Informe Técnico

Redes de contenido

A continuación se ofrece una explicación de cada método y herramienta utilizada en el desarrollo de esta aplicación.

Elementos

Clases implementadas

- **Content:** esta clase se utiliza para representar un contenido, cuenta con los atributos `name` (es una lista de nombres asociados al contenido) y `node_description` (es el nombre formal del contenido, se utiliza principalmente para apoyo visual).

  El método `insert_to_distances` reporta la cantidad de veces que se encontró un contenido en una vecindad establecida, esto es utilizado para determinar las relaciones entre contenidos. Se utiliza un diccionario que mapea `Dict[Content, List[int]]`.

```
class Content:
    def __init__(self, *name, node_description = ' ')
    def insert_to_distances(self, key_content, value)
```

- **Graph:** esta clase representa el grafo de relaciones entre los contenidos. Para su implementación se utilizó el módulo `networkx` debido a la gran cantidad de herramientas que otorga para el trabajo con grafos y la compatibilidad que muestra con `gephi` (herramienta utilizada para visualizar el grafo). La clase cuenta con los atributos `pdfs_info` (es la información extraída de los pdfs que se están procesando), `contents` (son los contenidos que se estan procesando), `distance_to_ponderate` (es la distancia máxima que se admite entre dos contenidos para considerar que están relacionados).

  El método `ponderate` se encarga de ponderar las aristas que relacionan contenidos dada la información de los pdfs y la distancia predefinida para que sea válida la relación.

  El método `fill_content_distance` utiliza los métodos `check_next_contents` y `match` para determinar todas las distancias entre los contenidos a analizar.

  El método `check_next_contents` revisa en una vecindad de un contenido dado en busca de nuevas relaciones entre contenidos.

El método `match` dado un contenido (`name_content_to_match`), y un grupo de palabras (`words`) comprueba si el grupo de palabras corresponde al contenido dado.

```
class Graph:
    def __init__(self, pdfs_info, contents, distance_to_ponderate)
    def ponderate(self, pdfs_info, distance_to_ponderate)
    def fill_content_distance(self, words, distance)
    def check_next_contents(self, main_content, words, index, distance)
    def match(self, words, content_first_word_index, name_content_to_match)
```

Ejecución en archivo main

Para ejecutar el algoritmo de realación lo primero que debe hacerse es definir el conjunto de pdfs a analizar, deben ser almacenados en la carpeta pdfs que se encuentra dentro del proyecto. Luego se deben definir los contenidos a procesar y alamacenarlos en la lista de contenidos `contents` predefinida en el archivo `main.py` . Luego se ejecuta el código con `Ctrl + F5` y se genera un archivo `.gexf` que puede visualizarse y analizarse con la herramienta **Gephi**.