Red de contenidos

Hansel Blanco Martí HANSEL.BLANCO@ESTUDIANTES.MATCOM.UH.CU
Grupo C311

Karlos Alejandro Alfonso KARLOS.ALFONSO@ESTUDIANTES.MATCOM.UH.CU
Grupo C311

Tutor(es):
MSc. Fernando Raúl Rodríguez Flores, *Facultad de Matemática y Computación, Universidad de La Habana.*

Resumen

Procesamiento de lenguaje natural en español, en archivos de formato *pdf*, que incluyen libros de texto, conferencias, anotaciones, de la asignatura Matemática Numérica para, a partir del modelo planteado, obtener grafo ponderado con los contenidos de la asignatura y analizar los datos obtenidos.

Abstract

Natural language processing in Spanish, in *pdf* format files, which include textbooks, conferences, annotations, of the Numerical Mathematics subject to, from the proposed model, obtain a weighted graph with the contents of the subject and analyze the obtained data.

Palabras Clave: procesamiento, lenguaje, natural, palabras, grafo, ordenación, ranking, datos.

1. Introducción

Cada persona es consciente de las poderosas características de las redes que se habitan, incluso si no se pudiera comprender completamente las fuerzas detrás de ellas. Un encuentro casual en una fiesta revela a un amigo común que vive a miles de kilómetros de distancia. El folclore nos dice que la mejor manera de encontrar un nuevo trabajo es "conectarse" a través de conocidos y amigos de amigos. Se ha navegado por la web y se percibe que solo unos pocos enlaces se han alejado mucho de las intenciones originales.

El mundo se encuentra altamente interconectado, en muchas escalas diferentes. Las conexiones son de lo que se tratan las redes: dos nodos están unidos por una arista cuando están relacionados de una manera específica; las personas están atadas a sus amigos, las ciudades están conectadas por carreteras y rutas aéreas, la flora y la fauna están unidas en una red alimentaria, los países están involucrados en relaciones comerciales; pero no todas las redes son físicas: la World Wide Web es una red virtual de información.

Hasta hace poco, pensar en redes a gran escala era intratable. Diferentes campos reaccionaron a esta deficiencia a su manera. Los sociólogos se limitaron a conjuntos de datos pequeños (menos de 50 personas). Los matemáticos se centraron en modelos puramente estéticos libres de aplicaciones. Los científicos utilizaron modelos simplificados que ignoraban las redes subyacentes. La era de la información moderna ha producido una gran cantidad de datos sobre las complejas redes que unen a las personas. En respuesta, ha surgido el campo de la Ciencia de Redes, a partir de las matemáticas, la informática, la sociología, la economía y las ciencias.

Las últimas dos décadas han sido testigo del desarrollo de una metodología coherente para analizar, comprender e interpretar datos de red. La Ciencia de Redes es altamente interdisciplinaria: el campo sintetiza múltiples tradiciones para lidiar con la comprensión de la estructura de las redes y cómo esta estructura afecta las interacciones dinámicas entre sus nodos.

La idea del proyecto surge a partir de la modelación hecha por Andrew Beveridge, profesor de Macalester College, St Paul, MN.

Dicha modelación se basa en los personajes de la famosa serie "Game of Thrones.". Para su análisis, obtuvo los datos de las novelas de George R. R. Martin: "A Song of Ice and Fire.".

2. Desarrollo de la aplicación

2.1 Sobre la implementación

La implementación del sistema se ha realizado en *Python*, este cuenta con muchas herramientas para el procesamiento de lenguaje natural, al ser tan empleado para realizar trabajos de manejo de datos, recuperación de información e inteligencia artificial contamos con una extensa documentación y ayuda para diversos problemas.

2.2 Procesamiento de lenguaje natural

El procesamiento de lenguaje natural converge al procesamiento de los archivos en formato *pdf*, dificultada por la conocida heterogeneidad de los documentos en cuestión.

2.2.1 De *PDF* a *PYTHON*

Para obtener la información de texto de cada archivo en formato *pdf* se ha utilizado la biblioteca *fitz*, guardando en un diccionario de *Python* los vocablos preprocesados(se explicará a continuación) de cada artículo.

En la Figura 1 se muestra el código en que se emplea dicha tarea.

```
def pdfs_info_dict(path : str):
pdf_files = read_pdfs(path)
pdfs_text_info = {}

pdf_index = 0
for pdf in pdf_files:
    with fitz.open(pdf) as document:
        words, nouns
        = preprocess(document)
        pdfs_text_info[pdf_index]
        = words, nouns
    pdf_index+=1
return pdfs_text_info

def read_pdfs(path : str):
    pdf_search = Path(path).glob("*.pdf")
    return [str(file.absolute())
        for file in pdf_search]
```

Figura 1: De *pdf* a *Pyhton*.

2.2.2 Preprocesamiento de texto

Para representar los documentos primeramente es necesario realizar un preprocesado, en el sistema se procede de forma simple, empleando recursos del procesamiento de lenguaje natural para modificar los documentos. Se han utilizado las bibliotecas: *numpy, nltk, spacy*

Las siguientes transformaciones son realizadas para cada documento del corpus:

1. Tokenizar el documento: Primeramente es necesario crear la representación básica de un documento, la división en tokens, de tal forma que se pueda realizar procesamientos más avanzados sobre cada palabra. Desde esta etapa del preprocesamiento se puede llevar a minúscula todos los caracteres de las palabras, eliminar símbolos y signos de puntuación.

2. Eliminación de stopwords: Las stopwords son palabras vacías de significado, que pueden servir de nexo entre entidades o funcionan como modificadores. Si se incluyen estas palabras en la representación del documento se puede afectar la precisión del modelo, debido a la alta frecuencia que poseen estas palabras en los textos.

3. Stemming: Este método busca relacionar palabras con igual significado pero que difieren en cuanto a la escritura, por tener prefijos o sufijos.

4. Lemmatizing: La lematización es un proceso lingüístico que consiste en, dada una forma flexionada (es decir, en plural, en femenino, conjugada, etc), hallar el lema correspondiente. El lema es la forma que por convenio se acepta como representante de todas las formas flexionadas de una misma palabra. Es decir, el lema de una palabra es la palabra que se encontraría como entrada en un diccionario tradicional: singular para sustantivos, masculino singular para adjetivos, infinitivo para verbos. Por ejemplo, *decir* es el lema de *dije*, pero también de *diré* o *dijéramos*.

En la Figura 2 se muestra el código de preprocesamiento.

```
def preprocess(document):
    accum_text = ''
    for i in range(document.page_count):
        page = document.load_page(i)
        text = page.get_text('text')
        accum_text += text

    stop_words =
        set(stopwords.words("spanish"))

    nlp =
        spacy.load('es_core_news_md')

    accum_text =
        remove_punctuation(accum_text)
    accum_text =
        remove_apostrophe(accum_text)

    spacy_document = nlp(accum_text)
    nouns = []
    words = []
    for token in spacy_document:
        if token.pos_ == "NOUN":
            nouns.append(token.lemma_.lower())
        words.append(token.lemma_.lower())

    return [word for word in words
        if word.casefold() not in stop_words],
        [noun for noun in nouns]
```

Figura 2: Preprocesamiento.

2.3 Construcción del grafo

Para el experimento, se ha decidido construir un grafo no dirigido y ponderado. Un grafo es un par de conjuntos V, E donde V contiene los vértices y E las aristas, en este caso, al ser no dirigido, no es relevante el orden en cada par de vértices que pertenece al conjunto E. Cada arista tiene una ponderación determinada por el algoritmo que describiremos adelante.

Para la construcción del grafo se ha utilizado la biblioteca *networkx*.

Lo primero es definir los contenidos que van a representar el conjunto de dichos vértices. Para ello, se crea la clase *Content* de la Figura 3.

```
class Content:
    def __init__(self, *name,
        node_description = '') -> None:
        self.name = name
        self.distances :
            Dict[Content, List[int]] = {}
        self.node_description
            = node_description

    (.....)

    def __str__(self) -> str:
        return self.node_description
```

Figura 3: Clase para los contenidos.

El parámetro *name* indica la lista de nombres asociados al contenido.

El parámetro *node-description* indica el nombre del contenido que hará que sea entendible la visualización del grafo.

La clase *Graph* definida, en términos de software, representa una capa por encima de la biblioteca usada y su grafo, incluyendo comportamientos necesarios para la ponderación en el modelo.

2.3.1 Modelación

La forma en que se convirtió esta compleja colección de palabras en una colección de redes de interacción fue vinculando dos contenidos cada vez que su nombre (o colección de nombres, en general, texto identificado como contenido) aparecieran en una vecindad de 1000 palabras (*distancia-vecindad* establecida), sumando una ponderación P a la arista de los contenidos A y B por cada aparición en la vecindad, de la forma:

$$P(A,B) = \frac{distancia - vecindad}{distancia(A,B) + 1}$$

De esta forma se garantiza que se suma a dicha ponderación un valor inversamente proporcional a la distancia, es decir, a mayor distancia menor valor a sumar, y viceversa.

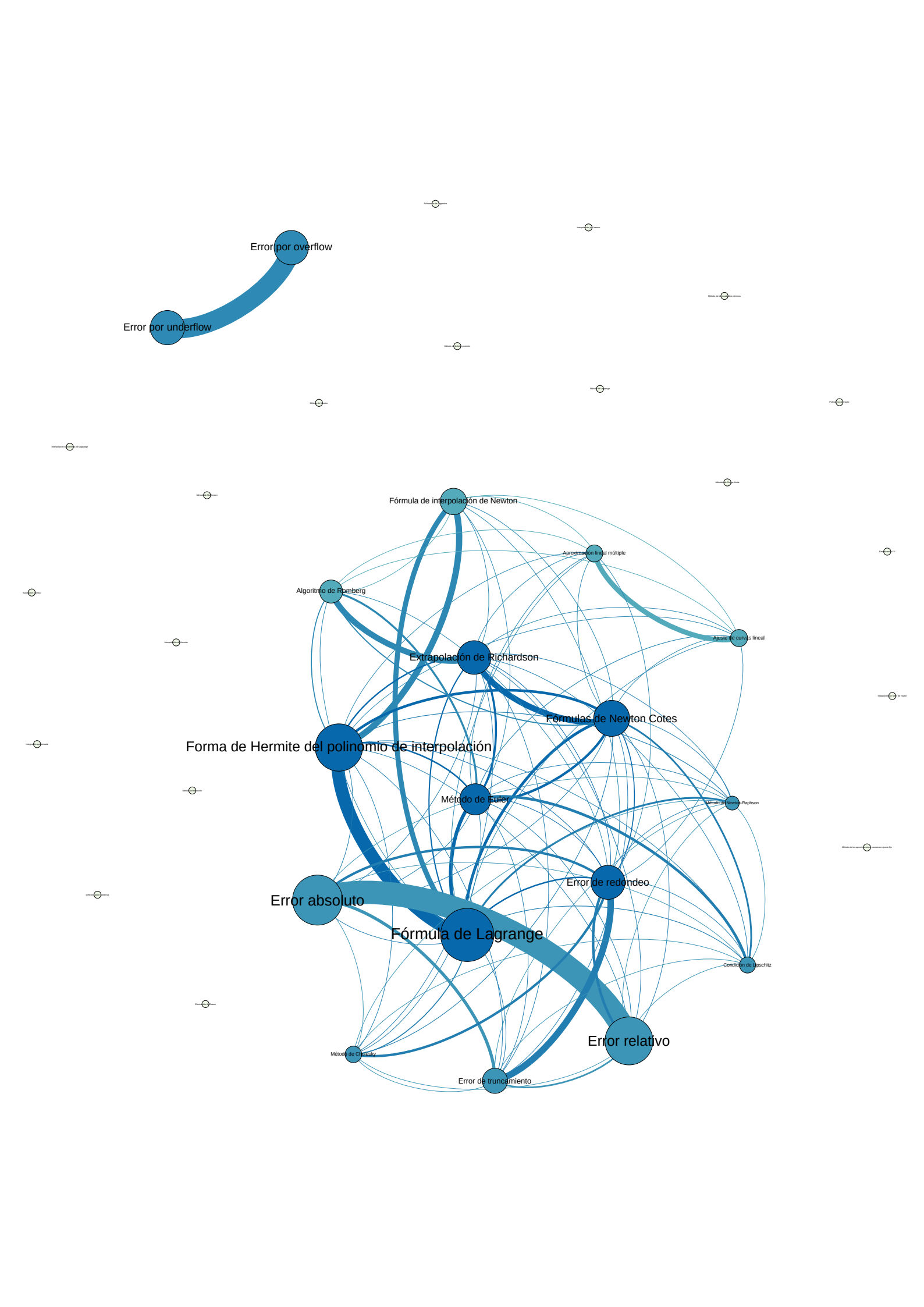
3. Resultados

Para el análisis de resultados se ha utilizado el software *Gephi*, herramienta destinada para la Ciencia de Redes.

En la siguiente página, se muestra una representación espacial del grafo resultante de aplicar el modelo descrito con la distribución Fruchterman Reingold.

Para el tamaño y la coloración de los vértices se ha realizado un ranking a partir de sus grados con pesos.

Para el grosor y la coloración de las aristas se ha realizado el ranking por ponderación.


Error por overflow
Error por underflow
Fórmula de interpolación de Newton
Aproximación lineal múltiple
Algoritmo de Romberg
Ajuste de curvas lineal
Extrapolación de Richardson
Fórmulas de Newton Cotes
Forma de Hermite del polinomio de interpolación
Método de Euler
Método de Newton-Raphson
Error de redondeo
Error absoluto
Fórmula de Lagrange
Condición de Lipschitz
Error relativo
Método de Cholesky
Error de truncamiento

3.1 Componentes conexas

El grafo tiene 21 componentes conexas, siendo 19 de estas vértices aislados o de grado 0, es decir, 19 contenidos que no están relacionados con el resto, según la definición de relación del modelo. Se muestran en la siguiente tabla:

Label	Grado
Diferenciación numérica	0
Eliminación de Gauss	0
Factorización LU	0
Integración aproximada	0
Integración por serie de Taylor	0
Interpolación baricéntrica de Lagrange	0
Interpolación de Hermite	0
Interpolación por tramos	0
Método de bisección	0
Método de la falsa posición	0
Método de Lagrange	0
Método de las aproximaciones sucesivas o pu...	0
Método de los cuadrados mínimos	0
Método de Newton	0
Método de Steffensen	0
Métodos de Runge-Kutta	0
Polinomios de Legendre	0
Polinomios de Taylor	0
Sustitución inversa	0

Figura 4: Contenidos aislados.

.

Las componentes conexas restantes (2) tienen cardinalidad 16 y 2 respectivamente.

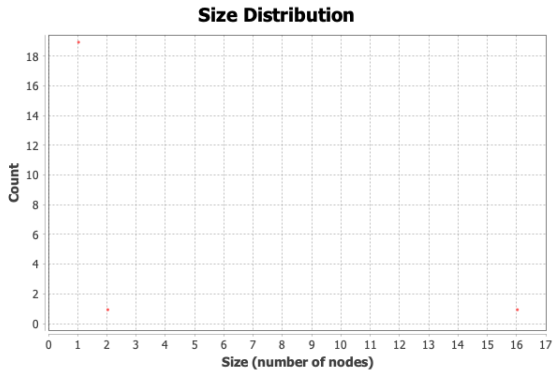


Figura 5: Tabla de distribución de tamaños de las componentes conexas.

.

3.2 Grado de los vértices

Para hacer legible la información, no se han incluido los vértices de grado 0, analizados en la sección de componentes conexas.

Se observa cuáles son los contenidos que más interactúan con el resto a partir del grado de los vértices.

El grado con pesos describe, a partir de las ponderaciones de las aristas que relacionan dicho vértice con sus adyacentes, cuánto interactúa dicha temática con el resto.

Label	Grado	Grado con pesos
Método de Euler	15	478.0
Extrapolación de Richardson	15	528.0
Error de redondeo	15	535.0
Fórmulas de Newton Cotes	15	575.0
Forma de Hermite del polinomio de interpo...	15	799.0
Fórmula de Lagrange	15	916.0
Método de Newton-Raphson	11	136.0
Condición de Lipschitz	11	178.0
Método de Cholesky	11	188.0
Error de truncamiento	11	359.0
Error relativo	11	812.0
Error absoluto	11	853.0
Aproximación lineal múltiple	9	196.0
Ajuste de curvas lineal	9	200.0
Algoritmo de Romberg	9	318.0
Fórmula de interpolación de Newton	9	385.0
Error por overflow	1	541.0
Error por underflow	1	541.0

Figura 6: Grado de relación de los contenidos.

.

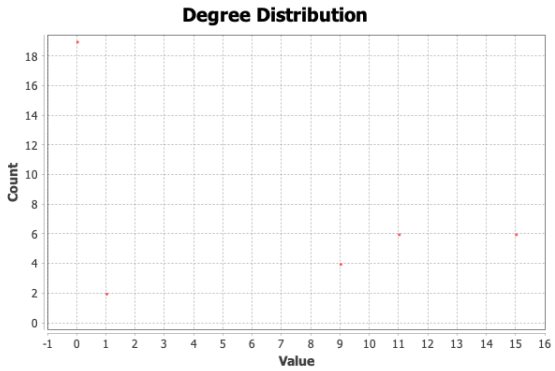


Figura 7: Tabla de distribución de grados de los contenidos.

.

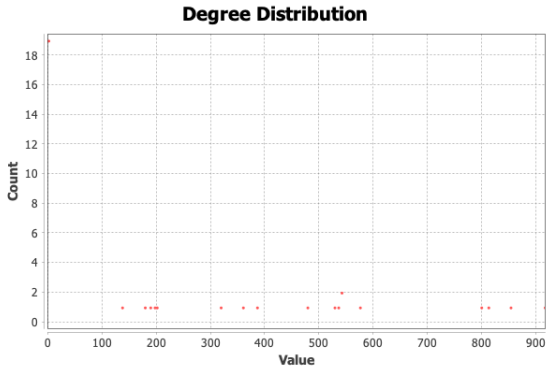


Figura 8: Tabla de distribución de grados con peso de los contenidos.

.

3.3 PageRank

PageRank es una familia de algoritmos creada y desarrollada por la compañía Google para optimizar las búsquedas de páginas web, que clasifica las páginas de los nodos de acuerdo a la frecuencia con la que un usuario siguiendo enlaces llega a la página del nodo de forma no aleatoria.

Expandiendo la idea al grafo modelado, con los parámetros: $P = 0.85$ (probabilidad usada para simular aleatoriamente que el usuario reinicia la navegación web) y $\epsilon = 0.001$ (criterio de parada, cuanto menor sea este valor, más tiempo tomará la convergencia) se ha obtenido la siguiente ordenación y distribución de contenidos.

Label	PageRank
Método de Euler	0.058497
Extrapolación de Richardson	0.058497
Fórmula de Lagrange	0.058497
Error de redondeo	0.058497
Fórmulas de Newton Cotes	0.058497
Forma de Hermite del polinomio de interpo...	0.058497
Error por overflow	0.047934
Error por underflow	0.047934
Método de Newton-Raphson	0.044128
Condición de Lipschitz	0.044128
Método de Cholesky	0.044128
Error de truncamiento	0.044128
Error relativo	0.044128
Error absoluto	0.044128
Aproximación lineal múltiple	0.037798
Ajuste de curvas lineal	0.037798
Algoritmo de Romberg	0.037798
Fórmula de interpolación de Newton	0.037798

Figura 9: Tabla de ordenación por PageRank (valores de PageRank mayores que 0,01).

.

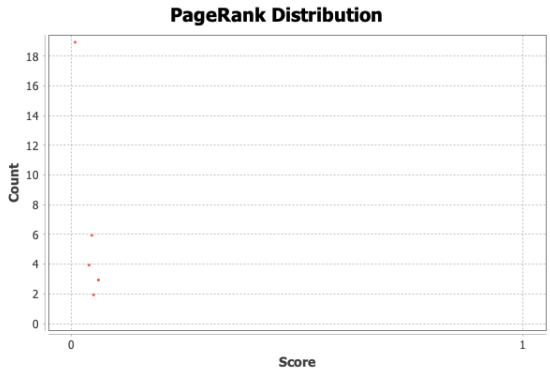


Figura 10: Tabla de distribución de PageRank calculados.

.

3.4 Peso de las aristas

En la Figura 11 se muestra la tabla con las interacciones más fuertes entre contenidos.

Origen	Destino	Weight
Error absoluto	Error relativo	657.0
Error por overflow	Error por underflow	541.0
Fórmula de Lagrange	Forma de Hermite del polinomio de interpolación	366.0
Error de redondeo	Error de truncamiento	185.0
Fórmula de interpolación de Newton	Forma de Hermite del polinomio de interpolación	176.0
Fórmulas de Newton Cotes	Extrapolación de Richardson	168.0
Extrapolación de Richardson	Algoritmo de Romberg	166.0
Fórmula de Lagrange	Fórmula de interpolación de Newton	143.0
Ajuste de curvas lineal	Aproximación lineal múltiple	142.0
Fórmula de Lagrange	Método de Euler	99.0
Error absoluto	Error de truncamiento	88.0
Fórmula de Lagrange	Fórmulas de Newton Cotes	79.0
Método de Euler	Condición de Lipschitz	76.0
Forma de Hermite del polinomio de interpolación	Fórmulas de Newton Cotes	74.0
Error relativo	Error de redondeo	72.0
Fórmulas de Newton Cotes	Método de Euler	69.0
Error de redondeo	Método de Cholesky	68.0
Error absoluto	Error de redondeo	68.0
Extrapolación de Richardson	Método de Euler	58.0
Algoritmo de Romberg	Método de Euler	56.0
Fórmula de Lagrange	Método de Newton-Raphson	52.0
Fórmulas de Newton Cotes	Condición de Lipschitz	49.0
Error relativo	Error de truncamiento	46.0
Forma de Hermite del polinomio de interpolación	Método de Euler	42.0
Fórmula de Lagrange	Error de redondeo	41.0
Forma de Hermite del polinomio de interpolación	Extrapolación de Richardson	40.0
Fórmula de Lagrange	Extrapolación de Richardson	34.0
Fórmulas de Newton Cotes	Algoritmo de Romberg	32.0
Forma de Hermite del polinomio de interpolación	Algoritmo de Romberg	28.0

Figura 11: Ranking por ponderación de aristas (Top 30).

.

4. Conclusiones

A modo resumen, en la asignatura Matemática Numérica:

1. Contenidos más relacionados con el resto:
   a) Método de Euler
   b) Extrapolación de Richardson
   c) Error de redondeo
   d) Fórmulas de Newton Cotes
   e) Forma de Hermite del polinomio de interpolación
   f) Fórmula de Lagrange
2. Contenidos más influyentes en sus relacionados:
   a) Fórmula de Lagrange
   b) Error absoluto
   c) Error relativo
   d) Forma de Hermite del polinomio de interpolación
   e) Fórmulas de Newton Cotes
   f) Error por overflow
   g) Error por underflow
   h) Error de redondeo
   i) Extrapolación de Richardson
   j) Método de Euler
3. Relaciones más fuertes entre contenidos:
   a) Error absoluto ——— Error relativo
   b) Error por overflow ——— Error por underflow
   c) Fórmula de Lagrange ——— Forma de Hermite del polinomio de interpolación

d) Fórmulas de Newton Cotes ——— Extrapolación de Richardson

e) Extrapolación de Richardson ——— Algoritmo de Romberg

f) Fórmula de Lagrange ——— Fórmula de interpolación de Newton

5. Recomendaciones

Para un análisis más exhaustivo o específico de la información se recomienda instalar la aplicación *Gephi*, hecha para todas las plataformas, y abrir el archivo *numerical-analysis-words.gephi* para investigar dentro de este software con la base de la investigación expuesta en este documento.

Referencias

[1] Andrew Beveridge. *Network of Thrones*. URL: https://networkofthrones.wordpress.com/

[2] *Gephi*. The Open Graph Viz Platform. URL: https://gephi.org/

[3] Robert Tarjan. *Depth-First Search and Linear Graph Algorithms*, in SIAM Journal on Computing.

[4] Page, Lawrence and Brin, Sergey and Motwani, Rajeev and Winograd, Terry (1999) *The PageRank Citation Ranking: Bringing Order to the Web*.

[5] Wikipedia. URL: http://en.wikipedia.org.

[6] *The Right It*, Alberto Savoia.

[7] *The Compound Effect*, Darren Hardy.