Tarea 1 – ISIS 4221

Procesamiento de Lenguaje Natural

Fecha de entrega: miercoles 27 de Agosto 6pm. (No se permiten entregas tardías – la plataforma se cierra automáticamente)

Reglas de codificación: use jupyter notebooks (python) y asegúrese de que el notebook se ejecute y contenga los resultados antes de enviarlo. Todas las clases, métodos, funciones y código libre DEBEN contener docstrings con una explicación detallada. Especifique en que líneas se debe cambiar las rutas para la lectura de archivos. Construya un notebook para cada punto.

Informe: Junto con los notebooks, deberá presentar un informe escrito (utilice formato pdf) con las respuestas a las preguntas y un breve resumen de la implementación.

Envío: *individual o en grupo de tres máximo* las tareas se envían a través de Bloque neón. Haga un zip que incluya todos los archivos. SOLO UNO HACE LA ENTREGA.

[10p] Implemente las siguientes métricas de evaluación de IR usando python+numpy (debe usar numpy):

- [1p] Precision (la relevancia es binaria)

```
>>> relevance_query_1 = [0, 0, 0, 1]
>>> precision(relevance_query_1)
0.25
```

- [1p] Precision at K (la relevancia es binaria)

```
>>> relevance_query_1 = [0, 0, 0, 1]
>>> k = 1
>>> precision_at_k(relevance_query_1, k)
0
```

- [1p] Recall at K (la relevancia es binaria)

```
>>> relevance_query_1 = [0, 0, 0, 1]
>>> k = 1
>>> number_relevant_docs = 4
>>> recall_at_k(relevance_query_1, number_relevant_docs, k)
0
```

- [1p] Average precision (la relevancia es binaria)
  - Suponga que el vector binario de entrada contiene todos los documentos relevantes.

```
>>> relevance_query_2 = [0,1,0,1,1,1,1]
>>> average_precision(relevance_query_2)
0.5961904
```

- [2p] Mean average precision -MAP- (la relevancia es binaria)
  - Entrada: una lista de vectores binarios, cada uno representa un vector de resultado de la consulta.
- [2p] DCG at K (la relevancia es un número natural)

```
>>>relevance_query_3  = [4, 4, 3, 0, 0, 1, 3, 3, 3, 0]
>>>k = 6
>>>dcg_at_k(relevance_query_3, k)
10.27964
```

- [2p] NDCG at K (la relevancia es un número natural)

```
>>>relevance_query_3  = [4, 4, 3, 0, 0, 1, 3, 3, 3, 0]
>>>k = 6
>>>ndcg_at_k(relevance_query_3, k)
0.7424
```

Comparación de estrategias de motores de búsqueda

A continuación, implementará un motor de búsqueda con cuatro estrategias diferentes.

1. Búsqueda binaria usando índice invertido (BSII)

3. Recuperación ranqueada y vectorización de documentos (RRDV)

Debe hacer su propia implementación usando numpy y pandas.

Conjunto de datos: hay tres archivos que componen el conjunto de datos.

- “Docs raws texts" contiene 331 documentos en formato NAF (XML; debe usar el título y el contenido para modelar cada documento).
- "Queries raw texts" contiene 35 consultas.
- "relevance-judgments.tsv" contiene para cada consulta los documentos considerados relevantes para cada una de las consultas. Estos documentos relevantes fueron construidos manualmente por jueces humanos y sirven como “ground-truth” y evaluación.

Pasos de preprocesamiento: para los siguientes puntos, debe preprocesar documentos y consultas mediante tokenización a nivel de palabra, eliminación de palabras vacías, normalización y stemming.

[25p] Búsqueda binaria usando índice invertido (BSII)

[10p] Cree su propia implementación del índice invertido usando los 331 documentos en el conjunto de datos.

[10p] Cree una función que lea el índice invertido y calcule consultas booleanas mediante el algoritmo de mezcla. El algoritmo de mezcla debe ser capaz de calcular: AND, y NOT.

[5p] Para cada una de las 35 consultas en el conjunto de datos, recupere los documentos utilizando consultas binarias AND (i.e. termino_1 AND termino_2 AND termino_3…). Escriba un archivo (BSII-AND-queries_results) con los resultados siguiendo el mismo formato que "relevance-judgments":

q01 dXX,dYY,dZZ…

Nota: pueden resultar archivos vacíos.

Nota: pueden resultar archivos vacíos.

[35p] Recuperación ranqueada y vectorización de documentos (RRDV)

[10p] Cree una función que, a partir del índice invertido, cree la representación vectorial ponderada tf.idf de un documento o consulta. Describa en detalle su estrategia, ¿es eficiente? ¿por qué si, por qué no?

[10p] Cree una función que reciba dos vectores de documentos y calcule la similitud del coseno.

[5p] Para cada una de las 35 consultas en el conjunto de datos, recupere los documentos clasificados -ordenados por el puntaje de similitud del coseno- (incluya solo los documentos con un puntaje superior a 0 para una consulta determinada). Escriba un archivo (RRDV-consultas_resultados) con los resultados siguiendo el siguiente formato:

q01 dXX: cos_simi(q01,dXX),dYY: cos_simi(q01, dYY),dZZ: cos_simi(q01,dZZ)…

[10p] Evaluación de resultados. Calcule P@M, R@M, NDCG@M por consulta. M es el número de documentos relevantes encontrados en el archivo de juicios de relevancia por consulta. Luego calcule MAP como una métrica general.

NOTA I: Para P@M y R@M suponga una escala de relevancia binaria. Los documentos que no se encuentran en el archivo "relevance-judgments" NO son relevantes para una consulta determinada.

NOTA II: Para NDCG@M utilice la escala de relevancia no binaria que se encuentra en el archivo "relevance-judgments".

[30] Repita el punto anterior (RRDV) pero utilizando GENSIM (use como nombre de archivo GESIM-consultas_resultados).