Trabajo Práctico

Tratamiento y Análisis del Texto

Bibliografía: [MIR] Capítulo 7, [TOL] Capítulo 3, [MAN] Capítulos 6 (parcial).
Paper: "What is a word, What is a sentence? Problems of Tokenization", Gregory Grefenstette , Pasi Tapanainen.
http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.28.5162&rep=rep1&type=pdf

1) Escriba un programa que realice operaciones simples de análisis léxico sobre la colección T12012-gr y calcule medidas básicas sobre la misma. Implemente los criterios del artículo de Grefenstette y Tapanainen para definir qué es una "palabra" (o término) y cómo tratar números y signos de puntuación. Trabaje con y sin eliminar palabras vacías y defina longitud mínima y máxima para los términos. Como salida, el programa debe generar:

   a) Un archivo (terminos.txt) con la lista de términos a indexar (ordenado), su frecuencia en la colección y su DF.

   b) Un segundo archivo (estadisticas.txt) con los siguientes datos:
      Cantidad de documentos procesados
      Cantidad de *tokens* y términos extraídos
      Promedio de *tokens* y términos de un documento
      Largo promedio de un término
      Cantidad de *tokens* y términos del documento más corto y del más largo
      Cantidad de términos que aparecen sólo 1 vez en la colección

   c) Un tercer archivo con:
      La lista de los 10 términos más frecuentes (y su TF)
      La lista de los 10 términos menos frecuentes (y su TF)

   Explique para qué utilizaría la información extraída.

2) Tomando como base su programa anterior, escriba un Tokenizer que cumpla con los siguientes requisitos:
   - Extraiga las abreviaturas tal cual están escritas (por ejemplo, Dr., Lic., S.A.,
   - NASA, etc.) y las trate como tokens.
   - Mantenga las formas unidas por guiones como un único token (por ej., Iran-Iraq)
   - Extraiga direcciones de correo electrónico
   - Extraiga URLs
   - Extraiga números (por ejemplo, cantidades, teléfonos)
   - Extraiga nombres propios (por ejemplo, Villa Carlos Paz, Manuel Belgrano, etc.) y los trate como un único token.

3) Repita el procesamiento del ejercicio 1 utilizando la colección T12012-qm. Verifique los resultados e indique las reglas que debería modificar para que el *tokenizer* responda al dominio del problema.

4) A partir de su programa del ejercicio 1, incluya un proceso de *stemming*. Implemente su solución con Snowball[1] ó con la versión que se encuentra en Sourceforge[2]. 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

1 http://snowball.tartarus.org/algorithms/spanish/stemmer.html
2 http://stemmer-es.sourceforge.net/

tienen la misma raíz pero que el *stemmer* los trató diferente y términos que son diferentes y se los trató igual.

5) En este ejercicio se propone verificar la predicción de ley de Zipf. Para ello, descargue desde Project Gutenberg[3] el texto del Quijote de Cervantes y escriba un programa que extraiga los términos y calcule las frecuencias. Con dichos datos y los estimados por Zipf grafique ambas distribuciones (haga 2 gráficos, uno en escala lineal y otro en log-log). ¿Cómo se comporta la predicción? ¿Qué conclusiones puede obtener? Repita el análisis podando un porcentaje x (use x = 5, 10 y 15%) de los términos más y menos frecuentes. ¿Con qué porcentaje de poda se mejora la predicción para este texto?

6) Suponga que tiene que construir un índice para recuperación y decide omitir aquellos términos cuya frecuencia es menor a 5. De acuerdo a la ley de Zipf, qué proporción del total de términos estaría omitiendo? Justifique. ¿Qué proporción está realmente omitiendo si indexa el texto del ejercicio anterior?

7) Para el texto del ejercicio 7 procese cada palabra en orden y calcule los pares (#palabras procesadas, #términos únicos vistos). Verifique si satisface la ley de Heaps.

3 http://www.gutenberg.org/dirs/etext99/2donq10.zip