Proyecto ALT (Tarea 3)

Recordatorio del objetivo del proyecto: "crear un sistema eficiente para realizar la búsqueda aproximada de una cadena respecto de todas las cadenas del diccionario y utilizar esa búsqueda aproximada para ampliar el motor de recuperación de información desarrollado en SAR de forma que acepte consultas con búsqueda aproximada."

Una vez completadas las dos tareas anteriores, ya disponéis de implementaciones para calcular distancias de edición entre dos cadenas (Levenshtein, Damerau restringido y Damerau intermedio). Por tanto, si queremos calcular las palabras, de un diccionario dado, próximas ortográficamente a un término escrito por el usuario, no tendremos más remedio que calcular la distancia entre dicho término y cada una de las palabras del diccionario. En esta tarea se pide que, siguiendo esta estrategia de búsqueda "término contra todos", evitéis la ejecución de la distancia de edición entre aquellas cadenas en las que algún tipo de cota optimista nos permita saber que dicha distancia será mayor al umbral establecido.

En cualquier caso, necesitáis un **diccionario** de términos y **resultados de referencia** para comprobar que vuestras implementaciones son correctas!. El propósito de este README es detallar los aspectos técnicos para hacer esto.

Obtención del diccionario

Para generar el diccionario se ha dejado el fichero *quijote.txt* en la sección "Material para la realización del proyecto" del apartado "Prácticas" en PoliformaT. Se trata de, cargar el fichero *quijote.txt*, limpiar el texto (convertir a minúsculas y tokenizar) y extraer el vocabulario (palabras únicas). Para hacer esto, se os proporciona el siguiente código, con el objetivo de que todos los alumnos compartáis el mismo preproceso y que los resultados obtenidos sean comparables.

```
def build_vocab(vocab_file):
    clean_re = re.compile('\W+')
    with open(vocab_file, "r") as fr:
        vocab = set(clean_re.split(fr.read().lower()))
    return vocab
```

Podéis integrar esta función como un método en la clase Python a diseñar, para obtener el diccionario de términos que debe utilizar el método *suggest* de dicha clase (ver trasparencia 44 del boletín).

Evaluación

Para que evaluéis vuestras implementaciones de las distancias de edición, se os proporcionan los ficheros *result_levenshtein_quijote.txt*, *result_restricted_damerau-levenshtein_quijote.txt* y *result_intermediate_damerau-levenshtein_quijote.txt*,

cuyas líneas contienen el término buscado, el threshold usado, el número de resultados obtenido tras la búsqueda y una enumeración de las palabras más similares ortográficamente al término buscado. Cada uno de los elementos está separado del resto mediante tabuladores. A continuación se muestran dos líneas de ejemplo extraídas del fichero *result_restricted_damerau-levenshtein_quijote.txt*:

```
casa    1   31  0:casa 1:basa 1:caba 1:cada 1:caja 1:cala 1:cama ...
quixot  2   6   2:quieto 2:quijo 2:quijote 2:quinto 2:quiso 2:quito
```

Los términos que incluyen los ficheros son: “casa”, “senor”, “jabón”, “constitución”, “savaedra”, “vicios”, “quixot”, “s3afg4ew” y “ancho”, con thresholds desde 1 hasta 5. No es necesario que automaticéis la evaluación (aunque sí recomendable), es suficiente con probar a mano varios de estos ejemplos con varios thresholds y ver que los resultados coinciden con los ficheros de resultados proporcionados.