TP02 - Analisis de Texto

Consignas y Respuestas

La instrucciones de como ejecutar cada punto se detallan en el README.md dentro de cada punto

1. Escriba un programa que realice análisis léxico sobre la colección RI-tknz-data. El programa debe recibir como parámetros el directorio donde se encuentran los documentos y un argumento que indica si se debe eliminar las palabras vacías (y en tal caso, el nombre del archivo que las contiene). Defina, además, una longitud mínima y máxima para los términos.

A continuación se detallan los resultados obtenidos. [VER CÓDIGO] (Carpeta: '1/')

Términos

La lista de terminos se ha dejado este archivo (1/terms.json). Debido a su tamaño no la presentamos en formato tabla.

Estadísticas

Stat	Qty
qty_docs	499
qty_tokens	434582
qty_terms	32244
avg_qty_tokens	870.9058116232464
avg_qty_terms	64.61723446893788,
avg_len_term	8.597351445230121
min_tokens_file	1
max_tokens_file	138
min_terms_file	0
max_terms_file	58
max_terms_file	16637

Ver archivo de stats ('1/stats.json')

La lista de los 10 términos más frecuentes y su CF

Termino	que	del	los	las	para	universidad	por	con	una	nacional	facultad
CF	4652	4518	3725	2990	2791	2243	2099	2045	1609	1444	1248

La lista de los 10 términos menos frecuentes y su CF

Termino	0108	426759	dptomat@unsl	uma	lix	xxxii	abel	gromov	natuales	universadad	calido
CF	1	1	1	1	1	1	1	1	1	1	1

Ver archivo de frecuencias ('1/frecuencies.json')

2. Tomando como base el programa anterior, escriba un segundo Tokenizer que implemente los criterios del artículo de Grefenstette y Tapanainen para definir que es una palabra" (o término) y cómo tratar números y signos de puntuación. Además, extraiga en listas separadas utilizando en cada caso una función especifica

A continuación se destacan las expresiones regulares en forma de diccionario. [VER CÓDIGO] (Carpeta: '2/')

```
PRE_RE_PATTERNS = {
   'mail': '[a-zA-Z0-9_\-]+@[a-zA-Z0-9]+(?:\.[a-zA-Z0-9]+)+',
   'url' : '(?:https:\/\/|http:\/\/){0,1}[a-zA-Z0-9\-]+ \
       (?:\.[a-zA-Z0-9]+(?:\-[a-zA-Z0-9]+){0,1})+ \
       (?:\/[a-zA-Z0-9]+(?:\-[a-zA-Z0-9]+){0,1})*(?:\?[a-zA-Z0-9\-\=\+\&]+){0,1}',
   'abbr': '(?=\s)[A-Za-z]\.(?=[A-Za-z0-9]\.)*(?=(?=\,|\?|\s[a-z0-9]))',
   'real': '[0-9]+\.[0-9]+'
}
POST_RE_PATTERNS = {
   'cel' : '[0-9]+(?:\-[0-9]+)+',
   'name': '[A-Z][a-z]+(?:(?:\s*|\s*\n\s*)[A-Z][a-z]+)*',
   'qty' : '[0-9]+',
   'word': '[a-zA-Z]+'
}
```

A cada termino se le agrega el tipo que RE con la que se proceso. Ver archivo de terminos (Carpeta: '2/terms.json')

3. Repita el procesamiento del ejercicio 1 utilizando la colección RI-tknz-qm. Verifique los resultados e indique las reglas que debería modificar para que el tokenizer responda al dominio del problema.

Para lograr la tokenización de los elemtos químicos y formulas en este punto agregamos las siguientes expresiones regulares:

```
{
  ...
  'formula_quim': '(?:[0-9]*[A-Z]+[a-z]*[0-9]*)+ \
    (?:\s[\+|\-]+\s(?:[0-9]*[A-Z]+[a-z]*[0-9]*)+)+',
  'elem_quim': '(?:[0-9]*[A-Z]+[a-z]*[0-9]*)+',
}
```

[VER CÓDIGO] (Carpeta: '3/')

4. A partir del programa del ejercicio 1, incluya un proceso de stemming. Luego de modificar su programa, corra nuevamente el proceso del ejercicio 1 y analice los cambios en la colección. ¿Qué implica este resultado? Busque ejemplos de pares de términos que tienen la misma raíz pero que el stemmer los trató diferente y términos que son diferentes y se los trató igual.

A continuación se destacan algunos terminos que fueron tratados diferente.

Raíz Esperada	Raíz resultante	Termino
univers	univers	universidad
univers	universitari	universitario
academ	academi	academia

[VER CÓDIGO] (Carpeta: '4/')

5. Sobre la colección CISI, ejecute los stemmers de Porter y Lancaster provistos en el módulo nltk.stem. Compare: cantidad de tokens resultantes, resultado 1 a 1 y tiempo de ejecución para toda la colección. ¿Qué conclusiones puede obtener de la ejecución de uno y otro?

A continuación se detallan las comporativas entre ambos stemmers

Tiempo de Ejecución

- **Porter** - tardó 14.943629026412964 segundos.
- **Lancaster** - tardó 11.641700029373169 segundos.

Terminos resultantes

	Lancaster	Porter
qty_terms	25215	27421
avg_qty_terms	50.53	54.95
avg_len_term	8.22	8.6
min_terms_file	0	0
max_terms_file	2990	2
qty_terms_one_appeareance	12616	13782

Terminos 1 a 1 (Top 10)

Lancaster	CF	Porter	CF
que	4657	que	4657
del	4538	del	4521
los	3728	para	2799
las	3001	universidad	2437
par	2868	por	2147
universidad	2437	con	2052
por	2148	una	1617
con	2052	nacion	1546
nac	1715	ciencia	1430
est	1606	facultad	1380
facultad	1382	carrera	1350

[VER CÓDIGO] (Carpeta: '5/')

Conclusiones

Dada la cantidad de terminos resultates y la comparativa del top 10, podemos ver que Lancaster es mas agresivo que Porter en cuanto al largo de las palabras (por ejemplo *nacion*, en porter fue tratada como tal pero el Lancaster se la trató como *nac*). A raíz del ejemplo anterior podemos observar que Lancaster no es muy preciso con la palabras cortas y pierde distinción entre los terminos al utilizar reducciones de los tokens (En Lancaster: *nac* tiene frecuencia 1715, y Porter: *nacion* tiene frecuencia 1546, es decir que hay 169 terminos que fueron tratados como *nac* pero no eran *nacion*, por lo tanto perdimos variedad de terminos).

Por otra parte utilizando Porter podemos observar que no se contemplaron como terminos palabras cortas como: *las* o *los* . Por último en cuanto a velocidad de ejecución podemos observar una mejora de Porter sobre Lancaster.

6. Escriba un programa que realice la identificación del lenguaje de un texto a partir de un conjunto de entrenamiento7. Pruebe dos métodos sencillos:

a. Uno basado en la distribución de la frecuencia de las letras.
b. El segundo, basado en calcular la probabilidad de que una letra x preceda a una y (calcule una matriz de probabilidades con todas las combinaciones).

Compare los resultados contra el módulo Python langdetect y la solución provista.

Comparación de soluciones

Lang Detector	Clasificados Correct. / Total	Accuracy
A.	257/300	85.67%
B.	292/300	97.33%
Con **langdetect**	295/300	98.33%

[VER CÓDIGO - 6.a | VER CÓDIGO - 6.b | VER CÓDIGO - 6.c] (Carpeta: '6/')

Propiedades del Texto

7. En este ejercicio se propone verificar la predicción de ley de Zipf. Para ello, descargue desde Project Gutenberg el texto del Quijote de Cervantes10 y escriba un programa que extraiga los términos y calcule sus frecuencias. Calcule la curva de ajuste utilizando la función Polyt del módulo NymPy11. Con los datos crudos y los estimados grafique en la notebook ambas distribuciones (haga 2 gráficos, uno en escala lineal y otro en log-log). ¿Cómo se comporta la predicción? ¿Qué conclusiones puede obtener?

8. Usando los datos del ejercicios anterior y de acuerdo a la ley de Zipf, calcule la proporción del total de términos para aquellos que tienen frecuencia f = {100; 1000; 10000}. Verifique respecto de los valores reales. ¿Qué conclusión puede obtener?

9. Para el texto del ejercicio 5 procese cada palabra en orden y calcule los pares (cant. términos totales procesados, cant. términos únicos). Verifique en que medida satisface la ley de Heaps. Grafique en la notebook los ajustes variando los parámetros de la expresión.

VER NOTEBOOK - 7, 8 y 9 (Carpeta: '7,8,9/')