**Informe de Trabajo en Grupo: Proyecto PageRank para Data Mining**

**Desarrollado por:**

* **Johana Duchi**
* **Daniela Jiménez**
* **Alex Pérez (Líder del equipo)**

**Introducción**

El proyecto presente implementa el algoritmo **PageRank** utilizando el enfoque de caminata aleatoria para calcular la importancia relativa de los nodos en un grafo dirigido. A través de la simulación de random walkers y la introducción de **spider traps** y **dead ends (en dos grafos independientes)**, se busca monitorear cómo estos afectan la convergencia del algoritmo y la distribución de probabilidad sobre los nodos del grafo.

**Desarrollo del Proyecto**

El líder de grupo distribuyó las tareas de manera equitativa, cada miembro contribuyó de manera significativa al desarrollo del sistema PageRank. A continuación, se describe la participación de cada integrante en el proyecto:

**Johana Duchi - Implementación de la Estructura del Grafo y Matrices de Adyacencia**

**Funciones Implementadas:**

* **generate\_graph\_with\_nodes()**
* **add\_random\_connections()**
* **adjacency\_matrix()**

**Descripción del Desarrollo:**

Se ecnargó de la implementación de las funciones relacionadas con la creación y configuración del grafo, así como la transformación de este en una **matriz de adyacencia estocástica**. Inicialmente, se generan nodos etiquetados de manera alfabética, y se definen conexiones aleatorias entre ellos.

* **generate\_graph\_with\_nodes()**: Genera el grafo dirigido con nodos y los conecta de manera aleatoria.
* **add\_random\_connections()**: Agrega conexiones adicionales entre los nodos para simular un sistema de enlaces similar al de la web.
* **adjacency\_matrix()**: Convierte el grafo en una matriz de adyacencia estocástica, donde cada fila representa las probabilidades de transición entre nodos.

**Herramientas Utilizadas:**

* **NetworkX**: Para la construcción del grafo dirigido.
* **NumPy**: Para realizar las operaciones de normalización de la matriz.
* **Pandas**: Para transformar las matrices de adyacencia en estructuras de datos legibles (DataFrame).

**Daniela Jiménez - Simulación del Algoritmo PageRank, Caminata Aleatoria y visualización de Grafo para Dead Ends**

**Funciones Implementadas:**

* **power\_iteration\_trace()**
* **introduce\_dead\_ends()**
* **walker\_simulation()**
* **display\_graph()**

**Descripción del Desarrollo:**

Responsible de implementar las simulaciones de **caminatas aleatorias** sobre el grafo, así como la implementación del algoritmo de **PageRank** utilizando la técnica de **iteración de potencias**. Estas simulaciones permiten estudiar cómo la importancia de cada nodo evoluciona a lo largo del tiempo. Implementó la posibilidad de agregar **dead ends**, los cuales representan nodos sin ninguna conexión saliente.

* **power\_iteration\_trace()**: Implementa el algoritmo de PageRank utilizando iteración de potencias, normalizando el vector de importancia de los nodos en cada iteración. Adicionalmente, guarda el progreso de las iteraciones para su análisis posterior.
* **introduce\_dead\_ends()**: Introduce nodos que no tienen conexiones salientes.
* **walker\_simulation()**: Simula un caminante aleatorio que recorre los nodos del grafo. En esta simulación, si el caminante se encuentra con un dead end, puede optar por teletransportarse a otro nodo aleatorio no visitado.
* **display\_graph()**: Visualiza el grafo utilizando la biblioteca **Matplotlib**, asignando diferentes colores a los nodos de acuerdo con su tipo (dead end, spider trap, o nodo regular).

**Herramientas Utilizadas:**

* **NumPy**: Para realizar operaciones matriciales necesarias en la iteración de potencias.
* **Random**: Para seleccionar de manera aleatoria el siguiente nodo a visitar en la simulación de caminata.
* **Input y Print**: Para la interacción con el usuario durante la simulación.

**Alex Pérez - Visualización del Grafo y Comportamiento de Spider Traps**

**Funciones Implementadas:**

* **introduce\_spider\_traps()**
* **walker\_simulation\_for\_spiders()**
* **add\_random\_connections\_for\_spiders()**
* **display\_graph\_for\_spiders()**

**Descripción del Desarrollo:**

Encargado de la visualización del grafo y de la implementación de **spider traps**, nodos que forman un ciclo cerrado, lo que lograría que el algoritmo de PageRank no converja correctamente. Se creó las funciones para la visualización del grafo con diferentes colores para distinguir los nodos regulares de los que forman parte de un spider trap.

* **introduce\_spider\_traps()**: Introduce spider traps en el grafo, donde un conjunto de nodos se enlazan de manera circular entre ellos.
* **walker\_simulation\_for\_spiders()**: Modifica el comportamiento del caminante para que considere la presencia de **spider traps**. Cuando el caminante entra en un spider trap, puede optar por teletransportarse o permanecer en el ciclo cerrado.
* **add\_random\_connections\_for\_spiders()**: Agrega conexiones aleatorias entre los nodos, excluyendo aquellos que forman parte de spider traps.
* **display\_graph()** y **display\_graph\_for\_spiders()**: Visualiza el grafo utilizando la biblioteca **Matplotlib**, asignando diferentes colores a los nodos de acuerdo con su tipo (dead end, spider trap, o nodo regular).

**Herramientas Utilizadas:**

* **Matplotlib**: Para la visualización gráfica del grafo y sus conexiones.
* **NetworkX**: Para la creación de las estructuras de grafo dirigidas.
* **Random**: Para seleccionar nodos aleatoriamente al momento de crear spider traps y añadir conexiones entre los nodos.

**Calificaciones**

**El líder de grupo asigna las siguientes calificaciones a cada uno de los integrantes según su desempeño en las tareas asignadas:**

* **Johana Duchi – 91/100**
* **Daniela Jiménez – 95/100**
* **Alex Pérez – 91/100**