MANUAL DE USUARIO

PREDICCIÓN POR N-GRAMAS: FASE 1

Miguel Abadía

25 Noviembre 2020

El objetivo de este documento es explicar cómo utilizar los archivos y funciones generados para realizar la predicción de palabras a partir de una frase definida. Esta predicción está basada en el modelo de N-gramas y el algoritmo Viterbi.

Este documento se divide en dos secciones:

1. Explicación de archivos y funciones
2. Modo de uso

1. EXPLICACIÓN DE ARCHIVOS Y FUNCIONES

Todas las funciones generadas están comentadas para favorecer su interpretación. En primer lugar, se puede encontrar una breve descripción y la definición de los parámetros de entrada y de salida. Además, se añaden comentarios línea a línea explicando qué se realiza en cada una.

Así, los archivos generados (entregados en la carpeta *Entrega Fase 1*) son los siguientes:

a. Scripts (se explica su utilización en la sección *2. Modo de uso*)
   - **scriptLanguajeModel**: Script para la generación de los modelos de lenguaje.
   - **scriptPrediccion**: Script para la obtención de predicciones.

b. Archivos de funciones (se explican las funciones a continuación)
   - **generateGeneralLanguajeModel**: Archivo que contiene todas las funciones necesarias para la generación de los modelos de lenguaje.
   - **viterbiNbest**: Archivo que contiene todas las funciones necesarias para la ejecución del algoritmo Viterbi.
   - **calcularPrediccion**: Archivo que contiene la función para la obtención de predicciones

c. Datos de entrada
   - **es-verbs.txt:** Archivo de verbos en español con todas sus conjugaciones. Extraído de la biblioteca *pattern* para Python.
   - **frasesTest.txt:** Ejemplo de frases en un archivo .txt
   - **test.json:** Ejemplo de frases en un archivo .json

Todas las funciones creadas se describen a continuación. En el propio código se puede ampliar esta información. Por ejemplo, en cada función se enumeran tanto los parámetros de entrada como los resultados.

1. Archivo *generateGeneralLanguajeModel*
   i. Función general:
   - *generateGeneralLanguajeModel:* Función que crea el modelo de lenguaje general de N-gramas (con N<=5) a partir de un archivo .txt o un array JSON.

   ii. Funciones de apoyo:
   - *replacePunctuation:* Se añade un espacio a los signos de puntuación para asegurar que no forman parte de ninguna palabra, sino que forman una independiente.
   - *replaceToInfinitive:* Función que reemplaza los verbos por su infinitivo.

   iii. Funciones relacionadas con el diccionario de N-gramas
   - *addSentenceToDictionary:* Se añade la frase al modelo de lenguaje general en forma de N-gramas.
   - *updateDict:* Función que actualiza el diccionario tras el procesado de un nuevo N-grama.

   iv. Funciones relacionadas con el cálculo de backoff
   - *generateDictBackoffKatz:* Función que genera el diccionario de parámetros backoff asociados a los bigramas incluídos en el diccionario de N-gramas proporcionado.
   - *addBigramToDictBackoffKatz:* Función que calcula el parámetro de backoff para un bigrama según el método de Katz.
   - *getCKatzTotal:* Función que obtiene el sumatorio de todos los $c_{katz}(w_{i-1}^{i})$ que comparten la misma palabra w_{i-1}.
   - *updateCKatzTotal:* Se actualiza el diccionario de backoff tras el procesado de un nuevo bigrama. Se deben actualizar los valores $\mathrm{c_{katz}}$ y $\mathrm{probBackoff}$ de todos los bigramas que compartan la misma primera palabra, $\mathrm{w_{i-1}}$, que el bigrama procesado.

2. Archivo *viterbiNbest:*
   i. Funciones relacionadas con el archivo Viterbi:
   - *viterbiNbest:* Función que devuelve las listas de palabras más probables para continuar la frase inicial dada.
   - *evaluateNgram:* Función que calcula la probabilidad logarítmica $P(w_n|w_1^{n-1})$.

   ii. Funciones relacionadas con el cálculo de backoff:
   - *calculateBackoffKatz:* Función que calcula la probabilidad de backoff de un bigrama dado.

- *getCKatzNew:* Función que obtiene el valor $c_{katz}(w_{i-1}^{i})$ si se añadiera un nuevo bigrama al diccionario de backoff.

iii. Otras funciones:

- *orderDictByValue*: Función que ordena un diccionario en función de su valor de probabilidad (de mayor a menor).
- *conjugateSentence:* Función que sustituye los verbos (en infinitivo) de una frase dada por su conjugación más probable.
- *convertToJSON:* Función que convierte en objeto JSON una lista de strings.

3. Archivo *calcularPrediccion:*
   i. Función única:
   - *calcularPrediccion:* Función que realiza la predicción a partir de una frase dada y devuelve los resultados en un array JSON. Utiliza diversas funciones creadas en el archivo *viterbiNbest.*

2. MODO DE USO

Para realizar la predicción de N-gramas, se han realizado dos scripts. En el primero, se generan los modelos de lenguaje (total y de infinitivos). En el segundo, se obtienen las predicciones a partir de una frase dada.

a. Script *scriptLanguajeModel*:

En este script se generan los modelos de lenguaje a partir de los archivos especificados. Estos archivos se especifican utilizando dos listas:

- *fileFormat:* lista de los formatos de los archivos
- *file:* archivos con los que generar los modelos de lenguaje

Es imprescindible que el formato de archivo se especifique en la lista *fileFormat* en el mismo orden que se indica el archivo en la lista *file* para que concuerde formato y archivo.

Un ejemplo sería el siguiente:

```
#Primero se inicializan las listas
fileFormat = []
file = []
#Luego se van añadiendo formatos y archivos
        #archivo1
fileFormat.append('txt')
file.append('frasesTest.txt')
        #archivo2
fileFormat.append('json')
file.append('test.json')
```

Los archivos generados son los siguientes:

i. Para el modelo de lenguaje general:
   - *listWords.txt:* Lista de palabras
   - *dictNgram.txt:* Diccionario de N-gramas
   - *dictBackoff.txt:* Diccionario de backoff

ii. Para el modelo de lenguaje de infinitivos:
   listWordsInf.txt: Lista de palabras
   dictNgramInf.txt: Diccionario de N-gramas
   dictBackoffInf.txt: Diccionario de backoff

b. Script *scriptPredicción*:

Este script es el que obtiene la predicción de los N-gramas que continúan la frase especificada. Como entrada se aplican los archivos generados en el script anterior (*scriptLanguajeModel*).

Además, se debe especificar la frase inicial y los parámetros de máximo orden del N-grama y de máximo número de resultados.

Un ejemplo sería el siguiente:

```
sentence = "Me voy a" #Frase inicial sobre la que calcular la predicción
maxNgram = 5 #Máximo orden del N-grama
maxNumPaths = 5 #Máximo número de resultados
```

Que daría el siguiente resultado al ejecutarse por línea de comandos:

```
Me voy a :  {'1': 'comprar una camisa', '2': 'ver una película'}
```

Por tanto, el array JSON sería el siguiente:

```
{'1': 'comprar una camisa', '2': 'ver una película'}
```

Así, la frase con mayor probabilidad es *'comprar una camisa'* y la segunda que más *'ver una película'*.