Prácticas de SAR

Sistemas de Almacenamiento y Recuperación de información

Práctica 2: Cuenta Palabras

Cuenta Palabras

Descripción del problema

Para hacer estudios sobre la autoría de unos documentos, se desea obtener estadísticas del estilo literario del autor (centrándonos en el uso del vocabulario, la complejidad estructural y la información contenida).

Requisitos del Proyecto

¿Qué debo hacer?

Escribe un programa en Python que analice ficheros de texto, calcule estadísticas exhaustivas sobre cada fichero y las escriba en disco.

- Recibirá uno o más nombres de fichero obligatoriamente.
- Por cada fichero de texto generará, como mínimo, un fichero con las estadísticas legibles.
- Para realizar el análisis se eliminarán todos los símbolos no alfanuméricos.

¿Qué debo hacer?

El programa aceptará los siguientes argumentos para configurar el análisis:

- `-h`, `--help`: muestra el mensaje de ayuda.
- `-l`, `--lower`: pasa todo el contenido a minúsculas antes de procesar.
- `-s` fichero (también `--stop` fichero): especifica un fichero con las **stopwords** que se deben eliminar.
- `-f`, `--full`: muestra por terminal las estadísticas completas. En caso contrario, solo se muestran las 20 primeras entradas de cada categoría.
- `-b`, `--bigram`: activa el análisis de **bigramas** (de palabras y de letras).
- `-e`, `--entropy`: activa el cálculo de la **Entropía de Shannon** y la redundancia.
- `-j`, `--json`: exporta los resultados en formato **JSON** en lugar del formato de texto plano.

¿Qué debo hacer?

Ayuda del programa (`-h`)

```
prompt> python SAR_p2_cuenta_palabras.py --help
usage: SAR_p2_cuenta_palabras.py [-h] [-s STOPWORDS] [-l] [-b] [-f] [-e] [-j
     ]
                                 file [file ...]

Compute comprehensive statistics from text files.

positional arguments:
  file                  text file to analyze.

optional arguments:
  -h, --help            show this help message and exit
  -s STOPWORDS, --stop STOPWORDS
                        filename with the stopwords.
  -l, --lower           lowercase all words before computing stats.
  -b, --bigram          compute bigram stats.
  -f, --full            show full stats (instead of top 20).
  -e, --entropy         compute Shannon entropy and redundancy.
  -j, --json            output statistics in JSON format.
```

Estadísticas a calcular (I)

Para cada fichero, el programa debe obtener y mostrar:

1. **Métricas Básicas:**

- Nº líneas.
- Nº palabras totales (incluyendo y excluyendo stopwords si aplica).
- **Vocabulario:** nº palabras distintas.
- **Símbolos:** nº caracteres totales y nº caracteres distintos.

2. **Distribuciones de Frecuencia:**

- Frecuencia de cada palabra (orden alfabético y por frecuencia).
- Frecuencia de cada letra (orden alfabético y por frecuencia).

Nota: Para el análisis se eliminarán todos los símbolos no alfanuméricos mediante `\W+`.

Estadísticas a calcular (II)

3. **Análisis de Bigramas (con `-b`):**

- **Bigramas de palabras:** pares de palabras consecutivas. Se considera cada línea como una frase independiente, añadiendo el símbolo `'$'` al inicio y al final de cada una.
- **Bigramas de letras:** pares de letras consecutivas que aparecen dentro de cada palabra.

Estadísticas a calcular (III)

1. **Análisis de Información (con -e):**

- **Entropía de Shannon (H):** mide la cantidad media de información por **símbolo**.

$$H(X) = -\sum_{i=1}^{n} P(x_i) \log_2 P(x_i)$$

donde n es el número de símbolos (letras) distintos en el texto; x_i es un símbolo y $P(x_i)$ es la probabilidad de aparición de x_i.

$$P(x_i) = \frac{\text{número de apariciones de } x_i}{\text{número total de símbolos}}$$

Nota

Por simplicidad en el cálculo de la entropía NO se considerará el espacio como un símbolo.

Estadísticas a calcular (IV)

- **Redundancia (R):** mide el grado de predictibilidad del texto.

La Redundancia (R) indica cuánto se podría comprimir el texto sin perder información y se calcula como:

$$R = 1 - \frac{H}{H_{max}}$$

donde H es la Entropía y H_{max} es la Entropía Máxima teórica calculada como:

$$H_{max} = \log_2(n)$$

¿Qué debo hacer?

Ejemplo de salida texto

```
Lines: 11
Number words (including stopwords): 77
Vocabulary size: 22
Number of symbols: 324
Number of different symbols: 23
Shannon entropy: 3.8114 bits/symbol
Redundancy: 15.74%
Words (alphabetical order):
    a: 2
    and: 12
    aux: 1
...
    spam: 27
    thermidor: 1
    top: 1
Words (by frequency):
    spam: 27
    and: 12
    egg: 9
...
    thermidor: 1
```

¿Qué debo hacer?

Ejemplo de salida texto (continuación)

```
Symbols (alphabetical order):
    a: 63
    b: 10
    c: 8
    d: 17
...
    s: 40
    t: 9
    u: 7
    v: 1
Symbols (by frequency):
    a: 63
    s: 40
    m: 29
    p: 29
...
    f: 3
    l: 2
    w: 2
    y: 2
```

¿Qué debo hacer?

Ejemplo de salida json


```
{
    "metadata": {
        "source_file": "spam.txt",
        "options": {
            "lower": false,
            "stopwords": false,
            "bigrams": false,
            "entropy": true
        }
    },
    "basic_stats": {
        "lines": 11,
        "words": 77,
        "vocab_size": 22,
        "symbols": 324,
        "unique_symbols": 23
    },
```

¿Qué debo hacer?

Ejemplo de salida json (continuación)


```
    "entropy_analysis": {
          "shannon_entropy": 3.811382500018934,
          "redundancy": 0.15743775877425015
      },
      "top_words": {
          "spam": 27,
          "and": 12,
          "egg": 9,
          "bacon": 6,
...
      },
      "top_symbols": {
          "a": 63,
          "s": 40,
          "m": 29,
          "p": 29,
...
      }
```

Nombre de los ficheros

El programa generará automáticamente el nombre del fichero de salida basándose en las opciones activadas. Se añadirá un sufijo al nombre original (separado por _) con los códigos de las opciones elegidas en este orden: **l**, **s**, **b**, **f**, **e**, **j**.

Ejemplos

- `python SAR_p2_cuenta_palabras.py spam.txt -l -b`
  generará el fichero `spam_lb_stats.txt`
- `python SAR_p2_cuenta_palabras.py spam.txt -l -j`
  generará el fichero `texto_lj_stats.json`
- `python SAR_p2_cuenta_palabras.py spam`
  generará el fichero `spam_stats.txt`

Plantilla

Plantilla: main (I)

```
if __name__ == "__main__":

    parser = argparse.ArgumentParser(description='Compute comprehensive
     statistics from text files.')
    parser.add_argument('file', metavar='file', type=str, nargs='+',
                        help='text files to analyze.')

    parser.add_argument('-l', '--lower', dest='lower',
                        action='store_true', default=False,
                        help='lowercase all words before computing stats.')

    parser.add_argument('-s', '--stop', dest='stopwords', action='store',
                        help='filename with the stopwords.')

    parser.add_argument('-b', '--bigram', dest='bigram',
                        action='store_true', default=False,
                        help='compute bigram stats.')

    parser.add_argument('-f', '--full', dest='full',
                        action='store_true', default=False,
                        help='show full stats (instead of top 20).')
```

Plantilla: main (II)

```
parser.add_argument('-e', '--entropy', dest='entropy',
                    action='store_true', default=False,
                    help='compute Shannon entropy and redundancy.')

parser.add_argument('-j', '--json', dest='json',
                    action='store_true', default=False,
                    help='output statistics in JSON format.')

args = parser.parse_args()
wc = WordCounter()
wc.compute_files(args.file,
                 lower=args.lower,
                 stopwordsfile=args.stopwords,
                 bigrams=args.bigram,
                 full=args.full,
                 entropy=args.entropy,
                 use_json=args.json)
```

Plantilla: WordCounter (I)

```
class WordCounter:

    def __init__(self):
        """
           Constructor de la clase WordCounter
        """
        self.clean_re = re.compile('\W+')

    def write_stats_text(self, filename, stats, use_stopwords, full):
        """
        Este método escribe en texto plano las estadísticas de un fichero

        :param
            filename: el nombre del fichero destino.
            stats: las estadísticas del texto.
            use_stopwords: booleano, si se han utilizado stopwords
            full: boolean, si se deben mostrar las stats completas

        :return: None
        """

        with open(filename, 'w', encoding='utf-8') as fh:
            # COMPLETAR
            pass
```

Plantilla: WordCounter (II)

```
    def write_stats_json(self, filename, source_file, stats, use_stopwords,
     full):
        """
        Este método escribe en formato JSON las estadísticas de un fichero
        :param
            filename: el nombre del fichero destino.
            source_file: el nombre del fichero fuente.
            stats: las estadísticas del texto.
            use_stopwords: booleano, si se han utilizado stopwords
            full: boolean, si se deben mostrar las stats completas

        :return: None
        """
        js = {
            "metadata": {
                "source_file": source_file,
                "options": {
                    "lower": use_stopwords,
                    "stopwords": use_stopwords,
                    "bigrams": 'biword' in stats,
                    "entropy": 'entropy' in stats
                }
            }
        }
        # COMPLETAR
```

Plantilla: WordCounter (III)

```
def file_stats(self, filename, lower, stopwordsfile, bigrams, full,
 entropy, use_json):
    """
    Este método calcula las estadísticas de un fichero de texto

    :param
        filename: el nombre del fichero.
        lower: booleano, se debe pasar todo a minúsculas?
        stopwordsfile: nombre del fichero con las stopwords o None si no
  se aplican
        bigram: booleano, se deben calcular bigramas?
        full: booleano, se deben montrar la estadísticas completas?
        entropy: booleano, se debe calcular la entropía de Shannon?
        use_json: booleano, se debe mostrar las estadísticas en formato
 JSON?
    :return: None
    """

    stopwords = [] if stopwordsfile is None else open(stopwordsfile,
 encoding='utf-8').read().split()

    # variables for results
    sts = {
            'nwords': 0,
```

Plantilla: WordCounter (IV)

```
    # variables for results
    sts = {
            'nwords': 0,
            'nlines': 0,
            'word': {},
            'symbol': {},
            }

    if bigrams:
        sts['biword'] = {}
        sts['bisymbol'] = {}

    if entropy:
        sts['entropy'] = None
        sts['redundancy'] = None

    # COMPLETAR
    new_filename = filename # CAMBIAR
    if use_json:
        self.write_stats_json(new_filename, filename, sts, stopwordsfile
 is not None, full)
    else:
        self.write_stats_text(new_filename, sts, stopwordsfile is not
None, full)
```

Plantilla: WordCounter (V)

```
def compute_files(self, filenames, **args):
    """
    Este método calcula las estadísticas de una lista de ficheros de
 texto

    :param
        filenames: lista con los nombre de los ficheros.
        args: argumentos que se pasan a "file_stats".

    :return: None
    """

    for filename in filenames:
        self.file_stats(filename, **args)
```