Grafos en YouTube

Objetivo:

- Usar el TDA grafo y sus algoritmos para modelar una red social y obtener información de ella.

En este caso trabajaremos con la red social conformada por los usuarios de YouTube. Los algoritmos que propuso la cátedra para resolver ciertos problemas fueron:

- Random walks: un algoritmo que realiza un camino aleatorio sobre un grafo. Comenzando desde un vértice y se mueve hacia un vecino random, y luego a otro también de manera random. Este proceso continúa hasta que el recorrido tenga un largo *n* prefijado. Por lo tanto, para realizar un random walk es necesario determinar el largo del recorrido, la probabilidad del movimiento hacia vértices vecinos y la cantidad de recorridos a realizar.
- Label programation: Este algoritmo se utilizó para solucionar el problema de saber que comunidades se forman dentro de la red social. Esto es la cantidad de vértices que se encuentran altamente conectados entre sí.
- Entropy walk: una opción alternativa, que con la utilización de random walks, soluciona también el problema de las comunidades.

Para desarrollar este trabajo utilizamos el lenguaje Python, por ser de alto nivel y por ende, más sencillo. El programa se ejecuta por terminal y para ejecuta comandos ingresados por teclado. Los comandos son:

- Similares: Dado un usuario encuentra los que son similares a él. Utilizando Random walk y contando por distintos usuarios visitados, los n mayores serán los similares al usuario origen.
- Recomendados: Similar al comando anterior, con igual implementacion pero de distinta devolución. Como indica el nombre este le devuelve al usuario una recomendación de usuarios parecidos a él.

- Camino: Encuentra el menor camino entre dos usuarios. Como el grafo es conexo puede ser que haya caminos que entre dos vértices que no sean posibles.
- Centralidad: Obtiene los usuarios más centrales de la red, es decir los más influyentes.

  ➔ *Centralidad exacta*: son los vértices que aparecen más veces entre todos los caminos mínimos existentes en el grafo. Por ende, utilizaríamos BFS para solucionar este problema, ya que devuelve un resultado más exacto.

  ➔ *Centralidad con random walks*: realizando muchos caminos largos, de forma aleatoria, comenzando desde cualquier vértice y con el criterio del comando similares, suena razonable pensar que los usuarios que más veces aparezcan entre todos los recorridos, deberían ser los más centrales.

Utilizamos la centralidad que no es exacta, es decir, la que se resuelve con Random Walks. Como bien dice su definición, sonaba razonable que si un usuario aparece muchísimas veces más que otros, sea un vértice influyente. Sin embargo, no es el más certero, aunque nos pareció muy convincente la opción que realizaba este algoritmo.

- Distancias: Dado un usuario, obtiene la cantidad de personajes que se encuentran a cada una de las distancias posibles, considerando las distancias como la cantidad de saltos entre ambos.
- Estadísticas: Notifica las estadíticas de la red. Su cantidad de usuarios, su cantidad de conecciones entre usuarios, un promedio de las mismas y la densidad de la red.
- Comunidades: Utilizando Label o Entropy, nos permite mostrar las comunidades que se encuentren en la red. En este caso se utilizó Label Programation por su buena explicación en la página donde se público el tp.