Métodos de Modelado y Optimización
Laboratorio 2

Descripción:

Un n-grama es una subsecuencia de n elementos de una secuencia dada. Este tipo de subsecuencias son muy utilizadas en muchos campos del conocimiento para permitir identificar patrones y comportamientos; y permitir modelar estos patrones como eventos probabilísticos. Son usados en el campo de procesamiento de lenguaje natural para analizar el habla e incluso generar modelos que permiten escribir "parecido" a un autor específico. En el laboratorio de hoy, utilizaremos cadenas de Markov para modelar una lista de palabras como una secuencia de n-gramas, para con ello diseñar un sistema que nos permita generar nombres de Pokemon aleatorios. Para ello su profesor le provee el archivo fuente de datos a utilizar para este experimento.

Implemente los siguientes métodos solicitados en lenguaje Python.

1. (5%) Método `load_words(filename)` que permite cargar un archivo de nombre `filename` compuesto por palabras separadas por saltos de línea. Cargue las diferentes palabras del documento de texto a un arreglo de Python, elimine cualquier caracter especial sobrante (saltos de línea, espacios en blanco, etc.).

2. (5%) Método `add_decorators(words, decorator, n)` que recibe un arreglo de hileras `words`, un caracter `decorator` y un número entero `n`. El método debe retornar un arreglo con cada una de las palabras de `words` en minúsculas y agregándoles `n`-veces el caracter `decorator` al inicio y final. No debe modificar el arreglo original.
    a. Entrada de ejemplo: `['hello', 'world'], '$', 1`
    b. Salida de ejemplo: `['$hello$','$world$']`
    c. Entrada de ejemplo: `['hello', 'world'], '$', 2`
    d. Salida de ejemplo: `['$$hello$$','$$world$$']`

3. (10%) Método `get_sequences(words, n)` que recibe un arreglo de hileras `words` y un número `n`. El método debe retornar un arreglo con todas las subsecuencias únicas de `n`-caracteres dentro de las hileras. El arreglo a retornar debe estar ordenado alfabéticamente.
    a. Entrada de ejemplo: `['$hello$','$world$'], 1`
    b. Salida de ejemplo: `['$', 'd', 'e', 'h', 'l', 'o', 'r', 'w']`
    c. Entrada de ejemplo: `['$$hello$$','$$world$$'], 2`
    d. Salida de ejemplo: `['$$', '$h', '$w', 'd$', 'el', 'he', 'ld', 'll', 'lo', 'o$', 'or', 'rl', 'wo']`

4. (25%) Método `calculate_transitions(words, sequences)` que recibe un arreglo de hileras `words` y un arreglo de subsecuencias `sequences` (n-gramas). El método debe retornar una matriz de tamaño L x L (donde L es la cantidad de subsecuencias

en `sequences`) con la probabilidad de transición entre cada pareja de subsecuencias obtenidas de procesar el arreglo de palabras.

a. Note que esto equivale a contar la cantidad de veces que cada subsecuencia aparece precedida por otra, lo que requiere procesar cada una de las palabras del arreglo `words`.
b. Note que aunque el tamaño de las subsecuencias sea mayor a 1, el proceso de lectura igual avanza de letra en letra.
    i. Ejemplo: `perro` leído con secuencias de tamaño 2 se leería como los bigramas `“pe”, “er”, “rr”, “ro”`
c. Entrada de ejemplo: `['$hello$','$world$'], ['$','d','e','h','l','o','r','w']`
d. Salida de ejemplo:
```
[ [0, 0, 0, 0.5, 0, 0, 0, 0.5]
  [1, 0, 0, 0, 0, 0, 0, 0]
  [0, 0, 0, 0, 1, 0, 0, 0]
  [0, 0, 1, 0, 0, 0, 0, 0]
  [0, 0.33, 0, 0, 0.33, 0.33, 0, 0]
  [0.5, 0, 0, 0, 0, 0, 0.5, 0]
  [0, 0, 0, 0, 1, 0, 0, 0]
  [0, 0, 0, 0, 0, 1, 0, 0]     ]
```
e. Note que los índices de las columnas/filas nos indican cuál subsecuencia representa, por ejemplo, en la primera fila tenemos dos casillas con probabilidad de 0.5, esto es porque el símbolo `‘$’` (fila 0) puede ir seguido por `‘h’` (columna 3) o por `‘w’` (columna 7). También note que las probabilidades de cada fila suman 1. Note que si hubiera 2 palabras cuyos primeros dos caracteres fueran `‘$h’` entonces la probabilidad de pasar a `‘h’` desde `‘$’` sería de 0.67 y solo un 0.33 para `‘w’`.

5. (5%) Método `create_model(words, ngrams)` que recibe un arreglo de hileras `words` y un número entero `ngrams`, este método ejecuta los llamados a las funciones `add_decorators`, `get_sequences` y `calculate_transitions`; para agregar los decoradores a las palabras, obtener las subsecuencias de caracteres de tamaño `ngrams` y crear la matriz de transiciones correspondientes. Este método debe retornar una tupla `(transition, sequences)` que contiene la matriz de transiciones del modelo así como el arreglo de subsecuencias del modelo.
    a. Entrada de ejemplo: `[‘hello’, ‘world’], 1`
    b. Salida de ejemplo:
```
([ [0, 0, 0, 0.5, 0, 0, 0, 0.5]
   [1, 0, 0, 0, 0, 0, 0, 0]
   [0, 0, 0, 0, 1, 0, 0, 0]
   [0, 0, 1, 0, 0, 0, 0, 0]
   [0, 0.33, 0, 0, 0.33, 0.33, 0, 0]
   [0.5, 0, 0, 0, 0, 0, 0.5, 0]
   [0, 0, 0, 0, 1, 0, 0, 0]
   [0, 0, 0, 0, 0, 1, 0, 0]     ],
 ['$','d','e','h','l','o','r','w'])
```
    c. Note que la salida consiste de una matriz de transiciones y las secuencias necesarias para interpretar las filas/columnas de dicha matriz.

6. (25%) Método `generate_word(model, seed)` que recibe una tupla `(transition, sequences)` generada por `create_model` y una semilla aleatoria `seed`. La función retorna la hilera-resultado de realizar un recorrido aleatorio usando la matriz de transiciones que forman parte de `model`.
    a. Note que el estado inicial y final son idénticos, el recorrido aleatorio trata de iniciar en el estado inicial y avanzar de manera aleatoria hasta volver a regresar a dicho estado.
    b. El estado inicial (de elegirse un caracter decorador como `'$'`) siempre se encontrará de primero en el arreglo de secuencias, dado que vienen con ordenamiento alfabético.
    c. Dado que este es un recorrido aleatorio con semilla, deberá crear un objeto de la clase random, este objeto se utilizará para generar los números aleatorios del recorrido:

```
import random
r = random.Random()
r.seed(semilla)
```

    d. El recorrido aleatorio se hace de la siguiente manera: mientras no hayamos llegado al estado final, deberemos avanzar un estado, para ello se genera un número en el intervalo [0,1]. Dado que la suma de todas las filas da 1, este número aleatorio nos indicará a cuál de todos los estados siguientes avanzar.
        i. Ejemplo, si tenemos la fila [0.3, 0.1, 0, 0.4, 0.2]
        ii. Se genera 0.1: el estado salta al de la columna 0
        iii. Se genera 0.35: el estado salta al de la columna 1
        iv. Se genera 0.45: el estado salta al de la columna 3
        v. Básicamente si el número aleatorio es menor al valor de la columna, entonces el estado cambia al de dicha columna; sino, se le resta el valor de dicha columna y se avanza la comparación a la siguiente columna.
    e. Caso de prueba: `model(n=1), seed=17` `'Mowaror'`
    f. Caso de prueba: `model(n=2), seed=42` `'Palassich'`
    g. Caso de prueba: `model(n=3), seed=21` `'Corinoon'`

7. (20%) Método `get_probability(model, word)` que recibe una tupla `(transition, sequences)` generada por `create_model` y una hilera `word`. El método retorna la probabilidad de generar dicha secuencia utilizando la matriz de transiciones del modelo.
    a. A la palabra ingresada se le deben agregar los decoradores y la conversión a minúsculas necesarias.
    b. Asuma que todas las subsecuencias de la hilera brindada por el usuario son válidas (no es necesario realizar validación)
    c. Entrada de ejemplo: `model, "mew"` con n=1, 2 y 3
    d. Salida de ejemplo: `3.43e-05, 0.00039, 0.0012` respectivamente

8. (5%) Agregue dentro del documento, como documentación interna/comentario, su razonamiento y respuesta para las siguientes preguntas:
    a. ¿Por qué la probabilidad de formar un nombre parece aumentar conforme `n` incrementa?

b. ¿Encontró algún otro nombre interesante para un Pokemon? ¿Con qué n y con qué semilla? ¿Encontró algún nombre que no tiene sentido? ¿Con qué n y semilla? (Incluya el código de demostración)
c. Explique en sus propias palabras, ¿de qué manera se están usando las cadenas de Markov para modelar los nombres?

Nota: Se permite utilizar la biblioteca de soporte numpy, pero no se permite utilizar bibliotecas que realicen el trabajo por usted. Cualquier otra biblioteca que se desee usar debe ser consultada primero con el profesor.