Artículo de investigación: Exportar de Grafos

Krikor Bisdikian Gazarian, Tecnológico de Monterrey Campus Santa Fe

En el siguiente artículo de investigación se compararán los diferentes métodos para exportar un grafo y maneras de visualizarlo y realizar un análisis de los datos.

CCS Concepts:•**Theory of computation** → *Graph algorithms analysis;*

Additional Key Words and Phrases: Grafo, exportar, importar

ACM Reference Format:
Krikor Bisdikian Gazarian. 2015. Artículo de investigación: Exportar de Grafos. 1, 1, Article 0 (October 2015), 2 pages.
DOI: http://dx.doi.org/10.1145/0000000.0000000

1. INTRODUCTION

Los grafos son estructuras que nos permiten modelar relaciones entre muchos elementos. Hay diferentes tipos de grafos que permiten modelar relaciones más específicas. Pueden ser dirigidos o no dirigidos, esto significa que la relación entre los elementos tiene sentido. Y también pueden tener peso estas relaciones. ‘

2. PASO A PASO

Para importar los datasets y guardarlos en los diferentes formatos se buscó primero en la documentación de SNAP como importar los datasets. Esto es relativamente fácil ya que SNAP incluye una función para hacerlo.

Luego, se buscó en internet ejemplos de ls diferentes formatos. Y por último se hicieron funciones que recorrieran los vértices y las aristas del grafo, lo cual es relativamente fácil ya que SNAP nos da iteradores y estas estructuras se guardan linealmente. Al recorrer las partes del grafo se iban guardando dependiendo del formato visto como ejemplo. Agregando antes el encabezado y los requerimientos necesarios.

3. VENTAJAS Y DESVENTAJAS

3.1. GraphML

Está basado en XML lo cual lo hace fácil de generar. Es de los más usados para almacenar gráficos. Lo malo es que no soporta tanta información.

3.2. GEXF

Es equivalente a GraphML ya que está basado también en XML, aunque es menos utilizado que GraphML.

3.3. GDF

Se utiliza para intercambiar datos en la industria de la navegación automotriz. No está hecho para ser utilizado a gran escala. Se debe convertir a un formato más eficiente para esto.

3.4. JSON Graph Format

No hay un estándar, cada librería genera su propio formato. Esto también es una ventaja ya que es más versátil.

4. COMPLEJIDAD TEMPORAL Y ESPACIAL

La complejidad temporal de los cuatro algoritmos es de O(V+E) ya que recorre todos los vértices y luego todas las aristas. La complejidad espacial es constante.

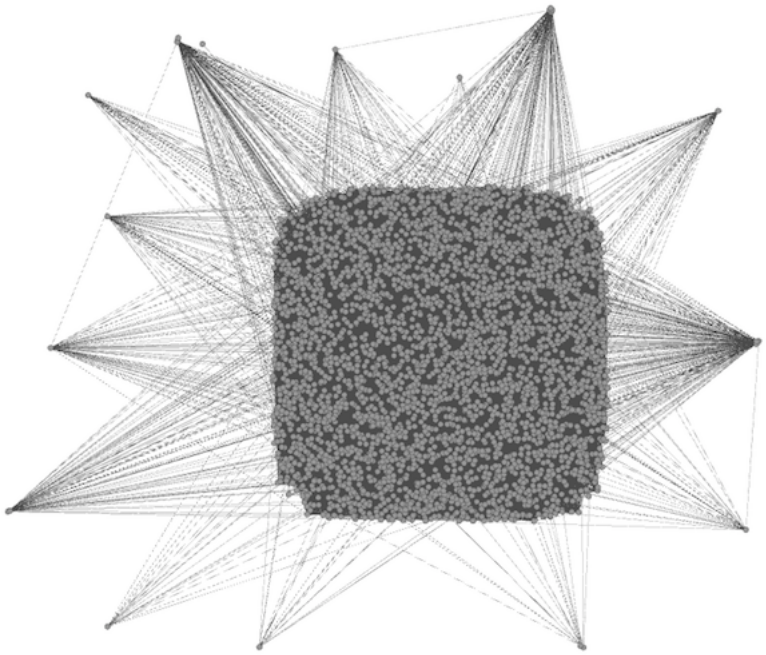

Fig. 1. Visualización grafo en Gephi

5. TIEMPOS DE EJECUCIÓN

— GraphML: 101
— GEXF: 111
— GDF: 63
— JSON Graph Format: 93

6. GEPHI

Los grafos pueden llegar a ser estructuras muy complejas como para imaginarlas sin tenerlas visualmente. Cuando modelan muchas interacciones entre muchos vértices, como lo son los grafos de redes sociales. Es muy difícil analizarlo sin visualizarla, y dibujarla no tendría sentido. Gephi nos proporciona una herramienta en la que podemos interactuar con nuestros grafos de manera gráfica. Aparte nos permite hacer varios análisis sobre el grafo. Uno de ellos, por ejemplo, es la distribución de grado del grafo. Esto no es nada más que cuantas relaciones hay en promedio entre los nodos, es algo tan sencillo como apretar un botón para obtenerse.

En el dataset utilizado que modela las relaciones en Facebook tenemos que la media de la distribución de grado es de 21.843, es decir, en promedio el usuario en esta muestra muy pequeña tiene 22 ”amigos”.

Online Appendix to:
Artículo de investigación: Exportar de Grafos

Krikor Bisdikian Gazarian, Tecnológico de Monterrey Campus Santa Fe

A. FUNCIÓN A GRAPHML

```
void exportGraphML(PNGraph g) {
    std::ofstream file ("./facebook_combined.graphml");
    if (file.is_open())
    {
        file << "<?xml version=\"1.0\" encoding=\"UTF-8\"?>\n";
        file << "<graphml xmlns=\"http://graphml.graphdrawing.org/xmlns\"
        xmlns:xsi=\"http://www.w3.org/2001/XMLSchema-instance\"
        xsi:schemaLocation=\"http://graphml.graphdrawing.org/xmlns
        http://graphml.graphdrawing.org/xmlns/1.0/graphml.xsd\">\n";
        file << "<graph id=\"G\" edgedefault=\"directed\">\n";
        for (PNGraph::TObj::TNodeI NI = g->BegNI(); NI < g->EndNI(); NI++)
        {
            file << "<node id=\"" << NI.GetId() << "\"/>\n";
        }
        int i = 1;
        for (PNGraph::TObj::TEdgeI EI = g->BegEI(); EI < g->EndEI(); EI++, ++i)
        {
            file << "<edge id=\"e" << i << "\" source=\"" << EI.GetSrcNId()
            << "\" target=\"" << EI.GetDstNId() << "\"/>\n";
        }

        file << "</graph>\n";
        file << "</graphml>\n";
        file.close();
    }
}
```

B. FUNCIÓN A GEXF

```
void exportGEXF(PNGraph g)
{
    std::ofstream file ("./facebook_combined.gexf");
    if (file.is_open())
    {
        file << "<?xml version=\"1.0\" encoding=\"UTF-8\"?>\n";
        file << "<gexf xmlns=\"http://www.gexf.net/1.2draft\" version=\"1.2\">\n";
        file << "<graph mode=\"static\" defaultedgetype=\"directed\">\n";
        file << "<nodes>\n";
        for (PNGraph::TObj::TNodeI NI = g->BegNI(); NI < g->EndNI(); NI++)
        {
            file << "<node id=\"" << NI.GetId() << "\" />\n";
```


DOI: http://dx.doi.org/10.1145/0000000.0000000

```
        }
        file << "</nodes>\n";
        file << "<edges>\n";
        int i = 1;
        for (PNGraph::TObj::TEdgeI EI = g->BegEI(); EI < g->EndEI(); EI++, ++i)
        {
            file << "<edge id=\"" << i << "\" source=\"" << EI.GetSrcNId()
            << "\" target=\"" << EI.GetDstNId() << "\" />\n";
        }
        file << "</edges>\n";
        file << "</graph>\n";
        file << "</gexf>\n";
        file.close();
    }
}
```

C. FUNCIÓN A GDF

```
void exportGDF(PNGraph g)
{
    std::ofstream file ("./facebook_combined.gdf");
    if (file.is_open())
    {
        file << "nodedef> name VARCHAR\n";
        for (PNGraph::TObj::TNodeI NI = g->BegNI(); NI < g->EndNI(); NI++)
        {
            file << NI.GetId() << "\n";
        }
        file << "edgedef>source VARCHAR, destination VARCHAR\n";
        for (PNGraph::TObj::TEdgeI EI = g->BegEI(); EI < g->EndEI(); EI++)
        {
            file << EI.GetSrcNId() << ", " << EI.GetDstNId() << "\n";
        }
        file.close();
    }
}
```

D. FUNCIÓN A JSON GRAPH FORMAT

```
void exportJSON(PNGraph g)
{
    std::ofstream file ("./facebook_combined.json");
    if (file.is_open())
    {
        file << "{ \"graph\": {\n";
        file << "\"nodes\": [\n";
        for (PNGraph::TObj::TNodeI NI = g->BegNI(); NI < g->EndNI(); )
        {
            file << "{ \"id\": \"" << NI.GetId() << "\" }";
            if (NI++ == g->EndNI())
            {
                file << " ],\n";
            }
```

```
            else
            {
                file << ",\n";
            }

        }
        file << "\"edges\": [\n";
        for (PNGraph::TObj::TEdgeI EI = g->BegEI(); EI < g->EndEI(); )
        {
            file << "{ \"source\": \"" << EI.GetSrcNId() << "\", \"target\": \""
            << EI.GetDstNId() << "\" }";
            if (EI++ == g->EndEI())
            {
                file << " ]\n";
            }
            else
            {
                file << ",\n";
            }
        }
        file << "} }";
        file.close();
    }
}
```