Word2Graph

Programación Orientada a Objetos 2017-1

Docente Jorge Eliecer Camargo Mendoza.
Integrantes

Juan Jesus Pulido Sanchez.

Harold Nicolas Saavedra Alvarado.

Johan Sebastian Salamanca Gonzalez.

Descripción del proyecto

Partiendo de un conjunto de archivos en formato .txt obtenidos de otras computadoras mediante Sockets, se genera una lista de adyacencia, y un grafo que nos muestra las relaciones entre los sujetos de los cuales se encontró información dentro de los archivos. Esto claro después de un proceso de análisis de cada uno de los archivos.

Inicio del Proceso

La página Principal del Sistema te indica qué opción deseas tomar para iniciar la aplicación, Master o Usuario.

Master: Donde se lleva a cabo el procedimiento principal (Análisis Morfológico y Creación de Grafo).

Usuario: De donde se va a extraer la información en archivo .txt

Primeros pasos

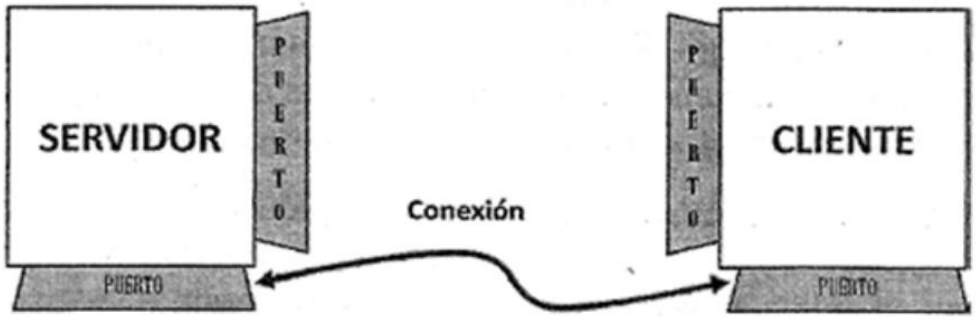


Figura 3.4. Conexión cliente-servidor.

Establecemos una conexión entre el computador que actuará de servidor y el cliente mediante el uso de sockets.
Después que se estableció la conexión a través de la ip y puerto ya especificado,
El servidor esperará por conecciones. mientras que el cliente enviará un conjunto de archivos en formato .txt

Selección de archivos

Después que el usuario agrega y selecciona los directorios, el programa crea un objeto de tipo “listFiles” que buscara en cada uno de estos directorios si existen archivos .txt, en caso de que existan almacenará la ruta absoluta de cada uno de ellos para luego enviarlos hacia el servidor.

Interfaz del Master

Antes de comenzar el Proceso, se debe tener en cuenta que ya finalizó el envío de archivos por parte de los usuarios, luego de su comprobación se procede a ejecutar el procedimiento, este una vez completado muestra un mensaje en pantalla avisando que el proceso ha finalizado y el botón Abrir Grafo se habilita para su Visualización en Gephi

Proceso Gramatical

En esta parte del proyecto se realiza el análisis del texto, para ser posteriormente separado a oraciones, y a palabras para ser clasificado morfológicamente, y de ésta manera llenar un diccionario con el cual compararemos si una palabra tiene comportamiento de Nombre o Conjunción, y de ser así, se crea una doble estructura de guardado inyectiva, en la cual tendrá una la palabra y la otra su respectiva clasificación, en esta segunda estructura se buscará el patrón Nombre-Conjunción-Nombre, en la cual se guardarán en otra estructura la cual no tendrá patrones repetidos, en la cual la conjunción fue transformada en un “;”, y de ésta manera se logró generar una lista de Adyacencia que pudiera ser Leída por la clase Graphs.

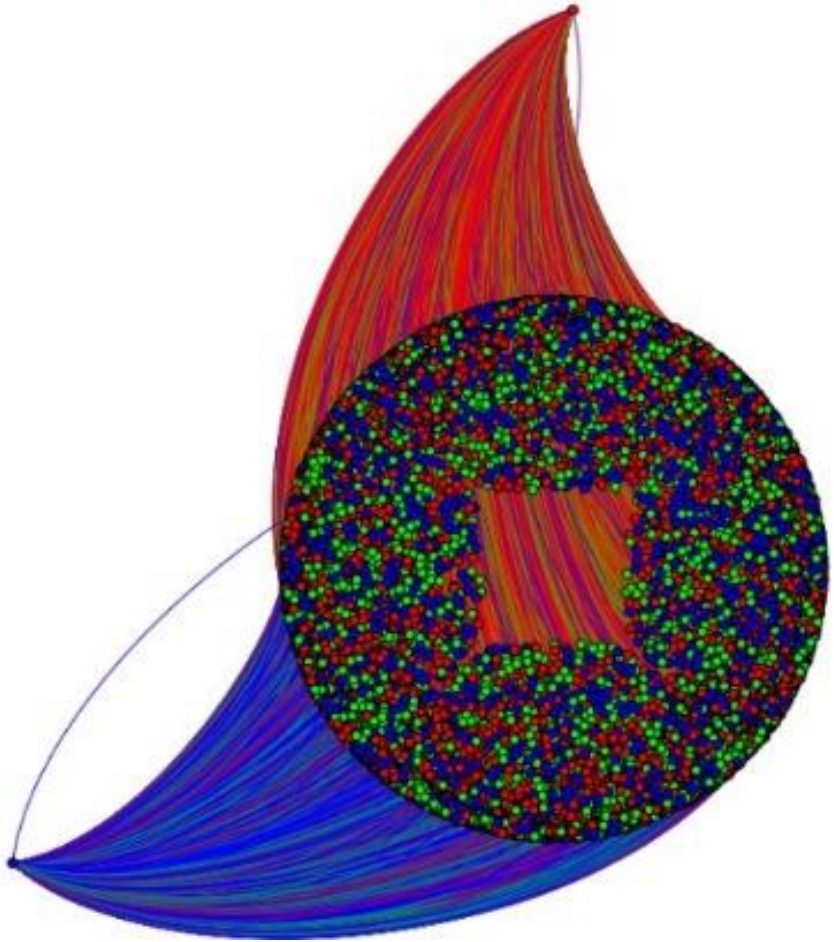
Manejo de Gephi

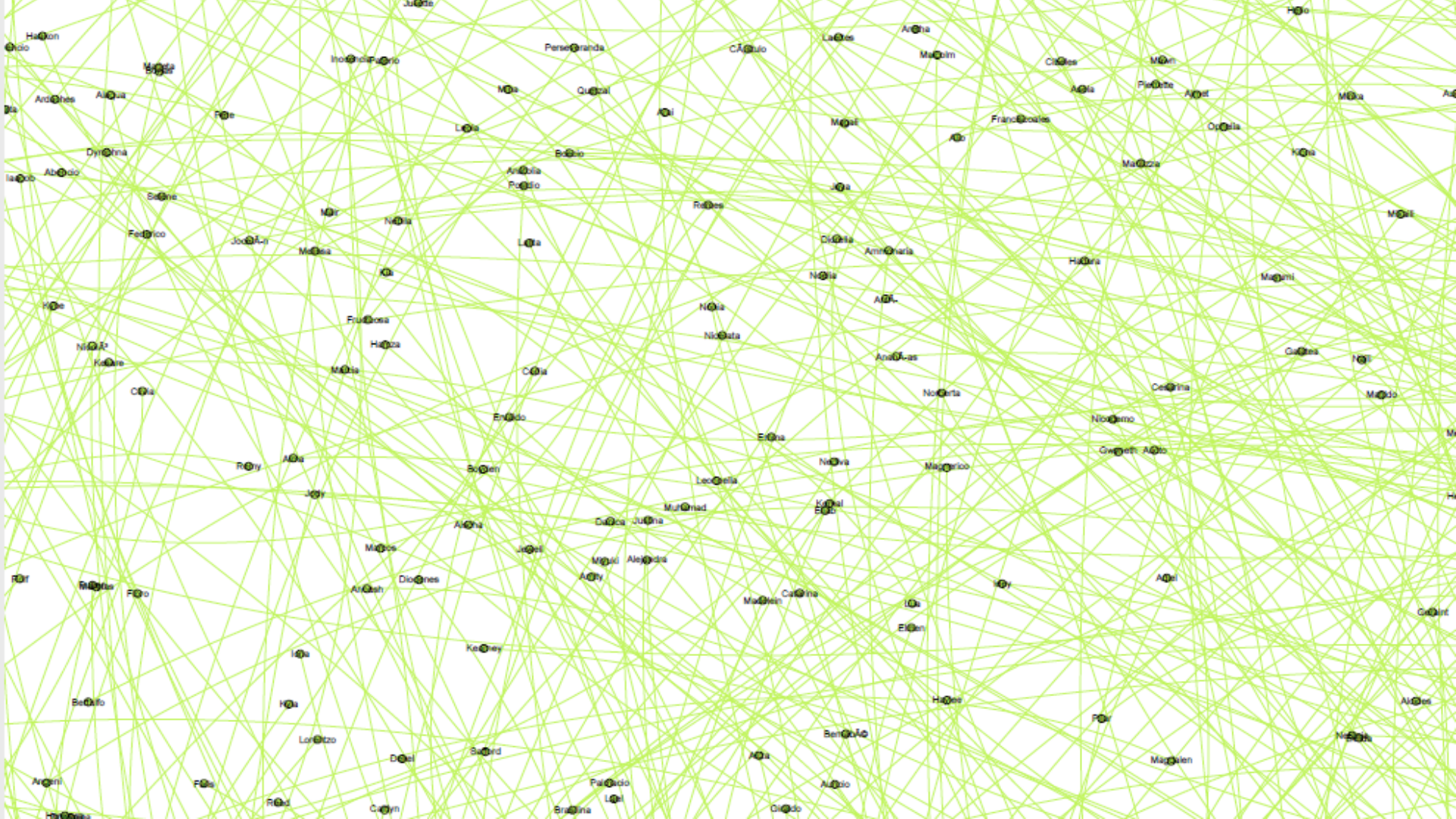
Como ayuda para graficar los datos utilizamos el software Gephi, Gephi es un software open-source Multiplataforma distribuido bajo la doble licencia CDDL 1.0 y GNU General Public License v3 , para lograr la correcta lectura de los datos escribiremos directamente en el pseudocódigo que maneja este software, este consta de una serie de declaraciones donde se establecen los nodos y sus características (color,tamaño,posición) , así mismo establecemos las conexiones entre estos nodos, declarando un nodo base y un nodo objetivo. El atractivo principal de este programa es su amplia capacidad para recibir datos, lo que nos deja manejar miles de nodos al tiempo.

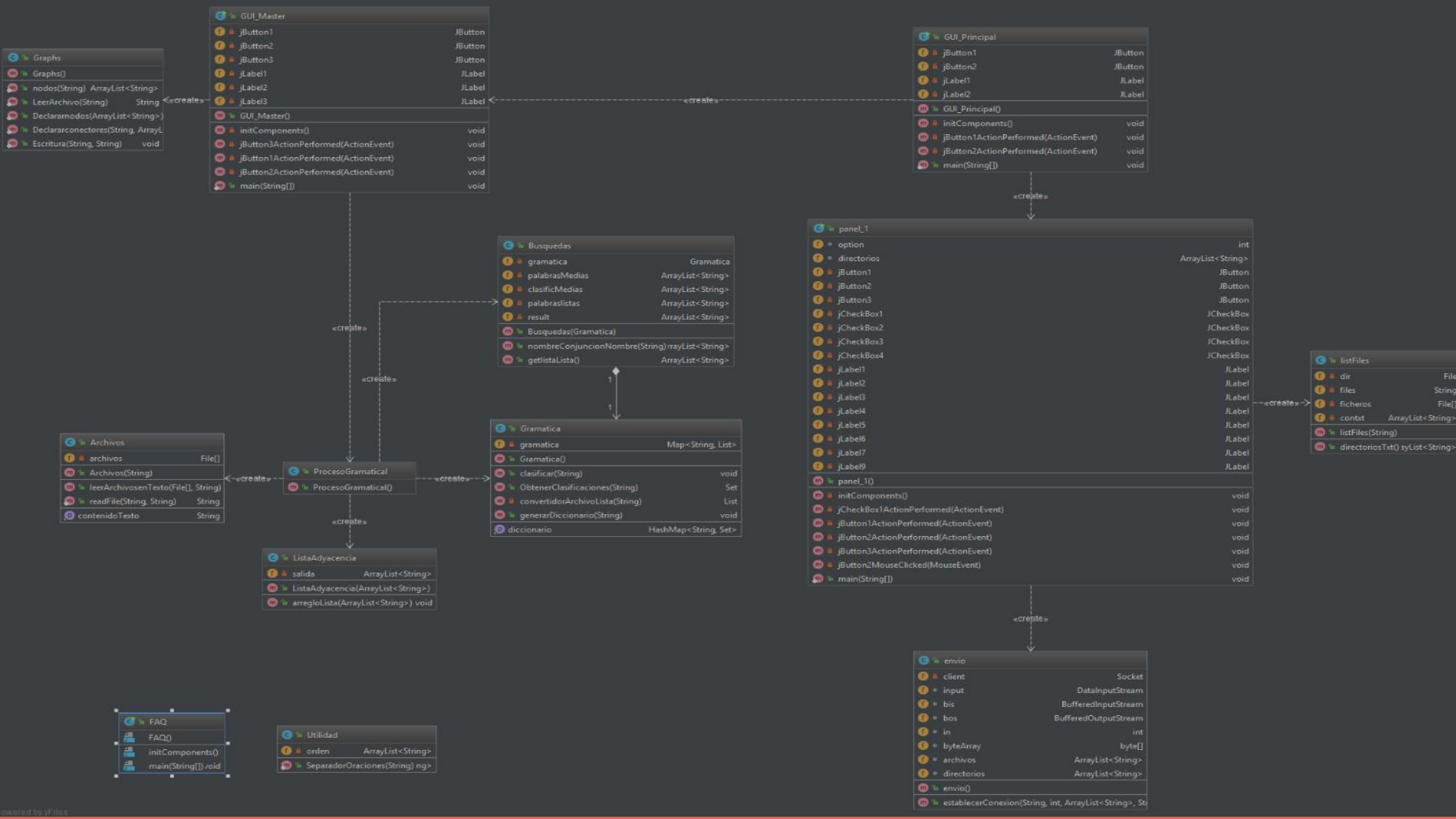
Creación de grafos

Partiendo de la lista de adyacencia previamente creada se genera un array que contendrá los nodos y un TreeMap que relaciona un “id” con cada nodo, esto con el fin de poder establecer las relaciones entre los nodos y así lograr generar un String que contenga el pseudocódigo necesario para que el Software Gephi grafique esta informacion correctamente, este pseudocodigo se guardara en un archivo de tipo .gexf

Ejemplo






Graphs
Graphs()
nodos(String) ArrayList<String>
LeerArchivo(String) String
Declararnodos(ArrayList<String>)
Declararconectores(String, ArrayL
Escritura(String, String) void
«create»
GUI_Master
jButton1 JButton
jButton2 JButton
jButton3 JButton
jLabel1 JLabel
jLabel2 JLabel
jLabel3 JLabel
GUI_Master()
initComponents() void
jButton3ActionPerformed(ActionEvent) void
jButton1ActionPerformed(ActionEvent) void
jButton2ActionPerformed(ActionEvent) void
main(String[]) void
«create»
GUI_Principal
jButton1 JButton
jButton2 JButton
jLabel1 JLabel
jLabel2 JLabel
GUI_Principal()
initComponents() void
jButton1ActionPerformed(ActionEvent) void
jButton2ActionPerformed(ActionEvent) void
main(String[]) void
«create»
Busquedas
gramatica Gramatica
palabrasMedias ArrayList<String>
clasificMedias ArrayList<String>
palabraslistas ArrayList<String>
result ArrayList<String>
Busquedas(Gramatica)
nombreConjuncionNombre(String) rrayList<String>
getlistaLista() ArrayList<String>
«create»
«create»
1
1
panel_1
option int
directorios ArrayList<String>
jButton1 JButton
jButton2 JButton
jButton3 JButton
jCheckBox1 JCheckBox
jCheckBox2 JCheckBox
jCheckBox3 JCheckBox
jCheckBox4 JCheckBox
jLabel1 JLabel
jLabel2 JLabel
jLabel3 JLabel
jLabel4 JLabel
jLabel5 JLabel
jLabel6 JLabel
jLabel7 JLabel
jLabel9 JLabel
panel_1()
initComponents() void
jCheckBox1ActionPerformed(ActionEvent) void
jButton1ActionPerformed(ActionEvent) void
jButton2ActionPerformed(ActionEvent) void
jButton3ActionPerformed(ActionEvent) void
jButton2MouseClicked(MouseEvent) void
main(String[]) void
«create»
listFiles
dir File
files String
ficheros File[]
contxt ArrayList<String>
listFiles(String)
directoriosTxt() syList<String>
Archivos
archivos File[]
Archivos(String)
leerArchivosenTexto(File[], String)
readFile(String, String) String
contenidoTexto String
«create»
ProcesoGramatical
ProcesoGramatical()
«create»
«create»
Gramatica
gramatica Map<String, List>
Gramatica()
clasificar(String) void
ObtenerClasificaciones(String) Set
convertidorArchivoLista(String) List
generarDiccionario(String) void
diccionario HashMap<String, Set>
ListaAdyacencia
salida ArrayList<String>
ListaAdyacencia(ArrayList<String>)
arregloLista(ArrayList<String>) void
«create»
envio
client Socket
input DataInputStream
bis BufferedInputStream
bos BufferedOutputStream
in int
byteArrey byte[]
archivos ArrayList<String>
directorios ArrayList<String>
envio()
establecerConexion(String, int, ArrayList<String>, St
FAQ
FAQ()
initComponents()
main(String[]):void
Utilidad
orden ArrayList<String>
SeparadorOraciones(String) ng>

Complejidades presentadas

Contextualizar oraciones.

Identificar tiempos de los verbos y familias de palabras (palabras derivadas).

Acceso a todo el diccionario de la RAE.

Deducir los Nombres detrás de los pronombres.

El simple hecho de determinar un patrón para hallar oraciones es difícil, y aún más en el español.

La librería de Gephi para Java es bastante confusa y un poco compleja de entender.

El Proceso de la GUI junto con el de los Sockets nos generó problemas, por lo cual no pudimos unirlas.

Futuras actualizaciones

Aplicar Deep learning y redes neuronales para contextualizar y reconocer patrones de familias de palabras (Raíz), así como reconocimiento de palabras desconocidas.

Ampliar Diccionario con todas las palabras de la RAE.

Desarrollo de un sistema capaz de deducir preguntas sobre el texto, analizando al 100% cada una de las oraciones.

Ampliar Gramática MorfoSintáctica del proyecto (objetivo Directo, circunstancial, tipos de sustantivos, morfemas, lexemas, préstamos lingüístico, entre otros...)

Utilizar Python (Tensorflow) Para representación de Palabras a Vectores (Word2Vec).

Utilizar una red Autogenerada por el sistema.

Creación de MultiGrafos Dinámicos de varias dimensiones junto a las diferentes implicaciones deducidas del texto por el sistema.