus de texto, basándose en la frecuencia con que estos bigramas aparecen en el corpus. Sin embargo, este modelo puede presentar problemas de estimación cuando hay bigramas que no aparecen en el corpus, lo que puede afectar la precisión de las predicciones.

Para resolver este problema, se ha utilizado el modelo de MLE con suavizado de Laplace, que agrega una unidad a cada conteo de bigramas para corregir las estimaciones de probabilidad y evitar que la probabilidad sea cero.

En general, este programa tiene aplicaciones prácticas en el procesamiento del lenguaje natural, como la generación de texto y la corrección ortográfica, y puede ser utilizado para mejorar la precisión de las predicciones de palabras en diferentes contextos (autocompletado de texto, corrección ortográfica, entre otras).

4 Bibliografía

Jurafsky, D., James H. M. (2021). *Chapter 3 N-gram Language Models.* Speech and Language Processing (3rd edition). Stanford University

5 Apéndice

5.1 Librerías

[ ]:
```
from nltk.corpus import stopwords
import io
from collections import Counter
import numpy as np
```

5.2 Carga y procesamiento de datos

[ ]:
```
symbols = list(set("«-;:,.\\-\"'/()[]¿?¡!{}~<>|\r_'\n'`"))

def preprocess(txt):
    '''
    Función para remover símbolos y cambiar a minúsculas un texto dado
    '''
    txt = txt.lower()
    return ''.join([c for c in txt if c not in symbols])
```

[ ]:
```
europarl = ' '.join([f'<s> {preprocess(line)} </s>' for line in io.open(f'./
 ↪europarl.es', 'r', encoding = 'UTF-8').readlines()])
```

5.3 Definir funciones

[ ]:
```
def mle_bigram(text):
    '''
    Regresa la probabilidad de ocurrencia de cada bigrama en un texto siguiendo
    el modelo de Máxima verosimilitud.
    '''
    # Get list of bigrams from the text
    bigrams = Counter([f'{text[i]} {text[i+1]}' for i in range(len(text)-1)])
    unigrams = Counter(text)

    # Get probabilites
    mle_probs = {}
    for key, value in bigrams.items():
        mle_probs[key] = value/unigrams[key.split()[0]]

    # Return probabilities
    return mle_probs

def mle_bigram_laplace(text):
    '''
    Regresa la probabilidad de ocurrencia de cada bigrama en un texto siguiendo
    el modelo de Máxima verosimilitud con suavizado de Laplace. También regresa
    el conteo de los unigramas en el texto para bigramas que no ocurran en el
```

```
    texto.
    '''
    # Get list of bigrams from the text
    bigrams = Counter([f'{text[i]} {text[i+1]}' for i in range(len(text)-1)])
    unigrams = Counter(text)

    # Get value of V
    v = len(unigrams)

    # Get probabilites
    mle_probs = {'v': v}
    for key, value in bigrams.items():
        mle_probs[key] = (value + 1)/(unigrams[key.split()[0]] + v)

    # Return probabilities
    return mle_probs, unigrams
```

[ ]:
```
def prob_oracion(sentence, probs, unigrams_laplace = False):
    '''
    Obtiene la probabilidad de ocurrencia de una oración a partir de las
    probabilidades de bigramas dadas. Toma como argumentos, la oración,
    las probabilidades de bigramas y la cuenta de unigramas si se utiliza
    la función suavizada de laplace.
    '''
    words = f'<s> {sentence} </s>'.split()
    prob = 1
    # Loop para multiplicar las probabilidades de cada bigrama
    for i in range(len(words)-1):
        if f'{words[i]} {words[i+1]}' in probs:
            prob *= probs[f'{words[i]} {words[i+1]}']
        # Caso en el que se está utilizando el suavizado de laplace en bigramas
        # que no aparecen en el texto
        elif unigrams_laplace != False:
            prob *= 1/(unigrams_laplace[words[i]] + probs['v'])
        # Caso MLE no suavizado en el que un bigrama no exista (0 probabilidad)
        else:
            prob = 0
            break
    return f'P(<s> {sentence} </s>): {prob}'


def top5(probs, word):
    '''
    Regresa las 5 palabras con mayor probabilidad de ocurrir después de una
    palabra dada. Toma como argumentos un diccionario con la probabilidad MLE o
    MLE suavizado de bigramas previamente calculado y la palabra base.
    '''
```

```
    # Regresa solo los bigramas que tengan la palabra dada como word_1, excluye
    # bigramas que representen el inicio o final de una oración
    res = {key:value for key,value in probs.items() if key.split()[0] == word␣
 ↪and '<s>' not in key and '</s>' not in key}
    # Ordena las palabras de mayor a menor probabilidad y regresa 5
    values = list(sorted(res.items(), key=lambda x:x[1], reverse=True)[:5])
    return values

def prediccion_palabras(probs, sentence):
    '''
    Itera la función top5 para las palabras de una oración dada.
    Imprime los resultados
    '''
    words = sentence.split()
    sent = ''
    for word in words:
        sent += f' {word}'
        print(f'\n{sent}...')
        predictions = top5(probs, word)
        for predict in predictions:
            print(f'\t{predict[0].split()[1]}: {predict[1]}')
```

5.4 Puntos solicitados

5.4.1 Calcular modelos de probabilidad MLE y MLE con suavizado de Laplace para un modelo de bigramas

[ ]:
```
# MLE
mle_probs = mle_bigram(europarl.split())

# Laplace
mle_laplace, mle_unigrams = mle_bigram_laplace(europarl.split())
```

5.4.2 Calcular probabilidades de las siguientes oraciones

Modelo MLE

[ ]:
```
print(prob_oracion('el parlamento debe enviar un mensaje', mle_probs))
print(prob_oracion('el parlamento debe enviar un consejo', mle_probs))
print(prob_oracion('el abismo entre pobres y ricos', mle_probs))
print(prob_oracion('el abismo entre ricos y pobres', mle_probs))
print(prob_oracion('el abismo de la cantera entre pobres y ricos', mle_probs))
print(prob_oracion('la comisión debe ser totalmente transparente', mle_probs))
print(prob_oracion('la comisión debe ser transparente', mle_probs))
```

```
P(<s> el parlamento debe enviar un mensaje </s>): 4.452453175934305e-13
P(<s> el parlamento debe enviar un consejo </s>): 3.3686200213034554e-13
P(<s> el abismo entre pobres y ricos </s>): 3.820757157773409e-17
```

```
P(<s> el abismo entre ricos y pobres </s>): 8.677949590529031e-15
P(<s> el abismo de la cantera entre pobres y ricos </s>): 0
P(<s> la comisión debe ser totalmente transparente </s>): 3.609720553302905e-11
P(<s> la comisión debe ser transparente </s>): 2.560495112476194e-09
```

Modelo MLE Suavizado Laplace

[ ]:
```
print(prob_oracion('el parlamento debe enviar un mensaje', mle_laplace,␣
 ↪mle_unigrams))
print(prob_oracion('el parlamento debe enviar un consejo', mle_laplace,␣
 ↪mle_unigrams))
print(prob_oracion('el abismo entre pobres y ricos', mle_laplace, mle_unigrams))
print(prob_oracion('el abismo entre ricos y pobres', mle_laplace, mle_unigrams))
print(prob_oracion('el abismo de la cantera entre pobres y ricos', mle_laplace,␣
 ↪mle_unigrams))
print(prob_oracion('la comisión debe ser totalmente transparente', mle_laplace,␣
 ↪mle_unigrams))
print(prob_oracion('la comisión debe ser transparente', mle_laplace,␣
 ↪mle_unigrams))
```

```
P(<s> el parlamento debe enviar un mensaje </s>): 6.008349191881304e-21
P(<s> el parlamento debe enviar un consejo </s>): 9.389536311730697e-20
P(<s> el abismo entre pobres y ricos </s>): 1.6784915159931863e-26
P(<s> el abismo entre ricos y pobres </s>): 1.1649530402666998e-24
P(<s> el abismo de la cantera entre pobres y ricos </s>): 2.2172365002890588e-37
P(<s> la comisión debe ser totalmente transparente </s>): 7.519960394741226e-19
P(<s> la comisión debe ser transparente </s>): 4.138712748160308e-15
```

5.4.3 Predicción de palabras (las 5 palabras más probables)

Modelo MLE

[ ]:
```
prediccion_palabras(mle_probs, 'los tribunales nacionales')
```

```
los…
        estados: 0.06970770248705874
        países: 0.04592690616121122
        derechos: 0.03643015607363873
        que: 0.030942279998443154
        ciudadanos: 0.025940917759701088

los tribunales…
        nacionales: 0.12121212121212122
        de: 0.11363636363636363
        y: 0.05303030303030303
        en: 0.045454545454545456
        del: 0.03787878787878788
```

```
los tribunales nacionales…
       de: 0.1360787824529991
       y: 0.135183527305282
       en: 0.050134288272157566
       que: 0.04207699194270367
       a: 0.018800358102059087
```

[ ]:
```
prediccion_palabras(mle_laplace, 'la unión europea')
```

```
la…
       comisión: 0.06891698430053096
       unión: 0.0376026143632926
       política: 0.011597914791569765
       sra: 0.009972998604625877
       ue: 0.009610563767389321

la unión…
       europea: 0.08063568964251633
       y: 0.0031380753138075313
       en: 0.0026058871026939734
       se: 0.0014314027747192248
       de: 0.0013396461865961976

la unión europea…
       y: 0.011088746569075937
       en: 0.007099725526075023
       de: 0.0064226898444647755
       que: 0.0038243366880146385
       no: 0.003751143641354071
```

[ ]:
```
prediccion_palabras(mle_laplace, 'parlamento europeo de las personas')
```

```
parlamento…
       europeo: 0.0348237633972966
       y: 0.0076792751648189755
       en: 0.004640713049243122
       aprueba: 0.0031122242274686015
       que: 0.0031122242274686015

parlamento europeo…
       de: 0.010966177998383932
       y: 0.006926007156874062
       en: 0.004155604294124437
       que: 0.0025010581399823
       para: 0.002327907961060449
```

```
parlamento europeo de…
       la: 0.1396769358566888
       los: 0.07254045073678128
       las: 0.04456804391794279
       que: 0.025683509101415775
       una: 0.015376697486275643

parlamento europeo de las…
       enmiendas: 0.010697552113214763
       personas: 0.009717441193344807
       que: 0.008904666284184357
       medidas: 0.008366800535475234
       empresas: 0.008163606808185122

parlamento europeo de las personas…
       que: 0.006718314973974095
       y: 0.0024613646451196384
       de: 0.002178912964532139
       mayores: 0.0016543598434410685
       en: 0.0015534842432312472
```

[ ]:
```
prediccion_palabras(mle_probs, 'ejemplo del modelo')
```

```
ejemplo…
       en: 0.13126491646778043
       de: 0.10441527446300716
       la: 0.07279236276849642
       el: 0.06921241050119331
       que: 0.03579952267303103

ejemplo del…
       consejo: 0.06169993117687543
       parlamento: 0.053578802477632484
       sr: 0.03695801789401239
       grupo: 0.025430144528561596
       día: 0.023158981417756366

ejemplo del modelo…
       de: 0.24013157894736842
       social: 0.18421052631578946
       europeo: 0.05592105263157895
       que: 0.039473684210526314
       en: 0.03289473684210526
```

Modelo MLE Suavizado Laplace

[ ]:
```
prediccion_palabras(mle_laplace, 'los tribunales nacionales')
```

```
los…
       estados: 0.03608766593477429
       países: 0.02377978768406422
       derechos: 0.018864693914549886
       que: 0.016024414318232177
       ciudadanos: 0.013435932558467457

los tribunales…
       nacionales: 0.0003539307128580946
       de: 0.00033311125916055963
       y: 0.00016655562958027982
       en: 0.00014573617588274483
       del: 0.00012491672218520986

los tribunales nacionales…
       de: 0.0031213660566742152
       y: 0.003100964971336475
       en: 0.001162861864251178
       que: 0.0009792520962115185
       a: 0.0004488238774302793
```

[ ]:
```
prediccion_palabras(mle_laplace, 'la unión europea')
```

```
la…
       comisión: 0.06891698430053096
       unión: 0.0376026143632926
       política: 0.011597914791569765
       sra: 0.009972998604625877
       ue: 0.009610563767389321

la unión…
       europea: 0.08063568964251633
       y: 0.0031380753138075313
       en: 0.0026058871026939734
       se: 0.0014314027747192248
       de: 0.0013396461865961976

la unión europea…
       y: 0.011088746569075937
       en: 0.007099725526075023
       de: 0.0064226898444647755
       que: 0.0038243366880146385
       no: 0.003751143641354071
```

[ ]:
```
prediccion_palabras(mle_laplace, 'parlamento europeo de las personas')
```

```
parlamento…
       europeo: 0.0348237633972966
       y: 0.0076792751648189755
       en: 0.004640713049243122
       aprueba: 0.0031122242274686015
       que: 0.0031122242274686015

parlamento europeo…
       de: 0.010966177998383932
       y: 0.006926007156874062
       en: 0.004155604294124437
       que: 0.0025010581399823
       para: 0.002327907961060449

parlamento europeo de…
       la: 0.1396769358566888
       los: 0.07254045073678128
       las: 0.04456804391794279
       que: 0.025683509101415775
       una: 0.015376697486275643

parlamento europeo de las…
       enmiendas: 0.010697552113214763
       personas: 0.009717441193344807
       que: 0.008904666284184357
       medidas: 0.008366800535475234
       empresas: 0.008163606808185122

parlamento europeo de las personas…
       que: 0.006718314973974095
       y: 0.0024613646451196384
       de: 0.002178912964532139
       mayores: 0.0016543598434410685
       en: 0.0015534842432312472
```

[ ]:
```
prediccion_palabras(mle_laplace, 'ejemplo del modelo')
```

```
ejemplo…
       en: 0.004457802162336614
       de: 0.0035501048894626434
       la: 0.002481039212522188
       el: 0.0023600129094723254
       que: 0.0012304340810069389

ejemplo del…
       consejo: 0.023310810810810812
```

```
        parlamento: 0.020244282744282745
        sr: 0.013968295218295219
        grupo: 0.009615384615384616
        día: 0.008757796257796258

ejemplo del modelo…
        de: 0.001535142311841341
        social: 0.0011824744834453573
        europeo: 0.0003734129947722181
        que: 0.00026968716289104636
        en: 0.0002281968301385777
```