Procesamiento De Grafos

Saúl Montes De Oca Martínez

30 de Octubre de 2019

Abstract

El siguiente documento tiene como propósito demostrar al lector la importación de un dataset y el manejo del mismo utilizando la herramienta Gephi para su análisis.

1 Introducción

Los grafos son un conjunto de puntos llamados vertices los cuales tiene la característica de contar con una unión por líneas llamadas aristas (que pueden contar con un peso) representado como G = (V, E). Pueden ser de dos tipos, dirigidos y no dirigidos. Para la implementación de la programación siguiente se uso el lenguaje C++ y se instaló la librería de SNAP (Stanford Network Analysis Project) para su uso en particular.

Figure 1: Grafo Dirigido

2 Importación de dataset de SNAP

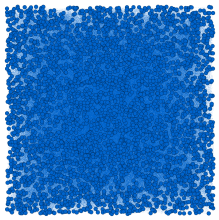
El dataset elejido para la importación del mismo fue el de Facebook encontrado en la página de SNAP en la sección de SNAP Datasets, dentro de Social Network como ego-Facebook. Concecuentemente se descargó el archivo .txt que contiene el grafo y se crearon los códigos basados de un repositorio de Github.

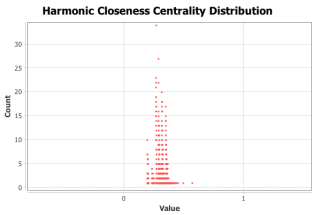
3 Implementación

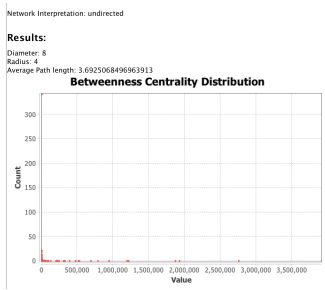
Para la implementación se uso como fuente un repositorio de github (mostrado como fuente en el mismo código) y de ahí se editaron los códigos que se podrán observar en la última sección del documento. Una vez obtenido los archivos exportados a sus formatos se empleo el uso de la herramienta Gephi explicada a continuación.

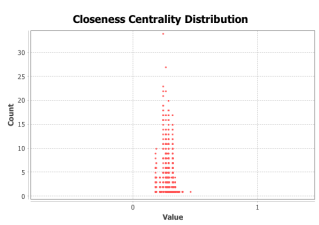
4 Gephi

Para el uso de Gephi se uso como muestra el archivo GraphML exportado del dataset, obtenido por los códigos. De este archivo en específico se obtuvo una gráfica usando dicho software, se exploro por la herramienta y se obtuvieron estos resultados de entre muchos.









5 Conclusion

En conclusión, se obtuvo una buena representación de un tipo de grafo además de extenderlo y exportarlo a más tipos de grafos para así visualizar las semejanzas y diferencias del uso de cada uno. En último término solo se llego a usar la visualización de un archivo solo, en específico el de GraphML, usando la herramienta Gephi.

References

Gephi 2008-2017. GDF Format. Consultado: 30 de octubre de 2019 de https://gephi.org/users/supported-graph-formats/gdf-format/

Gephi. GEXF File Format. Consultado: 30 de octubre de 2019 de https://gephi.org/gexf/format/

```
#include <iostream>
#include <fstream>
#include
    mauve"mauveSnapmauve.mauvehmauve"
#include <ctime>

typedef PNGraph MyGraphType;

bluevoid GraphML(MyGraphType g) {
    std::ofstream file
        (mauve"mauvefbmauve.mauvegraphmlmauve");
   blueif (file.is_open()) {
      file <<
          mauve"mauve<?mauvexmlmauve
          mauveversionmauve=\"1.0\"mauveencodingmauve=\"mauveUTFr
      file <<
          mauve"mauve<mauvegraphmlmauve
          mauvexmlnsmauve=\"mauvehttpmauve://mauvegraphmlmauve.ma
          mauvexmlnsmauve:mauvexsimauve=\"mauvehttpmauve://mauvew
          mauvexsimauve:mauveschemaLocationmauve=\"mauvehttpmauve
          mauvehttpmauve://mauvegraphmlmauve.mauvegraphdrawingmau
      file <<
          mauve"mauve<mauvegraphmauve
          mauveidmauve=\"mauveGmauve\"mauve
          mauveedgedefaultmauve=\"mauvedirectedmauve\">\mauvenmau

      bluefor
          (MyGraphType::TObj::TNodeI
          NI = g->BegNI(); NI <
          g->EndNI(); NI++)
        file <<
            mauve"mauve<mauvenodemauve
            mauveidmauve=\"mauve"
            << NI.GetId() <<
            mauve"mauve\"/>\mauvenmauve";

      blueint i = 1;
      bluefor
          (MyGraphType::TObj::TEdgeI
```

```
        EI = g->BegEI(); EI <
        g->EndEI(); EI++, ++i)
       file <<
           mauve"mauve<mauveedgemauve
           mauveidmauve=\"mauveemauve"
           << i <<
           mauve"mauve\"mauve
           mauvesourcemauve=\"mauve"
           << EI.GetSrcNId() <<
           mauve"mauve\"mauve
           mauvetargetmauve=\"mauve"
           << EI.GetDstNId() <<
           mauve"mauve\"/>\mauvenmauve";

    file <<
        mauve"mauve</mauvegraphmauve>\mauvenmauve";
    file <<
        mauve"mauve</mauvegraphmlmauve>\mauvenmauve";
    file.close();
  }
}

bluevoid GEXF(MyGraphType g) {
  std::ofstream file
      (mauve"mauvefbmauve.mauvegexfmauve");
  blueif (file.is_open()) {
    file <<
        mauve"mauve<?mauvexmlmauve
        mauveversionmauve=\"1.0\"mauve
        mauveencodingmauve=\"mauveUTFmauve-8\"?>\mauvenmauve";
    file <<
        mauve"mauve<mauvegexfmauve
        mauvexmlnsmauve=\"mauvehttpmauve://mauvewwwmauve.mauvegexfmauve.mauvenetmauve/1.2mauvedraftm
        mauveversionmauve=\"1.2\">\mauvenmauve";
    file <<
        mauve"mauve<mauvegraphmauve
        mauvemodemauve=\"mauvestaticmauve\"mauve
        mauvedefaultedgetypemauve=\"mauvedirectedmauve\">\mauvenmauve";
    file <<
        mauve"mauve<mauvenodesmauve>\mauvenmauve";
    bluefor
        (MyGraphType::TObj::TNodeI
        NI = g->BegNI(); NI <
        g->EndNI(); NI++)
      file <<
          mauve"mauve<mauvenodemauve
          mauveidmauve=\"mauve"
          << NI.GetId() <<
          mauve"mauve\"mauve
          mauve/>\mauvenmauve";
    file <<
        mauve"mauve</mauvenodesmauve>\mauvenmauve";
    file <<
        mauve"mauve<mauveedgesmauve>\mauvenmauve";
    blueint i = 1;
    bluefor
        (MyGraphType::TObj::TEdgeI
        EI = g->BegEI(); EI <
        g->EndEI(); EI++, ++i)
      file <<
          mauve"mauve<mauveedgemauve
          mauveidmauve=\"mauve"
          << i <<
          mauve"mauve\"mauve
          mauvesourcemauve=\"mauve"
          << EI.GetSrcNId() <<
          mauve"mauve\"mauve
          mauvetargetmauve=\"mauve"
          << EI.GetDstNId() <<
          mauve"mauve\"mauve
          mauve/>\mauvenmauve";
    file <<
    mauve"mauve</mauveedgesmauve>\mauvenmauve";
    file <<
        mauve"mauve</mauvegraphmauve>\mauvenmauve";
    file <<
        mauve"mauve</mauvegexfmauve>\mauvenmauve";
    file.close();
  }
}

bluevoid GDF(MyGraphType g) {
  std::ofstream file
      (mauve"mauvefbmauve.mauvegdfmauve");
  blueif (file.is_open()) {
    file <<
        mauve"mauvenodedefmauve>mauveidmauve
        mauve,mauveVARCHARmauve\mauvenmauve";
    bluefor
        (MyGraphType::TObj::TNodeI
        NI = g->BegNI(); NI <
        g->EndNI(); NI++)
      file << NI.GetId() <<
          mauve"mauve\mauvenmauve";

    file <<
        mauve"mauveedgedefmauve>mauvesourcemauve
```

```
          mauveVARCHARmauve,mauve
          mauvedestinationmauve
          mauveVARCHARmauve\mauvenmauve";
      bluefor
          (MyGraphType::TObj::TEdgeI
          EI = g->BegEI(); EI <
          g->EndEI(); EI++)
        file << EI.GetSrcNId() <<
            mauve"mauve,mauve
            mauve" <<
            EI.GetDstNId() <<
            mauve"mauve\mauvenmauve";

      file.close();
   }
}

bluevoid JSON(MyGraphType g) {
   std::ofstream file
       (mauve"mauvefbmauve.mauvejsonmauve");
   blueif (file.is_open()) {
      file << mauve"mauve{mauve
          mauve\"mauvegraphmauve\":mauve
          mauve{\mauvenmauve";
      file <<
          mauve"mauve\"mauvenodesmauve\":mauve
          mauve[\mauvenmauve";
      bluefor
          (MyGraphType::TObj::TNodeI
          NI = g->BegNI(); NI <
          g->EndNI(); ) {
        file << mauve"mauve{mauve
            mauve\"mauveidmauve\":mauve
            mauve\"mauve" <<
            NI.GetId() <<
            mauve"mauve\"mauve
            mauve}mauve";
        blueif (NI++ == g->EndNI())
          file << mauve"mauve
              mauve],\mauvenmauve";
        blueelse
          file <<
              mauve"mauve,\mauvenmauve";
      }

      file <<
          mauve"mauve\"mauveedgesmauve\":mauve
          mauve[\mauvenmauve";
      bluefor
          (MyGraphType::TObj::TEdgeI
          EI = g->BegEI(); EI <
          g->EndEI(); ) {
        file << mauve"mauve{mauve
            mauve\"mauvesourcemauve\":mauve
            mauve\"mauve" <<
            EI.GetSrcNId() <<
            mauve"mauve\",mauve
            mauve\"mauvetargetmauve\":mauve
            mauve\"mauve" <<
            EI.GetDstNId() <<
            mauve"mauve\"mauve
            mauve}mauve";
        blueif (EI++ == g->EndEI())
          file << mauve"mauve
              mauve]\mauvenmauve";
        blueelse
          file <<
              mauve"mauve,\mauvenmauve";
      }
      file << mauve"mauve}mauve
          mauve}mauve";

      file.close();
   }
}

blueint main() {
   MyGraphType dg =
       TSnap::LoadEdgeList<MyGraphType>(mauve"mauvefbmauve.mauve
   GraphML(dg);
   GEXF(dg);
   GDF(dg);
   JSON(dg);
   bluereturn 0;
}
```