UNIVERSIDAD DE CHILE
FACULTAD DE CIENCIAS FÍSICAS Y MATEMÁTICAS
DEPARTAMENTO DE CIENCIAS DE LA COMPUTACIÓN

BÚSQUEDA DE CAMINOS RELEVANTES EN GRAFOS RDF

MEMORIA PARA OPTAR AL TÍTULO DE
INGENIERO CIVIL EN COMPUTACIÓN

GONZALO JAVIER TARTARI BARRIGA

PROFESOR GUÍA:
AIDAN HOGAN

MIEMBROS DE LA COMISIÓN:
CLAUDIO GUTIÉRREZ GALLARDO
ANDRÉS FARÍAS RIQUELME

SANTIAGO DE CHILE
ENERO 2018

RESUMEN DE LA MEMORIA PARA OPTAR
AL TÍTULO DE INGENIERO CIVIL EN COMPUTACIÓN
POR: GONZALO JAVIER TARTARI BARRIGA
FECHA: ENERO 2018
PROF. GUÍA: AIDAN HOGAN

BÚSQUEDA DE CAMINOS RELEVANTES EN GRAFOS RDF

A partir de información representada en una colección de declaraciones en RDF (Resource Description Framework), esta representa intrínsecamente un multi-grafo etiquetado, dirigido. Con esto es posible utilizar algoritmos de búsqueda de caminos, que permiten encontrar la ruta más corta entre dos nodos.

Con el fin de estudiar tecnologías basadas en el desarrollo de la web semántica, el tema de memoria propuesto consiste en la búsqueda de caminos relevantes para grafos en RDF. Para esto se construye una herramienta la que, a partir de un archivo con información en RDF, sea capaz de resolver consultas sobre caminos entre un nodo de origen y un nodo objetivo de tal manera que estos contengan información relevante para el usuario.

La relevancia de los caminos se representa a través del peso asignado tanto a aristas, según su etiqueta, como a vértices, principalmente por métodos basados en su grado, PageRank o variaciones de estos.

Para la búsqueda del camino mínimo entre un nodo origen y un nodo objetivo se implementa una versión del algoritmo de Dijkstra para grafos ponderados.

Con el objetivo de tener la posibilidad de visualizar los resultados obtenidos y usar la herramienta en un contexto real para reportar resultados, es que se construye una aplicación que resuelva consultas de caminos entre dos nodos y represente esta respuesta gráficamente.

Finalmente se realizan pruebas de rendimiento del algoritmo utilizado y pruebas para comparar los caminos resultantes de los distintos métodos de ponderación del grafo. Además se lleva a cabo una evaluación con usuarios para la validación de la solución obtenida.

Agradecimientos

Primero quiero agradecer a mi familia por su apoyo, ya sea a través de consejos brindados o el tiempo compartido. Fueron un pilar clave para llevar a cabo mis estudios universitarios en otra ciudad.

Además, quiero agradecer al profesor Aidan Hogan por guiarme con sus comentarios, dedicación y paciencia a lo largo de todo el proceso en que consistió este trabajo de título. Sin su confianza y apoyo no hubiese podido realizar la memoria.

Tabla de Contenido

Capítulo 1

Introducción

El modelo de datos RDF (Resource Description Framework) es similar a los enfoques de modelado conceptual clásicos como entidad-relación o diagramas de clases, ya que se basa en la idea de hacer declaraciones sobre los recursos (en particular, recursos web) en forma de expresiones sujeto-predicado-objeto.

Originalmente diseñado como un modelo de datos para meta-datos, ha llegado a ser usado como un método general para la descripción conceptual o modelado de la información que se implementa en los recursos web, utilizando una variedad de notaciones de sintaxis y formatos de serialización de datos.

Al establecerse como modelo de datos estándar para describir y publicar datos en la Web ha motivado la implementación de múltiples aplicaciones lo que deja abiertas oportunidades de desarrollo para herramientas nuevas y se encuentra una gran cantidad de información disponible en este formato. Por otro lado el modelo de datos RDF se diferencia de otros por su característica de ser flexible, es decir al ser un grafo menos estructurado (sin un esquema claro) permite añadir información agregando nodos y aristas.

Bajo este contexto, el tema de memoria a realizar consiste en implementar algoritmos de búsqueda de caminos relevantes para el usuario entre dos objetos de interés, de tal manera que el camino entre ambos tenga peso mínimo. Es importante notar que no siempre basta con solo tomar el largo del camino como criterio para determinar el resultado de la búsqueda, por esto hay que considerar grafos ponderados para la solución del problema.

A continuación se presenta un ejemplo con dos posibles caminos resultantes en el grafo de la figura 1.1. El siguiente grafo tiene nodos con tipos (películas, países, actores) y aristas con etiquetas (actuó, nació) y dirección. El camino más corto entre 'Will Smith' y 'Jack Nicholson', sin tomar en cuenta la dirección de las aristas, es el camino marcado en rojo, pero como se observa el nodo 'Estados Unidos' conecta la mayoría de los nodos provocando que todas las búsquedas nos den como resultado un camino pasando por este vértice, lo que no tendría relevancia para el usuario. Otro camino posible sería el marcado en azul, el que ignora el nodo 'Estados Unidos' y recorre las películas en las que han trabajado los actores seleccionados. Si

bien el camino azul es más largo, es probable que sea de mayor interés para el usuario. Una idea general para incorporar esta característica es asociar un peso a cada nodo y/o arista del grafo, considerar el peso de un camino como la suma de los pesos de todas las aristas y/o nodos recorridos y encontrar el camino con peso mínimo. En el contexto de este ejemplo podemos asignarle un peso alto al nodo 'Estados Unidos', debido a su alta conectividad en el grafo, y así el resultado de camino mínimo correspondería al camino azul.

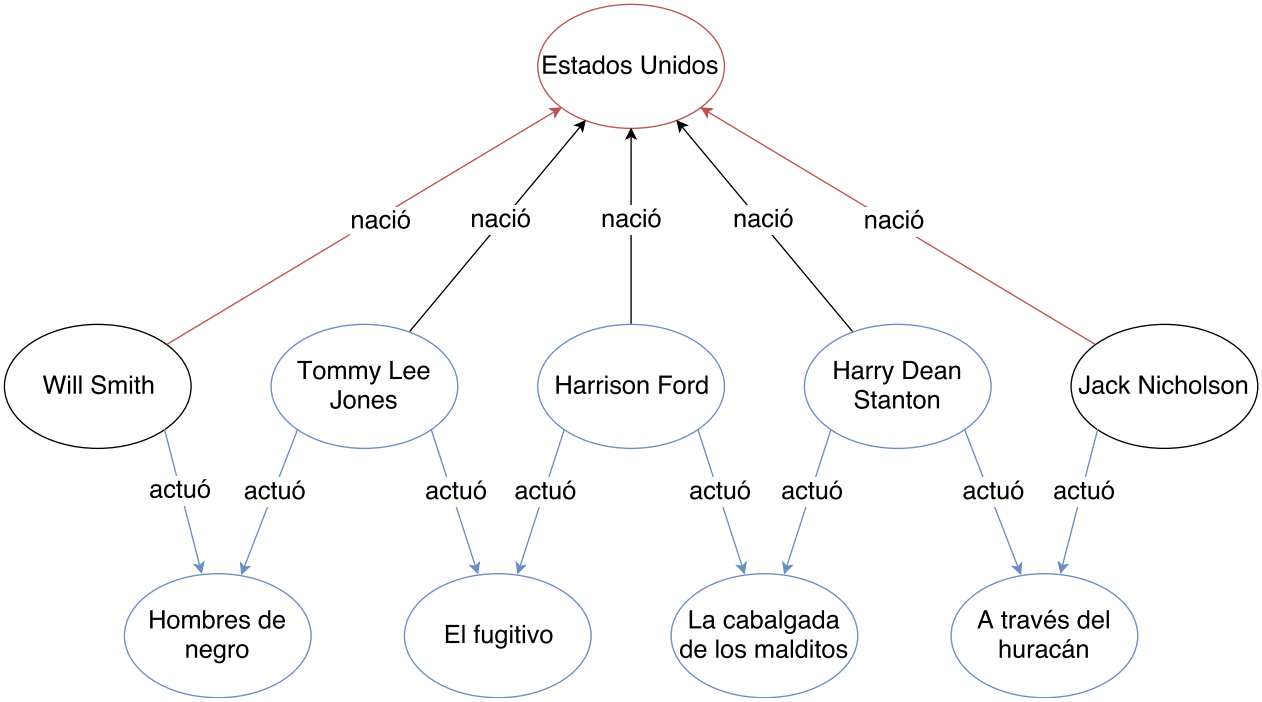


Figura 1.1: Ejemplo de caminos en un grafo.

Un caso en particular de aplicaciones de búsqueda de caminos es el número de Erdős, para describir la distancia colaborativa entre autores, y sus variaciones, entre otros. Actualmente existen múltiples aplicaciones (científicas, comercio, noticias, etc.) que hacen uso de grafos en este formato en los cuales una herramienta de estas características podría ser utilizada para responder consultas, como por ejemplo rutas entre un origen y un destino, profesionales ligados a un área en específico y otros.

1.1. Alternativas analizadas

A continuación se describen posibles algoritmos que se pueden utilizar para solucionar el problema de encontrar caminos en un grafo, una implementación de búsqueda de caminos para un grafo en RDF y software para manipular grafos con funcionalidades para búsqueda de caminos.

El algoritmo de Dijkstra es un algoritmo para la determinación del camino más corto dado un vértice origen al resto de los vértices en un grafo con pesos en cada arista. El algoritmo de búsqueda A* (una extensión del algoritmo de Dijkstra) encuentra, siempre y cuando se

cumplan unas determinadas condiciones, el camino de menor coste entre un nodo origen y uno objetivo.

Actualmente existe una herramienta llamada RelFinder [3] que realiza búsqueda en un grafo RDF entre dos objetos de interés y busca las relaciones existentes entre ellos, sin embargo, en esta implementación no se considera la valoración con pesos en el grafo. Por esto, si bien la aplicación despliega caminos que conectan los objetos, estos son pocos informativos debido a que incluyen nodos con un grado alto. Un ejemplo de esto es la inclusión de países en gran parte de los caminos resultantes debido a que conectan una alta cantidad de nodos. Además, existen bases de datos de grafos que implementan funciones para encontrar caminos más cortos entre dos nodos [1]. Ejemplos de software con estas características son Neo4j (software libre de Base de datos orientada a grafos, implementado en Java) y Gremlin (lenguaje y máquina virtual desarrollado por Apache TinkerPop).

Si bien estas herramientas y lenguajes incluyen la funcionalidad descrita, estas no soportan pesos en los grafos, es decir solo consideran grafos no ponderados. A partir de las alternativas estudiadas se concluye que actualmente no existe una solución que resuelva el problema de encontrar caminos relevantes entre dos nodos. Es por esto que se optó por el desarrollo de una implementación desde cero utilizando algoritmos de búsqueda de caminos en grafos como Dijkstra [5] y A* [2].

Para la implementación de grafos con ponderación se puede considerar la asociación de pesos a nodos o arcos. En un comienzo se asignarán pesos a los nodos según su grado, luego según su PageRank y finalmente se diseñará una nueva ponderación considerando los resultados obtenidos.

1.2. Objetivos

1.2.1. Objetivo general

El objetivo principal es implementar un algoritmo que, dado dos nodos en un grafo RDF, busque caminos cortos que sean de mayor relevancia para el usuario.

1.2.2. Objetivos específicos

- Implementar algoritmos de búsqueda del camino más corto en un grafo RDF.
- Encontrar una metodología para asignar peso a los nodos según su relevancia.
- Implementar la búsqueda del camino más corto en grafos con nodos con peso.
- Validar relevancia de los caminos encontrados.
- Desarrollar una aplicación para hacer búsqueda de caminos y mostrar los resultados obtenidos.

1.3. Plan de trabajo

Con el fin de cumplir los objetivos propuestos se planea el siguiente plan de trabajo:

- Implementar el algoritmo de caminos mínimos de Dijkstra para RDF.
- Probar las implementaciones con grafos de tamaño pequeño.
- Descargar y preparar grafo de Wikidata.
  - Extraer una muestra pequeña y conexa.
  - Cargar y parsear los datos a RDF.
  - Encontrar caminos sin pesos ni dirección.
  - Hacer experimentos de escala.
    - Solo trabajando con el grafo en memoria principal.
- Versión con grafos ponderados.
  - Asignar pesos basados en el grado de cada nodo.
  - Asignar pesos basados en PageRank.
  - Considerar otros métodos basados en los resultados obtenidos.
- Desarrollar una aplicación para la búsqueda de caminos y la visualización de los resultados.
- Evaluar los resultados de los grafos ponderados con usuarios.

1.4. Resultados de la solución implementada

Como resultado de este trabajo de memoria se obtiene una herramienta computacional capaz de encontrar caminos entre dos entidades presentes en una base de conocimiento en RDF, lo que aporta al desarrollo de nuevas aplicaciones al método de publicar información enlazada.

La solución ha sido implementada utilizando algoritmos de búsqueda de caminos mínimos en grafos RDF. Los datos utilizados corresponden a la base de conocimiento que provee Wikidata. Los caminos que genera la aplicación corresponden a los caminos con peso mínimo obtenidos utilizando distintos métodos de ponderación, tanto de nodos como aristas. Estos métodos son principalmente calculados utilizando PageRank o características de los componentes, como su grado o etiqueta.

Los resultados fueron comparados entre cada variación de ponderación utilizada, tanto en rendimiento como en la similitud de los caminos obtenidos. Por otro lado se realizó una evaluación con usuarios de los resultados para validar la relevancia de los caminos.

Capítulo 2

Marco Teórico

En este capítulo se exponen los principales conceptos involucrados con las tecnologías consideradas para el desarrollo del tema de memoria propuesto. También se hace una descripción de las soluciones existentes junto con sus características y la razón de la búsqueda a alternativas de estas.

2.1. La web de datos

La web de datos (o también llamada linked data) es un método para publicar información de manera que los datos se puedan vincular entre ellos y que tanto personas como máquinas tengan la capacidad de explorarlos. Con respecto a los datos enlazados, esto permite que información relacionada de distintas fuentes esté conectada, bajando las barreras para acceder a la información. Por otro lado, al añadir la capacidad de que los computadores puedan leer automáticamente la información amplía las posibilidades de compartir y consultar sobre información de distintos orígenes en un ambiente como la web. A continuación, se describen algunas de las tecnologías relacionadas tanto con la web de datos como con el trabajo que se realiza.

2.1.1. Términos RDF

A continuación se describen los términos definidos para la conformación de triples en RDF:

Literal Los literales son utilizados para identificar valores, por ejemplo números o fechas. Estos literales pueden ser una cadena de caracteres sin tipo y con la opción de una etiqueta especificando el lenguaje al que pertenece, o un literal compuesto por una cadena de caracteres junto con una IRI especificando el tipo de dato al que corresponde.

Nodo en blanco Los nodos en blanco son nodos en un grafo RDF que representan un recurso al cual no se le ha asignado una IRI o un literal. Estos nodos son interpretados de manera que indican la existencia de algo, pero sin identificarlo.

URI (cuyas siglas significan Uniform Resource Identifier) es una cadena de caracteres (String) que identifica un recurso en una red, por ejemplo en la World Wide Web. Su resolución, a partir de protocolos definidos, permite obtener una representación del recurso identificado. Una URI puede ser tanto una URL (Uniform Resource Locator) como una URN (Uniform Resource Name).

URL En adición a identificar un recurso web, una URL especifica cómo obtener una representación de este, especificando tanto un mecanismo de acceso como su ubicación en la red.

URN Identifica un recurso por un nombre en un espacio de nombres particular. Un URN puede ser utilizado para referirse a un recurso sin involucrar su ubicación o como acceder a esta.

Además existe un estándar que extiende el esquema URI llamado IRI (Internationalized Resource Identifier).

IRI IRI extiende las URIs con el uso de Conjunto de Caracteres Universal (en inglés Universal Character Set o UCS), mientras que las URIs están limitadas solo al uso de un subconjunto de ASCII. Esto permite el uso de tildes o caracteres correspondientes a otros idiomas como el Chino, Japonés, entre otros.

2.1.2. RDF

Resource Description Framework (RDF) es un modelo de datos que representa información sobre recursos en el World Wide Web.

RDF [7] está diseñado para situaciones en las que se necesite que la información sea procesada por computadores. Así es posible que los recursos sean intercambiados por aplicaciones que utilicen este framework sin pérdida de significado.

El modelo de datos RDF toma como idea base las declaraciones de recursos como expresiones en forma de triples sujeto-predicado-objeto. Estos términos tienen como principal identificador IRIs, junto con nodos en blanco y literales, como se muestra en la tabla 2.1.

Existen múltiples formatos de serialización para RDF, los que incluyen:

1. Turtle, un formato compacto y de fácil interpretación para las personas.
2. N-Triples, delimitado por líneas, menos compacto que Turtle.

Sujeto	Predicado	Objeto
IRI	IRI	IRI
Nodo en blanco		Nodo en blanco
		Literal

Tabla 2.1: Términos para RDF.

3. JSON-LD, serialización basada en JSON, no completamente alineada con RDF.
4. N3 o Notation3, una serialización no estándar y similar a Turtle.
5. RDF/XML, sintaxis basada en XML, difícil interpretación para las personas.

Un ejemplo de una representación de una declaración en RDF es la que se muestra en la figura 2.1. De la frase 'Santiago es la capital de Chile' se toma como sujeto de la declaración a 'Santiago', el sujeto a 'capital de' y el objeto a 'Chile'. Debido a que RDF describe recursos en términos de propiedades y valores de propiedades es posible representar declaraciones y recursos en forma de grafos, con nodos representando tanto los recursos como valores y aristas representando sus propiedades. Un ejemplo de esto se puede observar en la última transición de la figura 2.1.

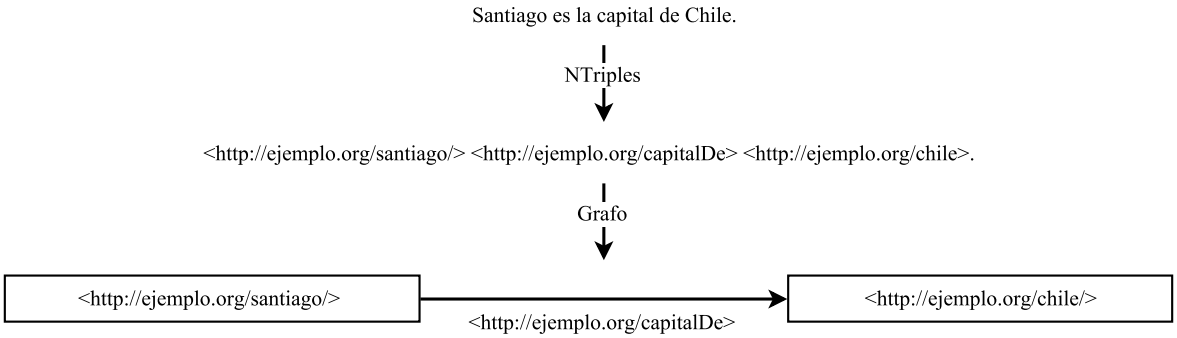


Figura 2.1: Ejemplo de una declaración transformada a RDF, primero como triple en formato NTriple, luego mostrada como grafo.

2.2. Wikidata

Wikidata [8] es una base de datos que puede ser leída y editada tanto por humanos como máquinas. Funciona como almacenamiento central de información estructural para proyectos de Wikimedia (entre otros incluye Wikipedia, Wikivoyage, Wikisource). Las principales características de Wikidata son las siguientes:

- **Es libre** La información es publicada bajo la licencia Creative Commons Public Domain Dedication 1.0, lo que significa que permite el reúso de la información en diferentes escenarios. Es posible hacer copias, modificar, distribuir y utilizar la información.
- **Colaborativa** Cualquier persona puede editar los datos disponibles.

- **Multilenguaje** Existen datos almacenados en más de 200 idiomas.
- **Es una base de datos secundaria** Es decir, también guarda fuentes y conexiones a otras bases de datos.
- **Estructura de datos definida** Wikidata utiliza una estructura de datos definida con artículos y propiedades.

2.2.1. Modelo de datos

El modelo de datos de Wikidata consiste en una colección de entidades. Las entidades son los elementos básicos y pueden ser tanto referenciados como descritos. Existen dos tipos de entidades: los artículos (items) y las propiedades (properties).

Un ejemplo de un artículo con cada sección de su estructura se puede ver en la figura 2.2. A continuación se detalla cómo se compone cada una de estas:

- Artículo
  - Identificador del artículo (número con prefijo Q)
  - Fingerprint, consiste en:
    - Etiqueta multilenguaje
    - Descripción multilenguaje
    - Alias multilenguaje
  - Declaraciones, cada una consistente en:
    - Reclamación, consistente en:
      - Propiedad
      - Valor
      - Clasificadores
    - Referencias
    - Rango
  - Enlaces a sitios
- Propiedad
  - Identificador de propiedad (número con prefijo P)
  - Fingerprint, al igual que los artículos.
  - Declaraciones, al igual que los artículos.
  - Tipo de dato

Artículo Los artículos (en inglés items) son como se definen en Wikibase los artículos de interés, normalmente corresponden a las cosas a las que se refieren los artículos de Wikipedia. Estos se identifican con una Q seguido de un número único.

Declaraciones Las declaraciones (en inglés statement) consisten de una propiedad (por ejemplo 'población') y de un valor (3,5 Millones) además de otros valores opcionales (calificadores y referencias).

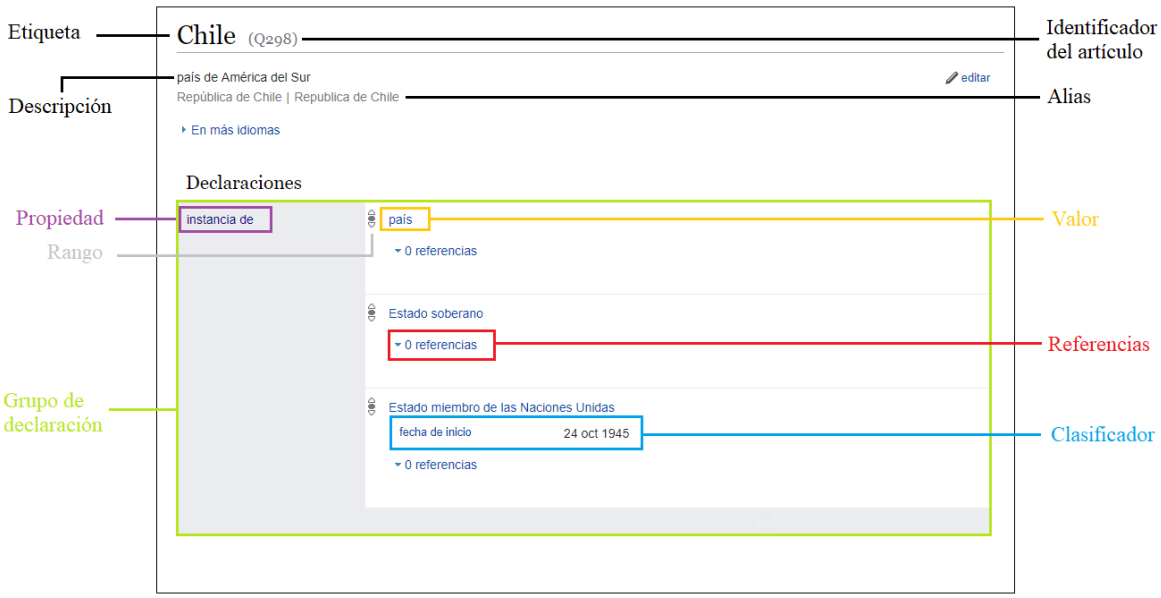


Figura 2.2: Ejemplo de un artículo de Wikidata.

2.2.2. Relación entre propiedades en RDF y en Wikidata

El modelo de datos de Wikidata al más simple nivel se empareja de buena forma con RDF, ya que la mayoría de la información está codificada por un item, una propiedad y un valor para esa propiedad, lo que sería el equivalente a un triple sujeto (rdf:subject), predicado (rdf:predicate) y valor (rdf:object) en RDF.

2.3. Algoritmos de búsqueda de caminos

Con respecto a la búsqueda de caminos mínimos para grafos se consideran principalmente los siguientes algoritmos: Algoritmo de Dijkstra y el algoritmo de búsqueda A*. A continuación se describe cada uno de ellos.

2.3.1. Algoritmo de Dijkstra

El algoritmo de Dijkstra [5] es un algoritmo que encuentra los caminos más cortos entre un nodo de inicio y cada uno de los nodos de un grafo. El primero en describir este algoritmo, y al que se debe su nombre, es Edsger Dijkstra en 1956. A partir de este algoritmo

Listing 2.1: Pseudocódigo del algoritmo de Dijkstra

```
1  función Dijkstra(grafo, inicio):
2    crear set de vértices Q
3    Por cada vértice v en el Grafo:
4      distancia[v] = infinito //distancia desde el nodo inicio a v
5      prev[v] = indefinido    //nodo previo a v en el camino
6      agregar v a Q           //nodos no visitados
7
8    mientras Q no este vacío:
9      u = el vértice en Q con la mínima distancia[u]
10     remover u de Q
11
12     por cada vértice vecino v de u:
13       dist_v = distancia[u] + distancia(u,v)
14       si dist_v < distancia[v]:
15         distancia[v] = dist_v
16         prev[v] = u
17
18   retornar distancia[], prev[]
```

existen distintas variaciones, como por ejemplo el algoritmo de búsqueda A* y el algoritmo de búsqueda de costo uniforme. Este algoritmo es usado en aplicaciones de distintas áreas, incluyendo búsqueda de rutas de caminos o protocolos de ruteo en redes. En Listing 2.1 se detalla el pseudocódigo correspondiente a este algoritmo.

En este pseudocódigo se inician los valores de las distancias entre cada nodo y el nodo de inicio con el valor infinito en la línea 4, luego se define el nodo anterior del camino en la línea 5. A continuación por cada nodo no visitado se remueve de la lista de nodos no visitados y por cada vecino de este se busca la distancia mínima al nodo de inicio, si es menor a la actual se re-define la distancia y se asigna el nodo previo. Finalmente se retornan las distancias y nodos previos.

La complejidad de este algoritmo es $O(|E| + |V|) = O(|V|^2)$, donde $|E|$ es el número de aristas y $|V|$ el número de vértices. Esto se debe a que se recorren $|V|$ vértices, cada vértice del grafo, y por cada uno de estos se busca el nodo que tiene distancia mínima al nodo de inicio, lo que está acotado por $|V|$.

Es posible utilizar cola de prioridades para extraer la mínima distancia (línea 9), que con una cola de prioridades implementada con un Heap de Fibonacci (Fredman & Tarjan 1984) se tiene una complejidad de $O((|E|+|V|)log|V|)$. Otro posible cambio, si solo se está interesado en la búsqueda del camino mínimo entre dos vértices i y f, es revisar en la línea 11 si el vértice u corresponde al vértice f y en caso de que así sea terminar de iterar para retornar el resultado. Esta variación del algoritmo de Dijkstra es también conocida como Búsqueda de costo uniforme (UCS).

2.3.2. Algoritmo de búsqueda A*

El algoritmo de búsqueda A* [2] es una variación del algoritmo de Dijkstra. Fue descrita por primera vez en 1968 por Peter Hart, Nils Nilsson y Bertram Raphael. La principal diferencia con Dijkstra es que al utilizar una función heurística para guiar el camino buscado y se cumplan determinadas condiciones se obtienen un mejor rendimiento.

En contraste con el algoritmo de Dijkstra, el parámetro para elegir el vértice u en la línea 9 del pseudocódigo deja de ser la función $g(n)$, donde $g(n)$ es la distancia mínima entre el nodo inicial y el nodo n. Ahora la función que se busca minimizar es $f(n) = g(n) + h(n)$, donde $g(n)$ es la descrita anteriormente y $h(n)$ es una función heurística que estima el costo entre el nodo inicial y el nodo n. Además la función $h(n)$ debe ser una heurística admisible, lo que significa que no debe sobrestimar el costo real de alcanzar el nodo objetivo. Finalmente, para que no sea necesario re-visitar nodos ya visitados se necesita que la función $h(n)$ sea monótona. De esta manera se garantiza que $f(n)$ sea el valor mínimo de n. Un ejemplo de una función heurística para la búsqueda de caminos en un mapa sería la distancia efectiva (la distancia directa) entre dos puntos.

La complejidad de este algoritmo varía según la calidad de la función heurística utilizada. Si la heurística es de mala calidad la complejidad puede ser exponencial, mientras que si la función es de buena calidad el algoritmo se puede ejecutar en tiempo lineal.

Este algoritmo es utilizado en variadas aplicaciones, como por ejemplo, para problemas de búsqueda de caminos en aplicaciones como video juegos o aplicaciones de procesamiento de lenguajes naturales.

2.4. Soluciones existentes

A continuación se describen soluciones existentes que buscan resolver el mismo o similares problemas al de encontrar caminos relevantes entre dos nodos en un grafo RDF. Además se explica el porqué de la necesidad de desarrollar nuevas ideas.

2.4.1. RelFinder

RelFinder [3] es una herramienta que busca relaciones entre dos conceptos y luego dibuja un grafo con todos los resultados obtenidos. Para lograr esto, la herramienta recibe dos conceptos como entrada por parte del usuario, luego busca estos conceptos en una base de datos en lenguaje RDF que tenga disponible un SPARQL endpoint.

Para llevar a cabo la búsqueda de relaciones realiza un proceso compuesto de varias consultas SPARQL. Ya que no se conoce el camino más corto entre ambos conceptos, el proceso de consultas consiste en realizar múltiples consultas iterando en el número de conexiones o saltos, partiendo desde largo cero.

Finalmente las relaciones encontradas son agregadas al grafo a dibujar una por una, comenzando desde la de menor largo. Los nodos encontrados son dibujados con su etiqueta correspondiente, mientras que las aristas contienen la información de la dirección de estas junto con su etiqueta. Un ejemplo del resultado de esta herramienta entre los conceptos 'Santiago, Chile' y 'Chile' se puede ver en la figura 2.3.

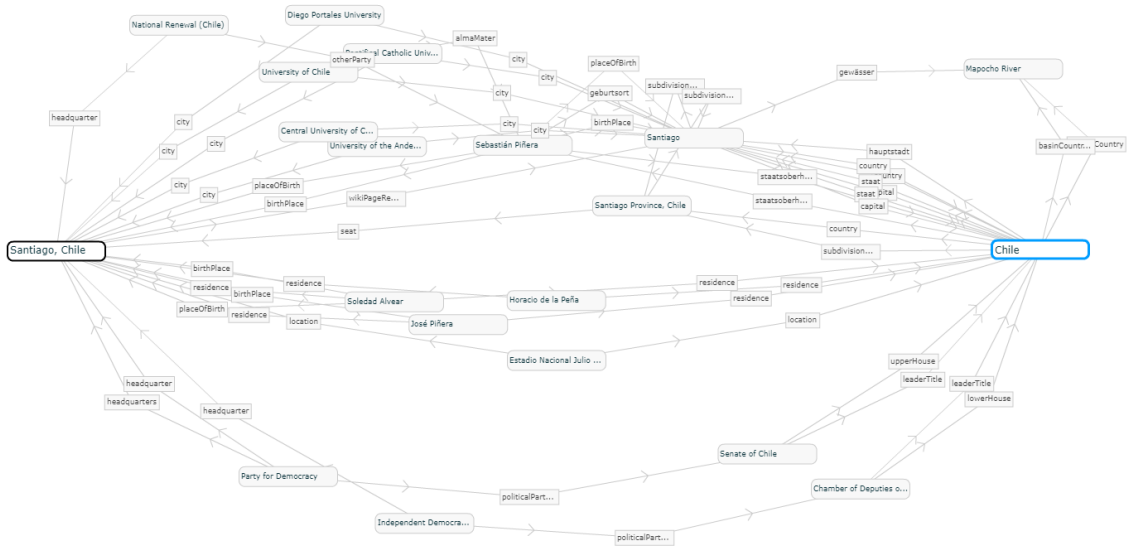


Figura 2.3: Ejemplo de búsqueda en la herramienta RelFinder [3], conceptos utilizados: Santiago, Chile y Chile.

Sí bien esta herramienta soluciona el problema de encontrar caminos entre dos conceptos, no le otorga valor a los nodos/aristas que recorre y solo se basa en el atributo relacionado al largo de los caminos a encontrar, a menos que se filtren manualmente los resultados.

2.4.2. gFacet

La idea básica de la herramienta gFacet [4] es visualizar y explorar la web de datos a través de una búsqueda por facetas y una visualización del grafo que forman. Esto lo logra por medio de la estructuración de un grafo en que los nodos corresponden a las facetas. Con la combinación de estos métodos es posible navegar por el grafo construido agregando facetas hasta encontrar las relaciones buscadas como se observa en la figura 2.4. La información en RDF es requerida a través de consultas SPARQL. El principal problema de esta solución es que los set de datos en RDF son comúnmente altamente interconectados, por lo que la búsqueda de un camino que relacione dos nodos puede tomar mucho tiempo, aún más si el usuario no tiene conocimiento previo de los conceptos que busca conectar.

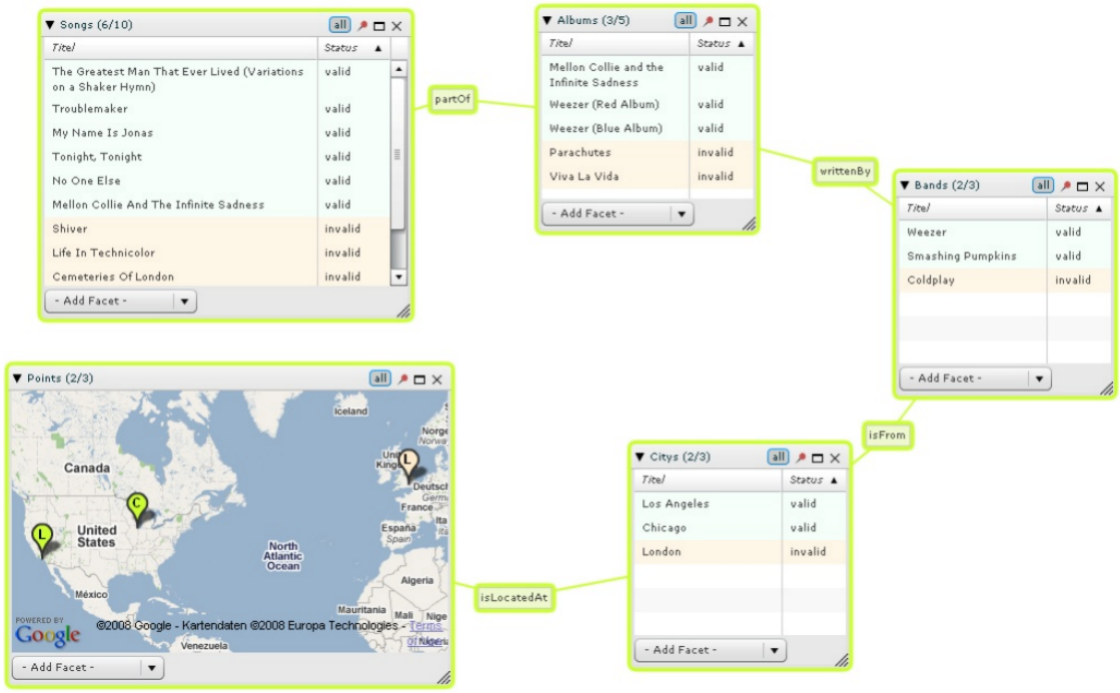


Figura 2.4: Ejemplo de búsqueda en la herramienta gFacet [4].

Capítulo 3

Especificación del problema

3.1. Descripción del problema

Con el desarrollo de tecnologías relacionadas con la web de datos (linked data) se ha producido un incremento en fuentes de información que utilizan RDF, como por ejemplo Wikidata. La representación estructurada de los datos abre nuevas formas en como esta puede ser obtenida o requerida, ya sea tanto por personas como por máquinas.

Una de las principales utilidades de guardar información por medio de datos enlazados es encontrar links o relaciones entre conceptos. Para lograr esto existen distintas estrategias, entre las más comunes se encuentran: Desde un punto de partida explorar incrementalmente el grafo siguiendo ciertas aristas, a partir del grafo completo comenzar a filtrar información o utilizar búsqueda por facetas (en el caso de gFacet).

El problema de estas soluciones es que requieren que el usuario explore o filtre manualmente los datos para encontrar la relación buscada, lo que para grafos con una gran cantidad de información puede resultar en un proceso que toma demasiado tiempo.

Otra estrategia para obtener relaciones entre conceptos utiliza algoritmos de búsqueda de caminos sin la necesidad de un usuario filtrando o navegando la información. Si bien herramientas (como el mencionado RelFinder) y software (como Neo4j y Gremlin) de estas características existen, estos no soportan grafos ponderados, por lo tanto la respuesta entregada se basa solo en el largo de los caminos encontrados.

Es por esto que la motivación de este trabajo es encontrar nuevas métodos para la búsqueda de caminos en un grafo RDF, de modo que el usuario obtenga una respuesta a través de un proceso simple, solo ingresando los nodos de interés, y que la búsqueda se base en características del camino más allá de su largo.

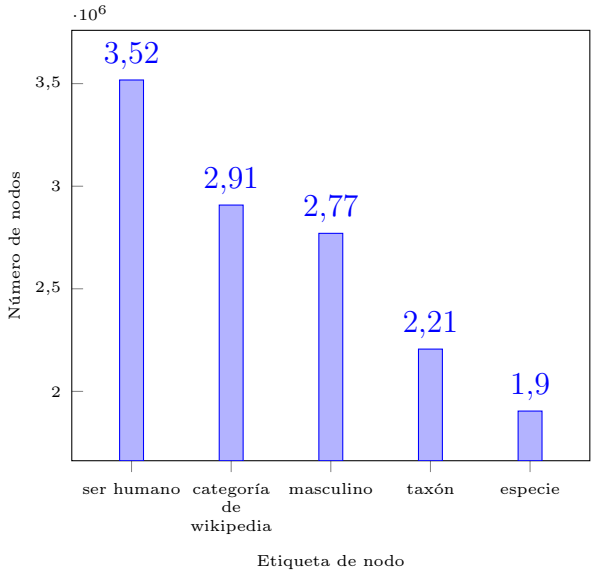
3.1.1. Problema de encontrar caminos en RDF

Existen distintos métodos para acceder a información en bases de datos en RDF. Considerando las herramientas y software estudiados es posible encontrar soluciones basadas en algoritmos que, ya sea a través de la aplicación de filtros o algoritmos iterativos, realizan consultas a un SPARQL endpoint o bien a partir de un archivo en uno de los formatos de serialización para RDF, crean un grafo en el cual se utilicen algoritmos de búsqueda de caminos mínimos.

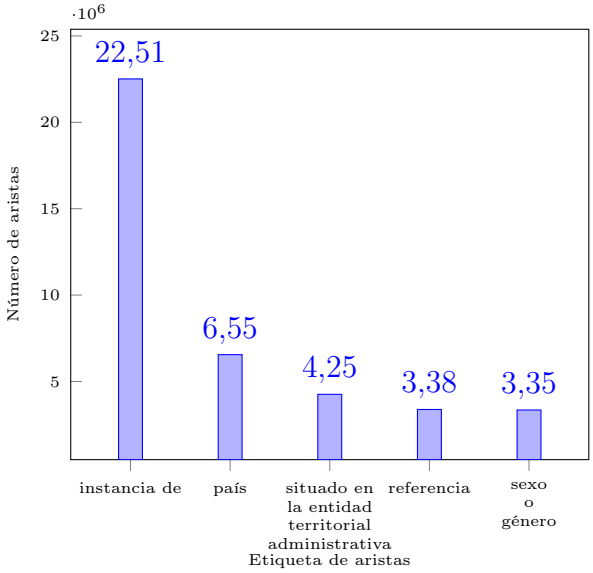
3.1.2. Problema de encontrar caminos interesantes

El problema se define como encontrar caminos entre dos nodos que son interesantes para un usuario. Definir de antemano, y de forma objetiva, cuáles caminos son interesantes y cuales no parece como una idea casi imposible e incluso, no es muy claro qué tan variante sería entre usuarios diferentes o aplicaciones diferentes. En lugar de una definición exacta de un camino interesante, seguiremos algunas intuiciones para plantear varias estrategias que evaluaremos con usuarios para lograr algunos resultados iniciales que indiquen sus preferencias.

Para empezar, uno de los problemas a solucionar es que la propiedad en la que se basan las búsquedas de caminos no sea solo minimizar el largo del camino. Como se ilustro en el ejemplo de caminos en la figura 1.1, debido a las características descriptivas de las bases de conocimiento en formato RDF, existen nodos que conectan una gran cantidad de conceptos y enlaces con la misma etiqueta que relacionan una gran cantidad de conceptos provocando que los caminos resultantes sean escasos en información relevante para el usuario. En la figura 3.1 se ven los 5 nodos con mayor grado y las 5 aristas que unen la mayor cantidad de conceptos. Como se puede apreciar el nodo ‘ser humano’ posee un grado muy alto y la arista con etiqueta ‘instancia de’ tienen una gran cantidad de apariciones, es por esto que, si por ejemplo, se intentara buscar un camino entre ‘persona 1’ y ‘persona 2’ solo minimizando el largo del camino, las relaciones resultantes serían siempre las mismas: ‘persona 1’ es una instancia de ‘ser humano’ y ‘persona 2’ es una instancia de ‘ser humano’.



(a) Los 5 nodos con mayor grado.



(b) Las 5 aristas que unen la mayor cantidad de conceptos.

Figura 3.1: Datos obtenidos de un dump de Wikidata con 25.081.334 nodos y 89.878.092 aristas.

Capítulo 4

Descripción e implementación de la solución

En este capítulo se describe el diseño e implementación de la solución al problema descrito. A continuación se descompone la solución en tres partes, las que abarcaran distintos aspectos relacionados con la herramienta a construir. Estas tres partes corresponden a los datos utilizados, a la búsqueda de caminos y a la aplicación para visualizar los resultados.

4.1. Datos utilizados

Para la elección de los datos a utilizar, tanto para las pruebas como en la versión final de la aplicación, se elige la base de conocimiento de Wikidata. Las características por las que se decide por la utilización de la información de esta fuente son las siguientes:

- **Tamaño** Cuenta con una gran cantidad de elementos, hasta la creación de este documento son 39.589.583 elementos de datos.
- **Libre** El contenido de Wikidata está disponible bajo una licencia libre.
- **Formato** La información es exportada utilizando distintos formatos estándares. En particular para este trabajo, existen formatos de serialización para RDF.
- **Datos enlazados** La información pueden ser enlazados a otros conjuntos de datos abiertos en la web de datos.

Dump de Wikidata

Los datos utilizados corresponden a una copia de la información disponible en Wikidata para cualquier persona que desee descargarla. La fecha del dump utilizado es el 7 de junio del 2017. El formato de serialización del archivo descargado es N-Triples y contiene un total de 1.211.813.217 triples.

4.2. Búsqueda de caminos

Para el desarrollo de los algoritmos orientados a resolver el problema de encontrar caminos que relacionen dos conceptos a partir de un archivo con información en RDF se utiliza como lenguaje de programación JAVA (versión jdk1.8.0_60), para los test de funcionalidades se utiliza JUnit 4 (versión junit_4.12.0.v201504281640) y como IDE (integrated development environment) se utiliza Eclipse (versión Mars.2 Release (4.5.2)). A continuación se describen las componentes implementadas.

4.2.1. Grafo RDF

Procesamiento y manipulación de datos RDF

Para el procesamiento y manipulación de los datos RDF se utiliza el framework RDF4J (versión RDF4J 1.0.3 onejar). RDF4J soporta todos los principales formatos de serialización para RDF, en particular el formato N-Triples correspondiente al formato de los datos obtenidos. De las funcionalidades que ofrece este framework se utilizan la lectura y parseo de los datos como se explica a continuación.

En la figura 4.1 se observa un diagrama de las componentes de RDF4J que se utilizan para el parseo de los datos en RDF. Inicialmente, se crea un RDFParser al que se le asigna un formato para el que se utilizará, para este trabajo se utiliza RDFFormat.NTRIPLES. Luego se asigna un RDFHandler que corresponde al componente encargado de manipular los datos a leer. Finalmente, se parsea un InputStream por el cual se leen los datos del archivo RDF.

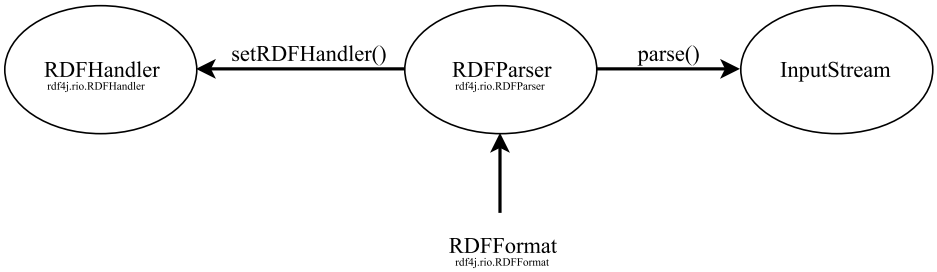


Figura 4.1: Diagrama con funcionalidades utilizadas de RDF4J.

RDFHandler recibe el flujo de triples correspondiente a los datos en RDF. Aquí es posible realizar el procesamiento de los datos, como por ejemplo filtrar, descomponer el triple (en sujeto, predicado y objeto), escribir los datos en otro archivo, entre otros.

Datos a utilizar

Como se señaló en la sección 2.2.1 existen dos tipos de entidades: artículos y propiedades. Las URIs correspondientes a estas entidades son las que se muestran en la figura 4.2, donde

el identificador de los artículos es un número único con prefijo Q y el identificador de las propiedades es un número con prefijo P.

Artículo: <http://www.wikidata.org/entity/ID>
Propiedad: <http://www.wikidata.org/prop/direct/ID>
Etiqueta: <http://www.w3.org/2000/01/rdf-schema#label>

Figura 4.2: URIs correspondientes a artículos, propiedades y predicados para etiquetas.

Otro identificador que se utiliza en este trabajo es el predicado correspondiente a las etiquetas. Para este caso el triple está compuesto por un sujeto el que puede ser tanto un artículo como una propiedad, un predicado correspondiente a la etiqueta de la figura 4.2, y un objeto que en este caso corresponde a un literal con la etiqueta correspondiente.

Subconjuntos de datos a utilizar

Debido al gran tamaño de datos que contiene la base de conocimiento de Wikidata y a que, debido al alcance del trabajo definido, solo se trabaja con datos en memoria principal es que se resuelve por crear subconjuntos de distintos tamaños del total de datos disponible. Estos sets de datos son utilizados tanto para la medición del rendimiento de algoritmos como para la comparación de resultados a obtener.

A continuación se detalla la forma en que se definen estos subconjuntos de datos. Como hemos visto cada artículo tiene un ID único que lo identifica, estos IDs consisten en números antecedidos por la letra Q. La forma en que se asignan los IDs a los artículos es aumentando el número que los forma cada vez que se crea un nuevo artículo. Por lo tanto cada subconjunto está formado por los triples en que tanto el sujeto como el objeto son menores a un cierto número. Debido a que las entidades más importantes son definidas primero, y que se asignan IDs en orden ascendiente, al construir grafos con un límite de ID se obtiene un grafo constituido por las entidades más importantes. Además, gracias a la construcción incremental de los datos de Wikidata es que los grafos que se construyen son en su mayoría conexos (con excepción de nodos que son modificados o eliminados).

Estructura del grafo

Como se observa en la figura 4.3 se tienen tres clases que contienen la información de un grafo. Estos son Graph, Vertex y Edge. A continuación, se describen sus variables y funciones (se ignorarán los setters y getters):

Graph Esta clase representa un grafo. La variable nodes corresponde a los nodos que forman el grafo. Las funciones que se señalan en la figura corresponden a métodos para calcular los pesos de los nodos.

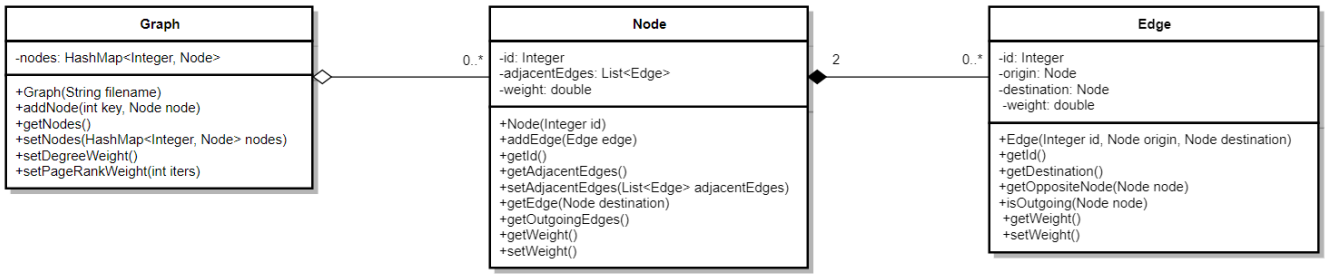


Figura 4.3: Diagrama con las clases que forman la estructura de los grafos.

Para la construcción del grafo se utiliza RDFParser como se señaló en la sección 4.2.1. En el handler se filtran los triples para crear el grafo como sigue:

- Se descompone cada declaración contenida en un triple en su sujeto, predicado y objeto.
- Los nodos del grafo serán los artículos (por ejemplo Barack Obama, Chile, Google, etc.), por lo tanto se filtran los triples donde tanto el sujeto como el objeto correspondan a un artículo. Además, si se trata de un subconjunto del total de datos se verifica que los IDs del sujeto y objeto sean menores al ID máximo que se utiliza para el subconjunto.
- El sujeto y el objeto son creados si no existían.
- Se crea la arista que une los nodos, el cual corresponde al predicado del triple (del tipo propiedad en Wikidata).

Node Esta clase representa a los nodos. Los atributos y funciones de esta clase son:

- id: Corresponde al número identificador del artículo.
- adjacentEdges: Son las aristas adyacentes al nodo.
- weight: Es el peso del nodo.
- getOutgoingEdges(): Retorna las aristas que tienen dirección saliente al nodo.

Edge Esta clase representa las aristas. Los atributos y funciones de esta clase son:

- id: Corresponde al número identificador del predicado.
- origin: Es el nodo origen de la arista.
- destination: Es el nodo destino de la arista.
- weight: Es el peso de la arista.
- isOutgoing(node): Indica si la dirección es saliente al nodo o no.

4.2.2. Algoritmo de Dijkstra

Para la implementación de algoritmos de búsqueda de caminos mínimos se descartó la utilización de A* debido a la falta de una función heurística que cumpliera con las caracterís-

ticas necesarias para obtener un mejor rendimiento a no utilizar de una heurística (Dijkstra). Es por esto que, para llevar a cabo este objetivo, el trabajo se centró en la implementación del algoritmo de Dijkstra con las características del grafo implementado.

Implementación

La implementación del algoritmo de Dijkstra es similar a la descrita en el pseudocódigo de la sección 2.3.1, con las siguientes modificaciones:

Cola de prioridad Para el almacenamiento de las distancias entre el nodo de inicio y el resto de los nodos en el grafo se utiliza una cola de prioridad en lugar de una lista. De esta manera, al momento de tomar el nodo no visitado con menor distancia al nodo de inicio solo es necesario tomar el nodo con mejor prioridad de la cola (menor distancia).

Nodo destino El algoritmo de Dijkstra tiene como objetivo encontrar todos los caminos más cortos desde un nodo inicial a todos los nodos del grafo. Como el objetivo de nuestra implementación es solo encontrar el camino mínimo entre dos nodos es que se adiciona un nodo destino al cual queremos llegar. Para esto, cambiamos la condición de término del loop en el que buscamos los vértices mínimos del set de vértices no visitados. Si bien en el algoritmo original la condición de término es que el set de vértices no visitados no esté vació, ahora agregamos la condición: si el nodo en el que estamos parados (para actualizar las distancias de los nodos vecinos a este) es igual al nodo destino se termina el loop. De esta manera el algoritmo se detiene en el nodo destino, es decir se encuentra la distancia mínima entre los dos nodos y se cuenta con los nodos sucesores al nodo destino que forman este camino.

Función distancia Debido a que el grafo ponderado a utilizar cuenta con pesos tanto en los nodos como en las aristas, dependiendo del método que se utilice para calcular el valor total del camino, este puede variar entre la distancia actual del nodo en el que nos encontramos más el valor del nodo vecino o la distancia actual más el peso del nodo vecino junto con el peso de la arista que los une.

4.2.3. Ponderación de grafos

En vista de que la búsqueda de los caminos se basa en minimizar la suma de los pesos de tanto nodos como aristas es que se diseñaron nueve variaciones de ponderación. Estas ponderaciones se obtienen a partir de tres métodos, los que se describen a continuación:

Grado de los nodos

Los grafos en RDF son altamente conexos, debido a su naturaleza descriptiva existen nodos con un grado muy alto, es decir conectan un gran número de conceptos. Con el fin de evitar estos nodos, y en consecuencia que los caminos buscados entre distintos pares de conceptos sean diferentes, es que a los nodos se les asigna un peso en relación a su grado.

PageRank

PageRank [6] es un algoritmo recursivo usado por Google para dar un rango (rank) a las páginas web en su motor de búsquedas. PageRank es una forma de medir la importancia de una página web, su definición según Google es la siguiente:

> *"PageRank trabaja contando el número y calidad de enlaces a la página para determinar un estimado aproximado de que tan importante la página web es. La suposición subyacente es que para las páginas web más importantes es más probable que reciban enlaces de otras páginas web.*[1] *"*

Este método no se basa solo en contar los enlaces que apuntan al nodo a evaluar sino que también en la importancia del nodo del que provienen, es decir:

- Un enlace proveniente de una página más importante es un enlace más importante.
- Un enlace proveniente de una página con menos enlaces es un enlace más importante.

Por lo tanto, una página con muchos enlaces entrantes desde páginas importantes (que tienen pocos enlaces salientes) es más importante.

Si se modelan las páginas web y sus enlaces como un grafo $G = (V, E)$ donde los vértices V son las páginas web y las aristas E son los enlaces, PageRank se calcula de la siguiente manera:

Definición 4.1 *Dado un grafo dirigido $G = (V, E)$, PageRank se define:*

$$rank_{\mathrm{i}}(v) := \mathrm{d} \times \sum_{u \in in(v)} \frac{rank_{\mathrm{i}-1}(u)}{|out(u)|} + \sum_{v' \in V'} \frac{rank_{\mathrm{i}-1}(v')}{|V|} + (1 - \mathrm{d}) \times \sum_{v'' \in V''} \frac{rank_{\mathrm{i}-1}(v'')}{|V|}$$

Donde d es un factor de amortiguación (típicamente $\mathrm{d} = 0,85$), v es un nodo $v \in V$, $rank_{\mathrm{i}}(v)$ es el valor de PageRank del nodo v para la iteración i y se tiene:

$$out(v) := \{v' \in V : (v, v') \in E\}$$
$$in(v) := \{v' \in V : (v', v) \in E\}$$

[1] https://web.archive.org/web/20111104131332/https://www.google.com/competition/howgooglesearchworks.html

$$rank_0(v) := \frac{1}{|V|}$$
$$V' := \{v \in V : |out(v)| = 0\}$$
$$V'' := \{v \in V : |out(v)| \neq 0\}$$

Con el fin de determinar la importancia de los nodos en nuestro grafo RDF es que se utiliza PageRank. Puesto que los grafos RDF son multi-grafos con dirección al igual que el grafo formado por las páginas web y sus enlaces es posible utilizar la definición 4.1, donde los nodos corresponden a artículos y las aristas a propiedades. Es decir, dado un grafo RDF, usaremos el grafo dirigido $G = (V, E)$ donde V es el conjunto de todos los sujetos y objetos en el grafo RDF y $(v, v') \in E$ si y solo si existe un predicado p tal que (v, p, v') está en el grafo RDF. Para este trabajo tanto los sujetos como objetos corresponden a IRIs de artículos.

Etiquetas de aristas

El caso de las aristas con etiquetas es similar al de artículos que conectan una gran cantidad de nodos en el sentido de que al tener un multi-grafo existen aristas con etiquetas que describen un gran número de artículos, como por ejemplo '*instancia de*'. Para evitar que estas aristas se repitan en los distintos caminos resultantes es que se les asigna un peso como sigue:

$$w(\mathrm{e}) = 1 + \frac{|\mathrm{e}|}{n}$$

Donde $w(\mathrm{e})$ es el peso de las aristas con etiqueta e, $|\mathrm{e}|$ es el número de aristas con la etiqueta e y n es la etiqueta con mayor número de aristas, es decir es el número de aristas con la misma etiqueta normalizado.

Normalización de pesos

A partir de las pruebas iniciales se observó que los caminos resultantes utilizando los métodos de PageRank y el grado de los nodos eran más largos de lo esperado. Esto porque el valor del peso de los nodos variaba demasiado, provocando que los caminos se extendieran al intentar evitar nodos con gran peso. Con el fin de normalizar el peso de los nodos obtenidos con el método PageRank o el grado de los nodos se utiliza la siguiente formula:

$$w_n(n) = 1 + \frac{w(n)}{w_{max}}$$

Donde $w_n(n)$ es el peso normalizado del nodo n, $w(n)$ es el peso del nodo n y w_{max} es el peso máximo de todos los pesos de nodos. El objetivo de esta normalización es que el largo de los caminos influya en la búsqueda del camino con menor peso. Para lograr esto el rango

del peso de los nodos ahora será $]1, 2]$ por lo tanto al agregar un nodo al camino el peso del camino aumentará en al menos uno independiente del peso no normalizado del nodo.

Variaciones para la ponderación de grafos

Finalmente las variaciones resultantes son:

Variación	Peso de nodos	Peso de aristas
Caso base [B]	1	0
Grado [G]	Grado de cada nodo	0
Grado y peso de arista [GA]	Grado de cada nodo	Relacionado con etiqueta de cada arista
Grado normalizado [GN]	Grado de cada nodo normalizado	0
Grado normalizado y peso de arista [GNA]	Grado de cada nodo normalizado	Relacionado con etiqueta de cada arista
PageRank [P]	PageRank de cada nodo	0
PageRank y peso de arista [PA]	PageRank de cada nodo	Relacionado con etiqueta de cada arista
PageRank normalizado [PN]	PageRank de cada nodo normalizado	0
PageRank normalizado y peso de arista [PNA]	PageRank de cada nodo normalizado	Relacionado con etiqueta de cada arista

4.2.4. Implementación de índice invertido

Hasta ahora la construcción del grafo y búsqueda de caminos solo contiene información sobre los identificadores de nodos y aristas, esto debido a la gran cantidad de datos y a que se trabaja solo en memoria principal. Una vez obtenidos los caminos resultantes es necesario un mecanismo para extraer los datos necesarios para los nodos y aristas. Para conseguir esto se implementa un índice invertido; se le llama invertido ya que se intenta hacer un mapping desde palabras a documentos (normalmente un índice hace un mapping de documentos a

palabras).

Lucene

Apache Lucene es un proyecto open source de libre acceso que permite la implementación índices de alto rendimiento y escalables. Esta librería escrita en JAVA permite construir un motor de búsqueda de texto sobre documentos. Para este trabajo se utiliza Apache Lucene versión 5.5.0.

El primer paso para la construcción del índice es tomar la información necesaria a partir del dump con los datos de Wikidata. Con este propósito se utiliza un RDFParser con un handler que filtra todos los triples. Como se ve en la figura 4.4 los triples a filtrar tienen como predicado a 'http://www.w3.org/2000/01/rdf-schema#label', este predicado describe la etiqueta del sujeto. En el objeto del triple se encuentra un literal con la etiqueta que corresponde. Ya que los artículos tienen diferentes etiquetas en distintos idiomas, en el literal se encuentra un tag que especifica el idioma en que el string se encuentra. Debido a las múltiples etiquetas por artículo es que se filtra el literal correspondiente al objeto del triple para solo utilizar un idioma.

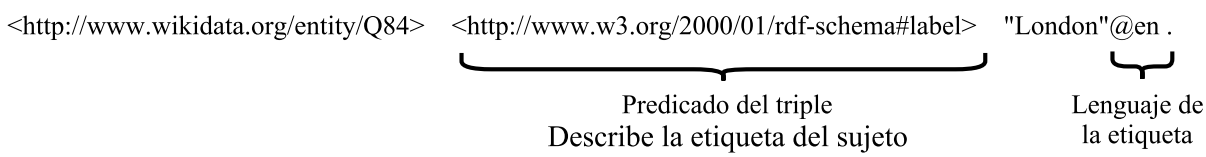


Figura 4.4: Ejemplo con triple que describe la etiqueta de un artículo.

Luego para la creación del índice a partir del archivo con los datos RDF filtrados se configura un escritor de índice de Lucene con la versión que se desea utilizar junto con un analizador en el idioma correspondiente. Finalmente, se definen los campos a extraer del documento, en este caso el predicado es el mismo en todos los triples por lo tanto solo se guardan el sujeto (el URI con el identificador) y el objeto (el literal con el string).

Con respecto a las consultas al índice se siguen los siguientes pasos:

- Se abre un lector (IndexReader de Lucene) en el directorio del índice construido.
- Se abre un buscador (IndexSearcher de Lucene) sobre el lector.
- Se utiliza el mismo analizador ocupado en la creación del índice.
- Se configura un parser (MultiFieldQueryParser de Lucene) para las consultas utilizando el analizador y los campos que se desean consultar.
- Se parsean las consultas y se buscan con el buscador abierto.

4.3. Aplicación

Con el fin de visualizar los resultados obtenidos, utilizar la herramienta desarrollada en un contexto real, realizar evaluaciones y reportar resultados se construye una aplicación web. A continuación, se discute el diseño y la implementación de cada uno de los módulos que la conforman.

4.3.1. Diseño

Uno de los objetivos de la aplicación es que sea accesible por los usuarios en un dominio público para su uso. Como se observa en la figura 4.5 la aplicación fue alojada en un servidor como un servicio web. Las principales componentes de esta herramienta son un servicio web que incorpora los algoritmos descritos en la sección 4.2 para la búsqueda de caminos, una base de datos para almacenar las evaluaciones de los usuarios a las distintas variaciones para ponderar el grafo y la aplicación que recibe las consultas y gráfica los resultados.

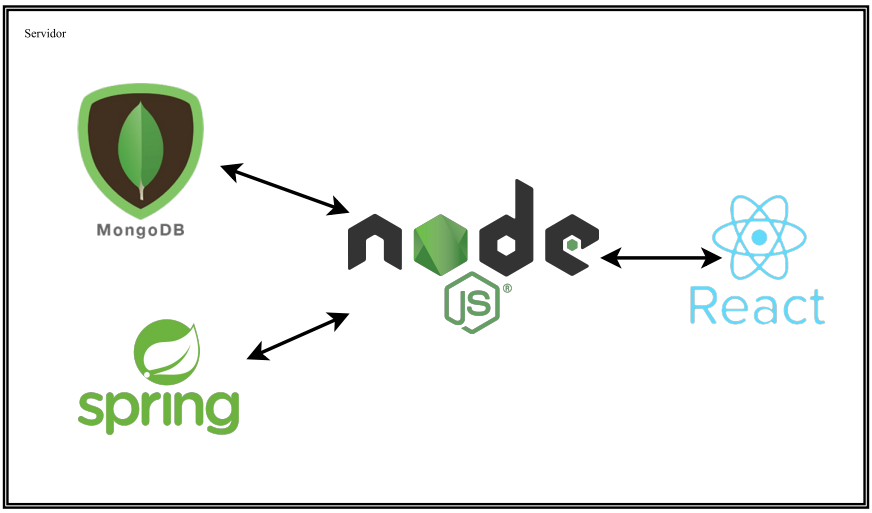


Figura 4.5: Tecnologías utilizadas para cada componente de la aplicación.

4.3.2. Servicio web RESTful

Con el fin de implementar una aplicación simple que aloje los algoritmos desarrollados en la sección 4.2 es que se construye un servicio web RESTful.

Un servicio web RESTful es construido para trabajar en un entorno Web. En el estilo de

arquitectura REST, los datos y funcionalidad son considerados recursos y son accedidos utilizando URIs, típicamente a través de la Web. Este estilo de arquitectura está restringido a una arquitectura cliente/servidor y está diseñado para usar protocolos de comunicación sin estado, típicamente HTTP.

Para la construcción del software se utilizó Maven, para la creación del servicio que recibe las HTTP requests y genera las respuestas se utilizó Spring framework como se ve en la figura 4.6. Los controladores creados que realizan el mapping de las requests fueron dos: DijkstraSearchController y IndexKeywordSearchController.

DijkstraSearchController DijkstraSearchController recibe requests para /dijkstraSearch y hace un mapping de los siguientes parámetros:

- **start:** El ID del nodo inicial.
- **end:** El ID del nodo de término.
- **weightedEdges:** Un boolean que determina si se consideran los pesos de las aristas.
- **nodeWeight:** El método para ponderar los nodos.

IndexSearchController IndexSearchController recibe requests para /IndexSearch y hace un mapping de los siguientes parámetros:

- **URI:** La URI del artículo o propiedad al que se desea retornar la etiqueta.

Luego estos controladores utilizando los algoritmos explicados en 4.2 retornan la respuesta generada.

4.3.3. Base de datos

Para el almacenamiento de las evaluaciones a las distintas variaciones de ponderación al grafo se utiliza una base de datos en MongoDB. MongoDB es una base de datos gratis, open source, clasificada como NoSQL y orientada a documentos.

El uso de la base de datos en este trabajo solo está acotada al almacenamiento de evaluaciones, por lo tanto, el esquema utilizado es simple. Este lo forma una colección de documentos llamada 'scores' que contiene documentos con los campos de la figura 4.7.

4.3.4. Aplicación

La aplicación web se construyó utilizando Node.js, npm para la administración de paquetes y la librería JavaScript ReactJS para la interfaz de usuario. Se utilizó el framework Express para construir la aplicación web.

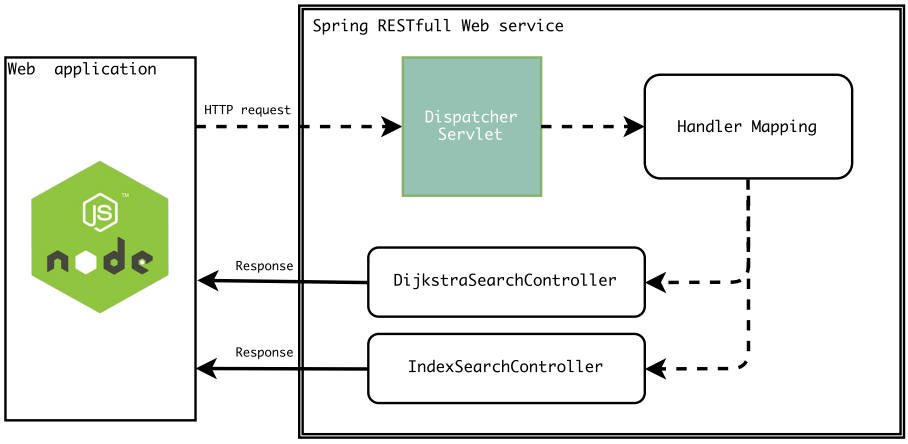


Figura 4.6: Spring RESTful web service workflow.

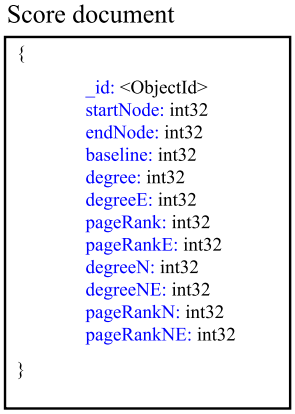
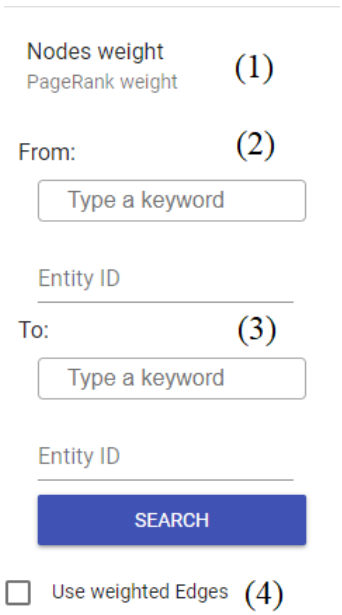


Figura 4.7: Schema de documentos en la colección scores.

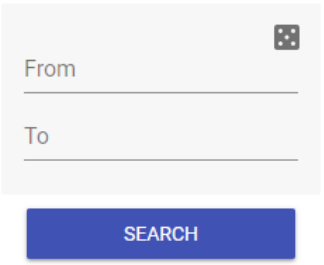
La aplicación web se encarga de recibir las consultas de los usuarios. Como se observa en la figura 4.8a, para esto existe un formulario en la interfaz de usuario donde (1) se selecciona el método para ponderar los nodos del grafo, (2) se ingresa el nodo de partida del camino a buscar (se puede ingresar su ID o buscar por el label), (3) ingresar el nodo de termino del camino y (4) se selecciona si se desea utilizar aristas ponderadas.

En la figura 4.8b se observa el formulario para búsqueda de caminos en todas las variaciones de ponderación de grafos implementadas.

Para la búsqueda de artículos por etiqueta se utiliza un input con auto-completado, la



(a) Búsqueda con una sola variación de ponderación de grafos.



(b) Búsqueda con todas las variaciones de ponderación de grafos.

Figura 4.8: Formularios para búsqueda de caminos.

lista de etiquetas se obtiene utilizando la API de Wikidata.

Una vez que se han ingresado los datos para la búsqueda y se presiona el botón de búsqueda se realiza un requerimiento HTTP al servicio web y se espera una respuesta. Si se recibe un error se muestra el error encontrado al usuario, si no se realiza un nuevo requerimiento con las etiquetas de los nodos y aristas para finalmente dibujar el camino resultante.

4.3.5. Visualización

Para la visualización de los grafos se utilizó la librería vis.js. En específico se utilizaron los módulos relacionados con redes, la que crea una visualización que despliega redes consistentes en nodos y aristas.

Como se observa en la figura 4.9 los caminos son dibujados como nodos con etiquetas y aristas con dirección y etiquetas. Además, es posible realizar un acercamiento/alejamiento en los caminos en caso de que su largo dificulte su observación y es posible mover los nodos en caso que las etiquetas tengan un largo que las superposicione.

Si se desea también es posible realizar búsqueda entre dos conceptos con todas las variaciones para ponderar los grafos como se observa en la figura 4.10. En este caso se despliegan nueve caminos correspondientes a cada variación, además se puede seleccionar una evaluación para cada camino la que es guardada en la base de datos de la aplicación

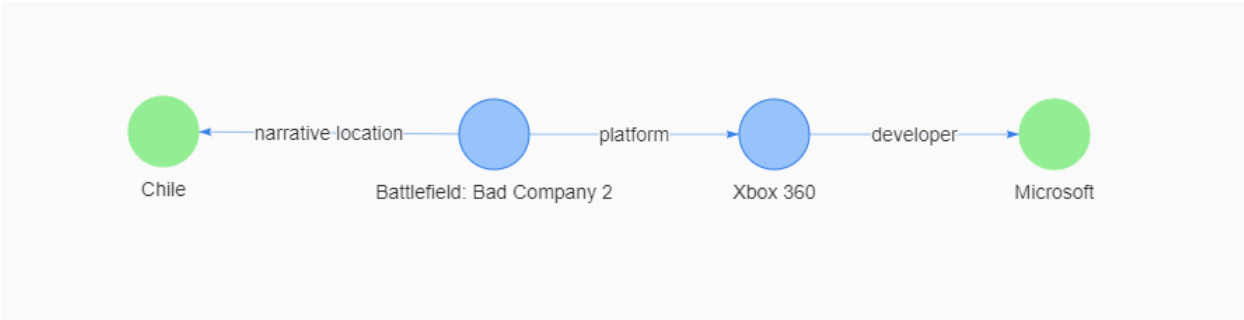


Figura 4.9: Ejemplo de camino entre Chile y Microsoft.

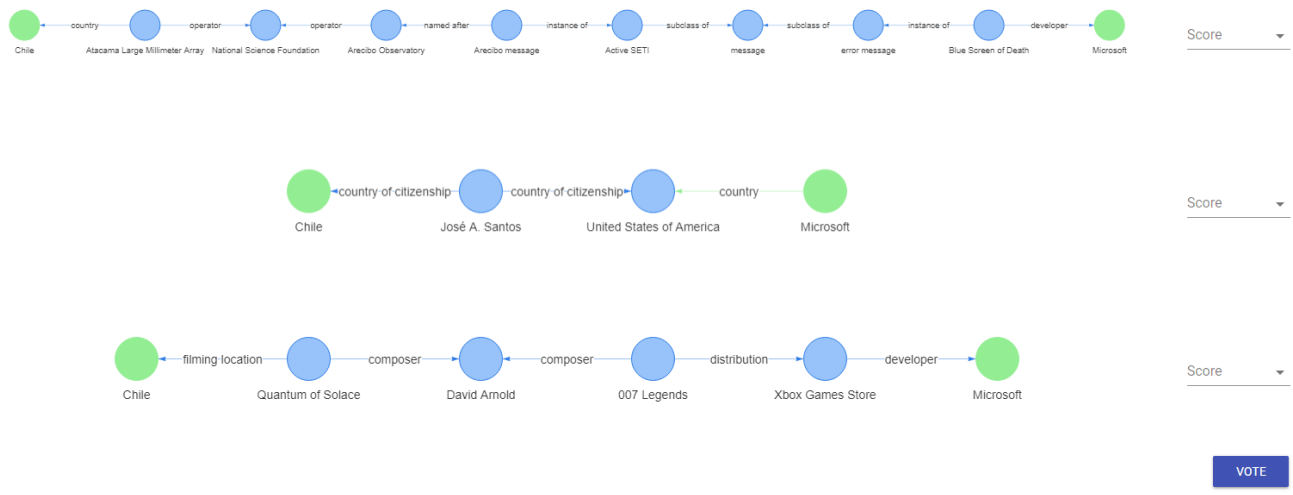


Figura 4.10: Ejemplo de múltiples caminos entre Chile y Microsoft.

Capítulo 5

Resultados

En este capítulo se exponen y analizan los resultados obtenidos en el trabajo realizado. El capítulo se compone de dos secciones, las que abarcan los resultados de experimentos realizados y una evaluación con usuarios sobre las distintas variaciones de ponderación de grafos utilizadas.

La principal pregunta que se busca contestar es:

- ¿Cuáles estrategias de ponderación de grafos producen caminos más relevantes en la estimación de los usuarios?

Además, con el fin de alcanzar un mejor entendimiento del problema y conocer la escalabilidad del algoritmo utilizado, se pretende contestar las siguientes interrogantes:

- ¿Cómo varían los caminos entre las estrategias? ¿Cuáles características (por ejemplo, largo del camino) tienen? ¿Cómo varía el largo de los caminos con distintas muestras de datos?
- ¿Cómo es el rendimiento del algoritmo de Dijkstra? ¿Cómo varía el rendimiento con respecto al tamaño de los datos? ¿Cómo varía el rendimiento entre las distintas estrategias de ponderación de grafos?

Las siguientes pruebas y evaluación pretenden generar respuestas iniciales a estas preguntas.

5.1. Resultados de pruebas

Para el estudio de los algoritmos implementados para la búsqueda de caminos en un grafo RDF se realizaron una serie de pruebas. El objetivo de estas pruebas es constatar el rendimiento de estos algoritmos y la comparación de los caminos resultantes de las distintas ponderaciones de grafos.

Las características de la máquina utilizada para la realización de las pruebas son las siguientes: 2× Intel Xeon Quad Core E5-2609 V3 CPUs, 32GB de RAM, y 2× 2TB Seagate 7200 RPM 32MB Cache SATA hard-disks en una configuración RAID-1.

Para la construcción de los gráficos se utilizaron las siguientes abreviaciones que señalan las distintas variaciones para la ponderación de grafos: Caso base - B, Grado - G, PageRank - P, Pesos de nodos normalizados - N y Peso de aristas A. Por ejemplo, GNA se refiere a un grafo ponderado con la variación Grado normalizado y con peso de aristas.

5.1.1. Grafos utilizados

Dado que el grafo de Wikidata es muy grande (formado por 25.081.334 nodos y 89.878.092 aristas) y que el proceso de encontrar caminos puede ser costoso, se eligieron muestras de Wikidata de varios tamaños para ver el rendimiento del algoritmo con distintos niveles de escala. En la figura 5.1 se observan los nodos y aristas de los grafos a utilizar en las pruebas. Estos grafos se obtienen tomando todos los nodos con identificador menor a un número como se explicó en la sección 4.2.1.

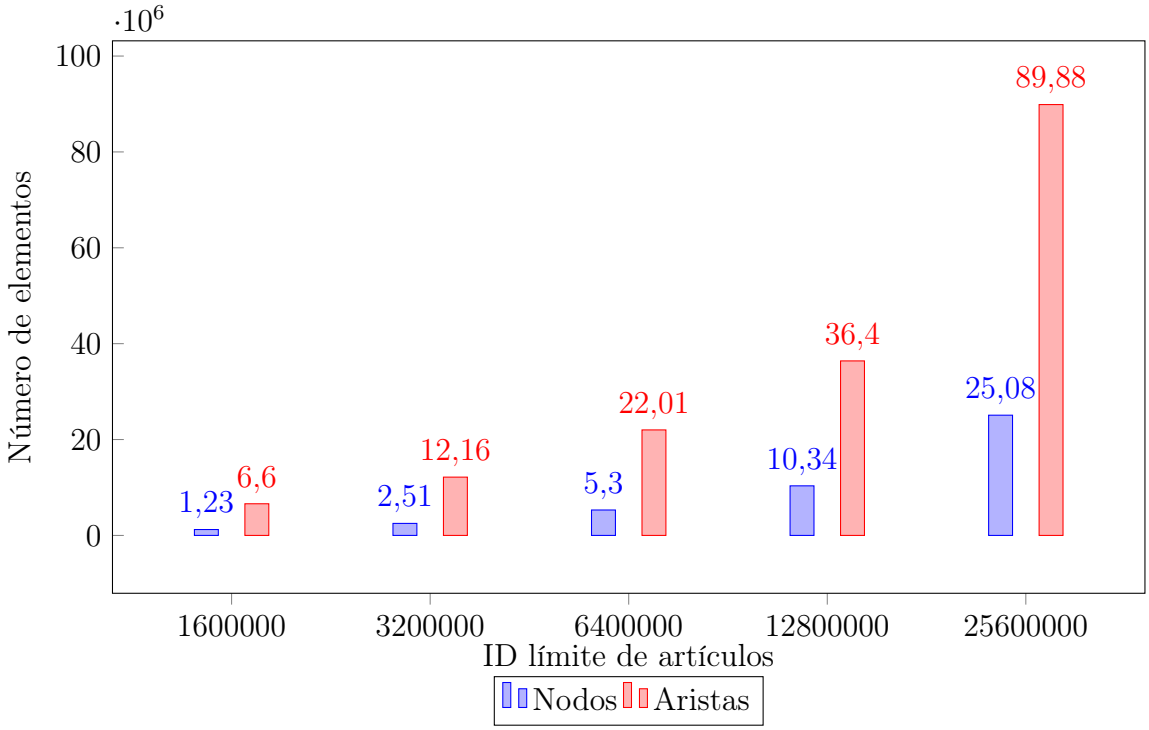


Figura 5.1: Número de nodos y aristas por grafo.

La cantidad de nodos por grafo es similar al número de identificador límite utilizado para los artículos. Esto se debe a que los números identificadores van aumentando a medida que se agregan nuevos artículos en Wikidata. La cantidad de nodos de los grafos es menor al ID límite fijado debido a que la información en Wikidata es modificable, por lo que hay artículos que pueden ser modificados o eliminados provocando la existencia de IDs que no contengan información.

5.1.2. Conjuntos de prueba

Para la realización de las pruebas del algoritmo de búsqueda de caminos en grafos RDF se utilizaron dos conjuntos de 100 pares de artículos, los que se describen a continuación:

Conjunto 1 En este conjunto los pares de artículos son iguales para todos los grafos. Ya que los grafos a utilizar aumentan en cantidad de nodos, se crea este conjunto de pares de artículos de modo que estos artículos estén presentes en todos los grafos. Los artículos tienen un ID que puede ir desde 1 a 100.000 y fueron escogidos de forma aleatoria. Por ejemplo, si el par 298 y 8556 está presente en el conjunto para el grafo con IDs menores a 1.600.000 también estará presente en el conjunto para el grafo con IDs menores a 6.400.000.

Conjunto 2 En este conjunto los pares de artículos son distintos para cada grafo. Los artículos tienen un número de ID aleatorio entre 1 y el número de ID límite del grafo. Por ejemplo, para el grafo con ID límite 6.400.000 tiene pares de artículos entre 1 y 6.400.000.

5.1.3. Experimento

Para la realización de las pruebas se efectuó la búsqueda de los caminos entre cada par de artículos en cada conjunto de prueba. Cada búsqueda se ejecutó en cada uno de los grafos creados y se repitieron para cada variación de ponderación de grafos. Las características medidas de cada búsqueda fueron las siguientes:

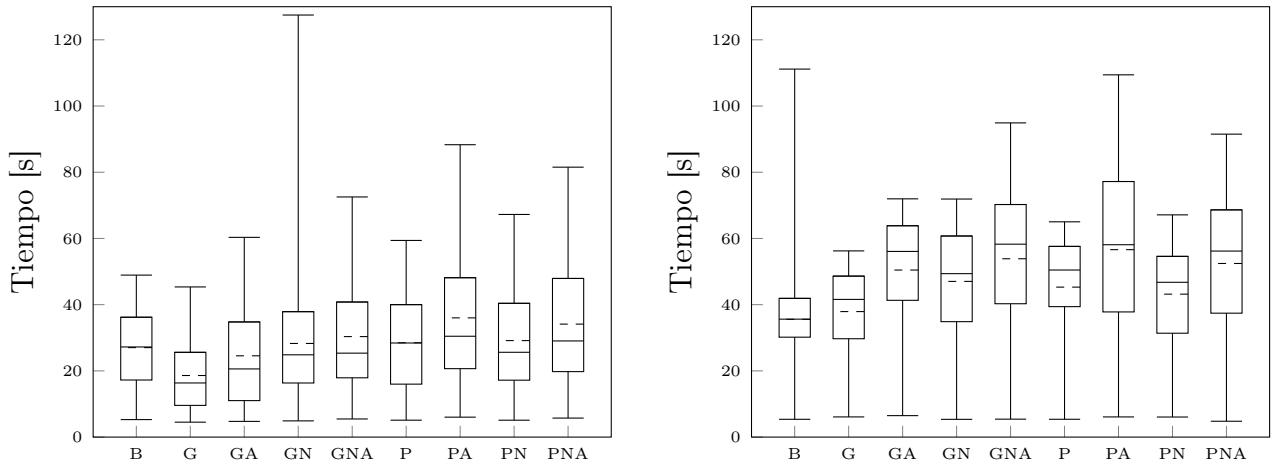
- Largo del camino resultante.
- El tiempo que tomó la búsqueda del camino.
- El total de nodos visitados.
- El grado máximo de los nodos visitados.
- El peso de los nodos visitados.

5.1.4. Rendimiento

El primer resultado a analizar se grafica en la figura 5.2 y corresponde al rendimiento del algoritmo de búsqueda de caminos. Para esto se midió el tiempo que tomó la búsqueda de caminos entre dos conceptos.

Diferencia de rendimiento entre conjuntos de nodos a buscar

Comenzaremos analizando la diferencia entre los tiempos de búsqueda obtenidos en los distintos conjuntos de prueba utilizados. Como se aprecia en las figuras 5.2a y 5.2b existe un aumento en el tiempo que toma buscar conceptos que abarcan todo el conjunto de datos en



(a) Conjunto de pares de nodos con ID menores a 100.000.

(b) Conjunto de pares de nodos aleatorios dentro del grafo.

Figura 5.2: Gráficos con tiempo [s] de búsqueda en grafo con todos los nodos por cada variación de ponderación.

relación a los que tienen ID menor a 100.000. Esto se debe a que los artículos con un número de identificador menor son más importantes que los artículos con un identificador mayor. Por ejemplo, los primeros cinco artículos son Universo, Tierra, vida, muerte y ser humano. Es así como la búsqueda de caminos entre conceptos más importantes implica que los nodos se encuentran más cercanos y conectados, por lo tanto como se observa en la figura 5.3 la cantidad de nodos a visitar es menor.

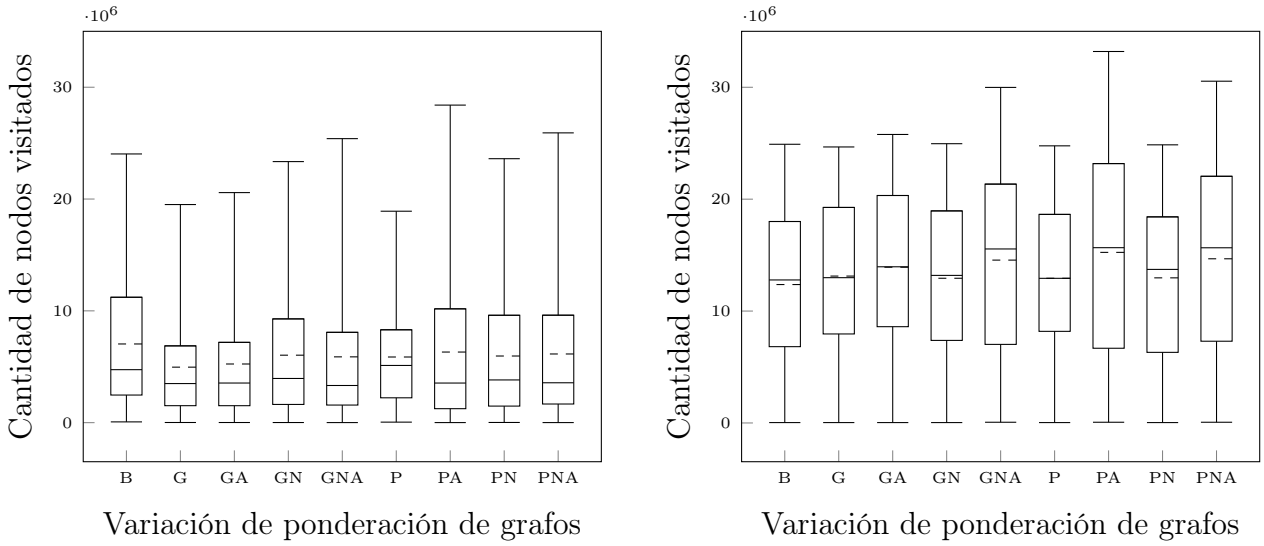
Rendimiento en distintas variaciones de ponderación de grafos

Como se explicó en la sección 2.3.1, el rendimiento del algoritmo de búsqueda de caminos utilizado está directamente relacionado con la cantidad de nodos y aristas visitadas. Si bien no contamos con el número de aristas de los nodos visitados, en las figuras 5.3a y 5.3b con 5.2a y 5.2b es posible apreciar la relación directa que existe entre los nodos visitados entre las distintas ponderaciones del grafo y los tiempos de búsqueda obtenidos en las pruebas.

Rendimiento en distintas escalas de grafos

Para visualizar el rendimiento de la búsqueda de caminos entre los distintos grafos utilizaremos la ponderación de grafos con mayor tiempo en promedio de búsqueda, el que corresponde a PageRank con peso en las aristas.

Como se observó previamente en la figura 5.1, los grafos utilizados para la realización de las pruebas presentan un crecimiento significativo tanto de nodos como aristas, esto debido a que los IDs límites fueron escogidos de manera que crecieran exponencialmente. Como se



(a) Conjunto de pares de nodos con ID menores a 100.000.

(b) Conjunto de pares de nodos aleatorios dentro del grafo.

Figura 5.3: Gráficos con nodos visitados en grafo con todos los nodos por cada variación de ponderación.

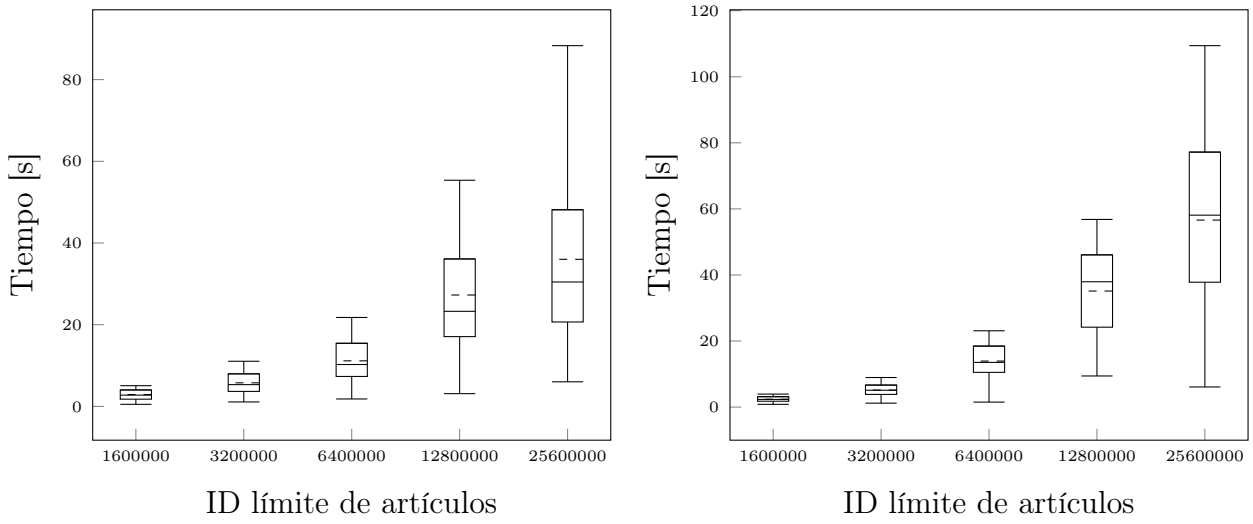
observa en las figuras 5.4a y 5.4b, el algoritmo presenta un comportamiento similar al crecimiento descrito por los grafos en que se utilizó.

En general, podemos ver que el rendimiento del proceso para grafos más grandes es probablemente demasiado lento para disponer de una aplicación interactiva en línea. Por ejemplo, para el grafo completo, el usuario tendrá que esperar, en promedio, un minuto con la estrategia más costosa, en algunos casos dos minutos. Este rendimiento no cambia mucho entre las distintas estrategias. Esta observación es esperable dado el tamaño del grafo y el costo general de computar caminos, lo que puede necesitar visitar muchos nodos como vimos en la figura 5.3.

5.1.5. Largo de los caminos

En la figura 5.5 observamos una de las principales características de los caminos y más simple de medir, su largo. Cuando se comparan las variaciones de ponderación del grafo se puede apreciar que el largo de los caminos es similar tanto para el conjunto de pares de nodos menores a 100.000 (figura 5.5a) como para el conjunto con los pares aleatorios en todo el grafo (figura 5.5b). A continuación se explican los resultados en detalle:

Caso base (B) Los caminos correspondientes a esta variación corresponden a los caminos de largo mínimo, ya que todos los nodos tienen el mismo peso. Esto resulta en los caminos de menor largo entre todas las ponderaciones.



(a) Conjunto de pares de nodos con ID menores a 100.000.

(b) Conjunto de pares de nodos aleatorios dentro del grafo.

Figura 5.4: Gráfico con tiempos de búsqueda en distintas escalas de grafos (los grafos crecen exponencialmente) con ponderación PageRank y aristas ponderadas.

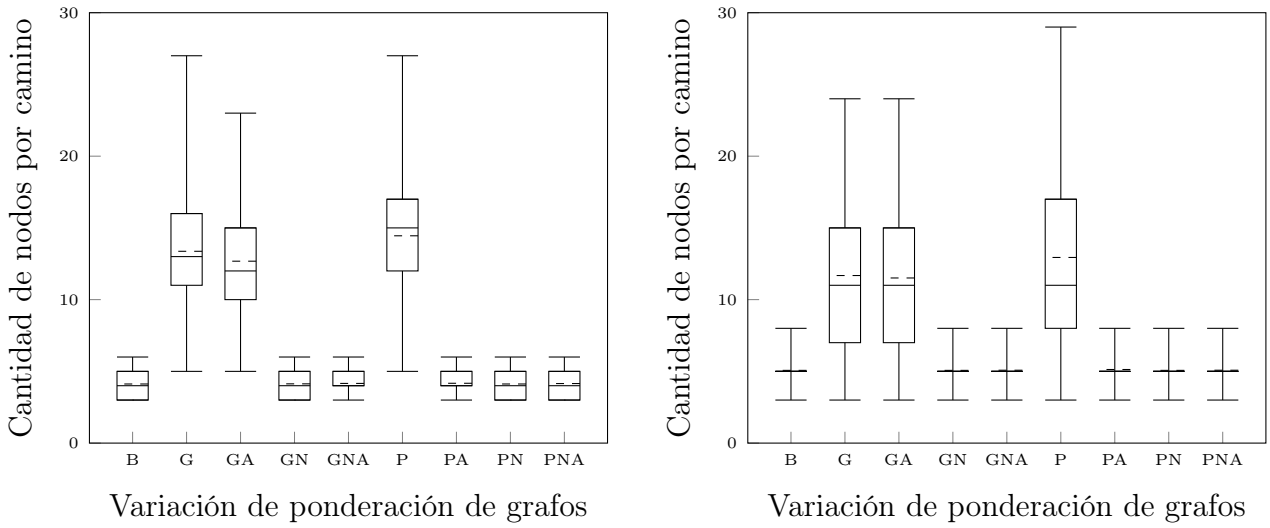
Grado (G), Grado y peso de arista (GA), PageRank (P) Los caminos resultantes de estas variaciones tienen caminos más largos por los siguientes motivos:

- **Grado (G) y PageRank (P)**: Debido a la diferencia entre los pesos de los nodos, los caminos resultantes se desvían de nodos con alto peso produciendo caminos más largos.
- **Grado y peso de arista (GA)**: Los pesos de las aristas están normalizados (entre 1 y 2) mientras que los pesos de los nodos no (grado de los nodos por lo general es alto), por esto los pesos de las aristas no influyen en gran medida en el largo de los caminos y se obtienen caminos de largo similar a los caminos en grafos con ponderación Grado.

PageRank y peso de arista (PA) Los caminos resultantes utilizando esta variación se ven influenciados en el largo del camino debido a que los valores de PageRank son bajos comparados con el peso de las aristas (la suma de todos los pesos en PageRank es aproximadamente 1 mientras que el peso de cada arista está entre 1 y 2). Es decir los resultados se ven afectados en mayor medida por el número de aristas del camino, lo que es equivalente al largo del camino.

Variaciones normalizadas (*N*) Estas ponderaciones están normalizadas utilizando la fórmula explicada en la sección 4.2.3 ($w_n(n) = 1 + \frac{w(n)}{max(w_{max})}$). Debido a esto, el largo del camino aún influye en el los caminos resultantes, por lo que se obtienen caminos de menor largo que sus variaciones sin normalizar.

Con respecto a la comparación del largo de los caminos entre los diferentes grafos, por la mayor parte, estas no varían en gran medida. En la figura 5.6 se puede apreciar el largo de los caminos obtenidos en los distintos grafos para la variación Caso base y Grado. Con la



(a) Conjunto de pares de nodos con ID menores a 100.000.

(b) Conjunto de pares de nodos aleatorios dentro del grafo.

Figura 5.5: Gráficos con largos de los caminos resultantes en el grafo con todos los nodos por cada variación de ponderación de grafos.

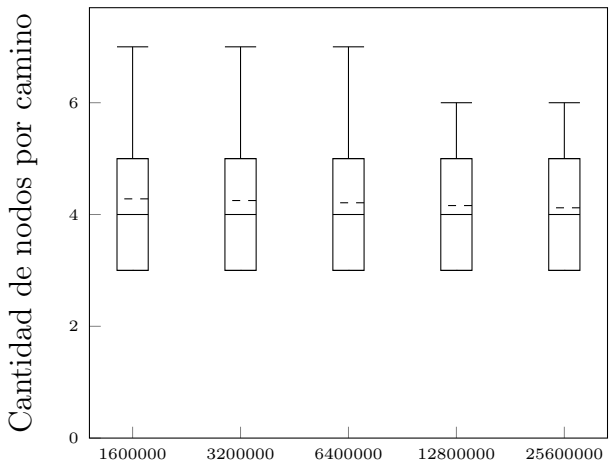
variación Caso base (figura 5.6a) se aprecia que los caminos no crecen en extensión ya que se minimiza el largo de camino para la búsqueda. En la variación Grado (figura 5.6b) se aprecia un leve cambio debido al cambio de grado de los nodos al aumentar el ID límite de artículos.

5.1.6. Similitud de los caminos

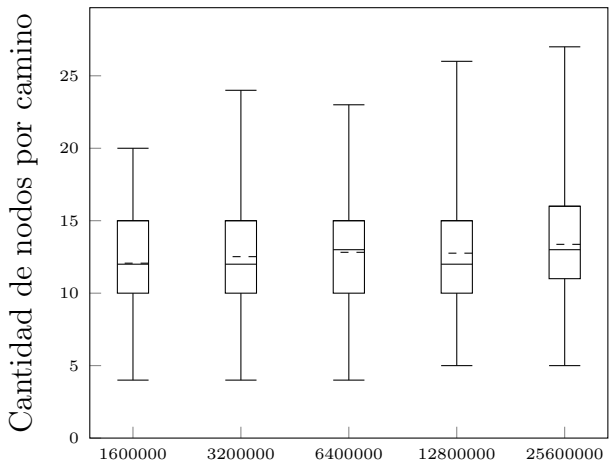
En esta sección se hace una comparación de los caminos resultantes para cada variación de ponderación de grafos. Para esto se tomó cada camino resultante entre cada par de artículos de una variación y se comparó con el resto de las variaciones. Por ejemplo, en la figura 5.7a se tiene que la casilla Grado (G) con Grado y peso de arista (GA) tiene un valor de 85 %, esto significa que de los 100 pares de artículos 85 entregaron como respuesta el mismo camino.

A partir de las figuras 5.7 se analizarán los distintos resultados de la comparación de caminos:

Grado (G), Grado y peso de arista (GA) Si se observan las filas o columnas correspondientes a estas variaciones se aprecia una baja cantidad de resultados iguales, excepto cuando se comparan entre ellos. La explicación a esto es que, al igual que en la comparación del largo de los caminos, estas variaciones asignan valores altos a los nodos por lo que los caminos resultantes los evitan. Estos nodos son los que conectan gran parte del grafo (por esto su alto peso), es decir por lo general forman parte de los caminos cuando su búsqueda se basa en el largo de este. Así es como estas ponderaciones de grafos resultan por lo general en caminos diferentes al resto. Con respecto a la similitud entre ambos, esto se debe a que los valores de los pesos de las aristas están en el rango]1,2] por lo que solo producen diferencias



(a) Variación de ponderación de grafos usada: Caso base.



(b) Variación de ponderación de grafos usada: Grado.

Figura 5.6: Gráficos con largos de los caminos en los distintos grafos para el conjunto de artículos con ID menor a 100.000.

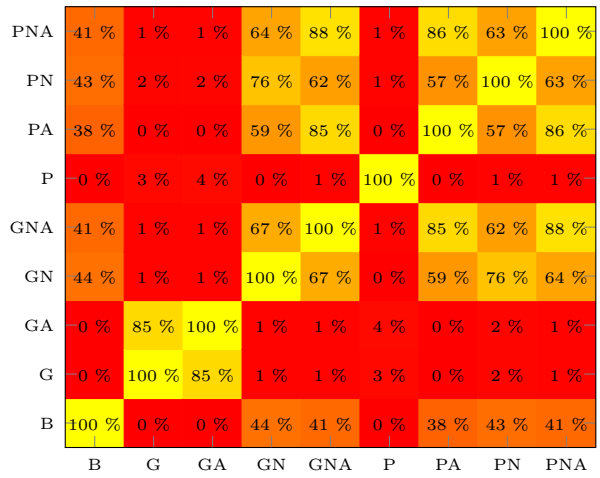
en casos borde (dos aristas diferentes unen el mismo nodo o a nodos con el mismo peso).

PageRank (P) A pesar de que los valores de PageRank son bajos (menores a 1), en esta variación son el único valor involucrado en la creación de los caminos por lo tanto este método de ponderación al ser diferente al resto resulta en caminos diferentes.

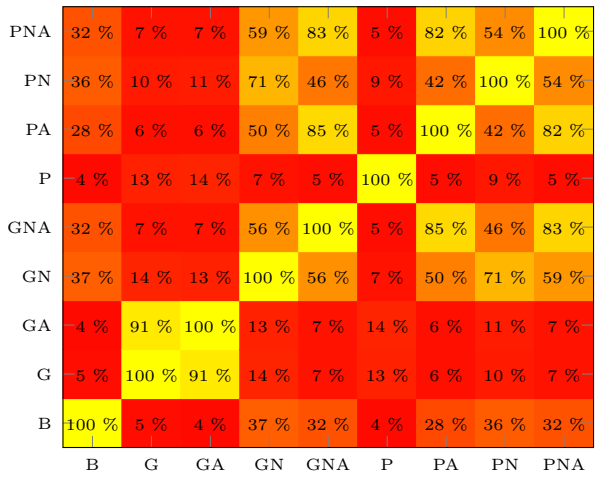
PageRank y peso de arista (PA) Como se explicó en la sección sobre comparación del largo de los caminos, los valores de PageRank son bajos en comparación al peso de las aristas, por lo tanto, estos solo influyen en casos borde.

Caso base (B) y variaciones normalizadas (*N*) Para el caso de las variaciones Caso base y las ponderaciones normalizadas, el porcentaje de caminos iguales es entre 30 % y 40 % debido a que en todos estos casos se incluye el largo del camino en la forma de encontrar el camino resultante (debido al método para normalizar los valores).

Como conclusión general, aquí podemos ver que, aparte de algunas excepciones (D/DE, por ejemplo), la elección de una estrategia de ponderación cambia significativamente los resultados. Así la siguiente pregunta es ¿cuál estrategia da caminos que los usuarios estiman más relevantes?



(a) Conjunto de pares de nodos con ID menores a 100.000.



(b) Conjunto de pares de nodos aleatorios dentro del grafo.

Figura 5.7: Heat maps con porcentaje de caminos iguales entre variaciones de ponderación de grafos considerando todos los datos de Wikidata.

5.2. Evaluación de usuarios

Con el objetivo de evaluar los resultados del algoritmo de búsqueda de caminos en un grafo RDF con distintas ponderaciones de nodos y aristas se organizó una prueba con usuarios. A continuación se describe la evaluación llevada a cabo.

5.2.1. Interfaz utilizada

Se utilizó la aplicación web descrita en la sección 4.3.4 para el ingreso de consultas, visualización de los caminos resultantes y sistema de votación. Considerando los resultados de tiempo de búsqueda de los distintos grafos (ver figura 5.4) se optó por un grafo con ID de artículo límite de 1.600.000. Los usuarios participantes de la prueba corresponden a diez alumnos del curso La Web de Datos (CC6202).

La prueba con usuarios busca evaluar una serie de caminos entre dos artículos con un puntaje entre 1 y 7. Los datos utilizados corresponden a un subconjunto de la base de datos de Wikidata. Para realizar la prueba los participantes se dirigen a la aplicación web, ingresan los pares de artículos que se le asignan e ingresan un puntaje por cada camino resultante.

5.2.2. Conjuntos de pares de artículos

Los pares de artículos asignados a cada usuario se crearon de la siguiente manera. Primero, se crearon dos grupos de pares de artículos como se observa en la figura 5.8.

Artículos relacionados

La lista de la figura 5.8a está constituida por pares en que cada artículo guarda una relación con el artículo con el que conforma el par. Por ejemplo, George W. Bush y Barack Obama ambos fueron presidentes de los Estados Unidos o Lionel Messi y Stephen Curry ambos son deportistas de alto rendimiento.

Artículos no relacionados

La lista de la figura 5.8b corresponde a pares en que cada artículo no tiene una relación directa con el artículo con el cual forma el par. Por ejemplo, la montaña Everest y la FIFA o la Odisea y Facebook.

Pares de artículos relacionados

- George W. Bush a Barack Obama
- Bolivia a Portugal
- Elvis Presley a Deep Purple
- Lionel Messi a Stephen Curry
- Torre Eiffel a Torre de Pisa
- Alan Turing a Leonhard Euler
- Samsung a Apple inc.
- Ford a Volswagen
- Al Pacino a Bill Murray
- CNN a BBC

(a) Pares relacionados.

Pares de artículos no relacionados

- Donald Trump a Pablo Neruda
- Steam a Barcelona FC
- Everest a FIFA
- El señor de los anillos a Apollo 11
- RMS Titanic a Saturno
- Netflix a Napoleón Bonaparte
- Segunda guerra mundial a Platón
- Los beatles a Star Wars
- La odisea a Facebook
- Rayos X a Amazon.com

(b) Pares no relacionados.

Figura 5.8: Grupos con pares de artículos para evaluación de usuarios.

Finalmente, los pares entregados a los usuarios fueron 10. De estos pares, 5 fueron elegidos aleatoriamente de la lista con pares relacionados y 5 de la lista con pares no relacionados.

5.2.3. Resultados del estudio de usuarios

En esta sección se analizan los resultados obtenidos de la evaluación con usuarios. A modo de ejemplo en la figura 5.9 se observan los caminos que obtuvieron mejor puntaje.

Figura 5.9: Cinco de los caminos con mejor puntaje.

Evaluaciones resultantes

En total se realizaron 81 evaluaciones, donde por cada evaluación se le asignó un puntaje a los 9 caminos resultantes de cada variación de ponderación del grafo. En la figura 5.10 se observa las evaluaciones obtenidas por cada par de artículos. Para analizar la dispersión de los puntajes obtenidos se tomaron los 4 caminos con mayor cantidad de evaluaciones y se calculó su desviación estándar. Estos caminos corresponden a dos conceptos relacionados (Al Pacino a Bill Murray y Bolivia a Portugal) y dos conceptos no relacionados (Donald Trump a Pablo Neruda y El señor de los anillos a Apollo 11). En la figura 5.11 se observa que la desviación estándar es menor a 2,6 y que los conceptos relacionados tienen una menor dispersión (alrededor de 1,5) que los conceptos no relacionados. Esta diferencia en la desviación estándar probablemente se deba al hecho de que los usuarios están más de acuerdo con los caminos relevantes para pares relacionados que con pares no relacionados, dado que en el caso previo, a menudo, hay un camino más natural entre los nodos.

Evaluación por caminos

En la figura 5.12 se observa el promedio de cada evaluación de las distintas ponderaciones de grafo por cada par de artículos buscado. Aquí se aprecia que los pares de artículos que se encuentran relacionados en algún ámbito tienden a tener una mejor evaluación por parte de los usuarios.

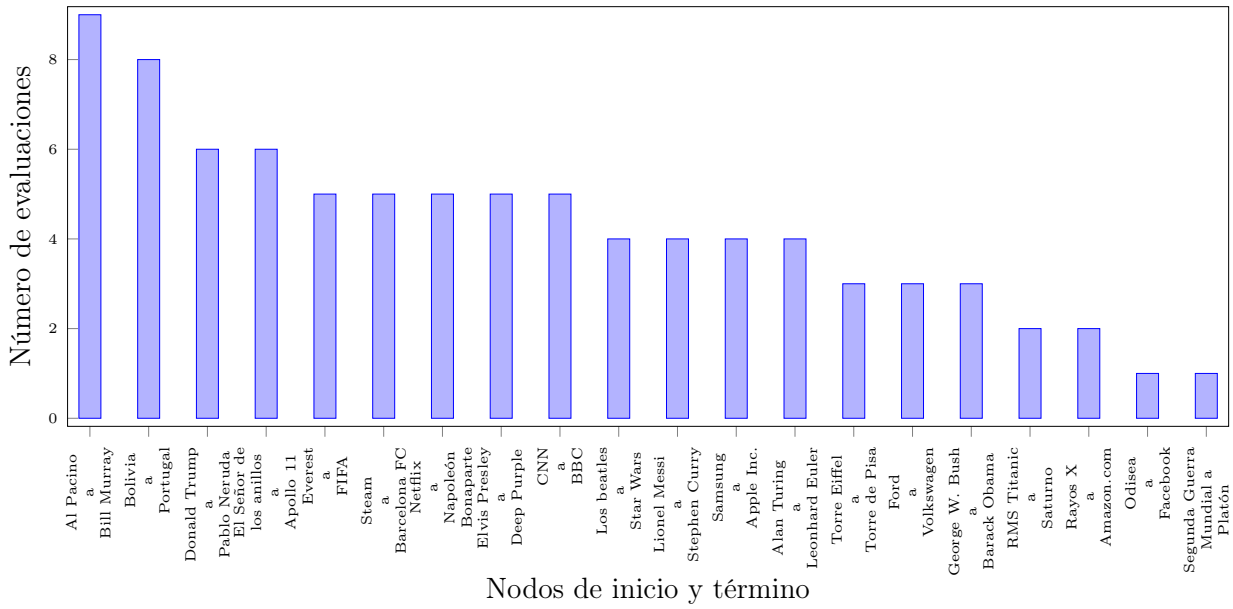


Figura 5.10: Número de evaluaciones por camino.

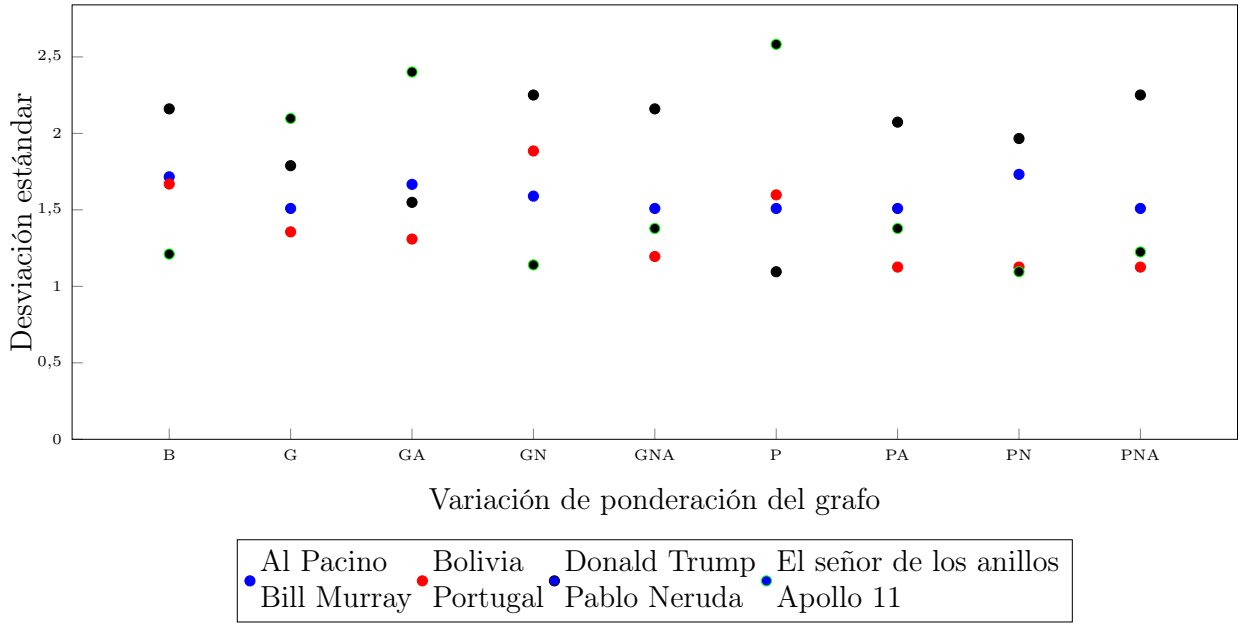


Figura 5.11: Desviación estándar de caminos resultantes según ponderación del grafo.

Evaluación por variaciones de ponderación de grafos

Al analizar los datos de las evaluaciones con respecto a la variación de ponderación del grafo (figura 5.13) se observa que en todos los casos el promedio de las evaluaciones para conceptos que se encontraban relacionados obtuvieron un mejor puntaje que en los conceptos no relacionados.

Por otro lado, si observamos la figura 5.14 se tiene que la ponderación Caso base fue la peor evaluada, independiente del grupo de pares de artículos. Además, se observa que en ambos

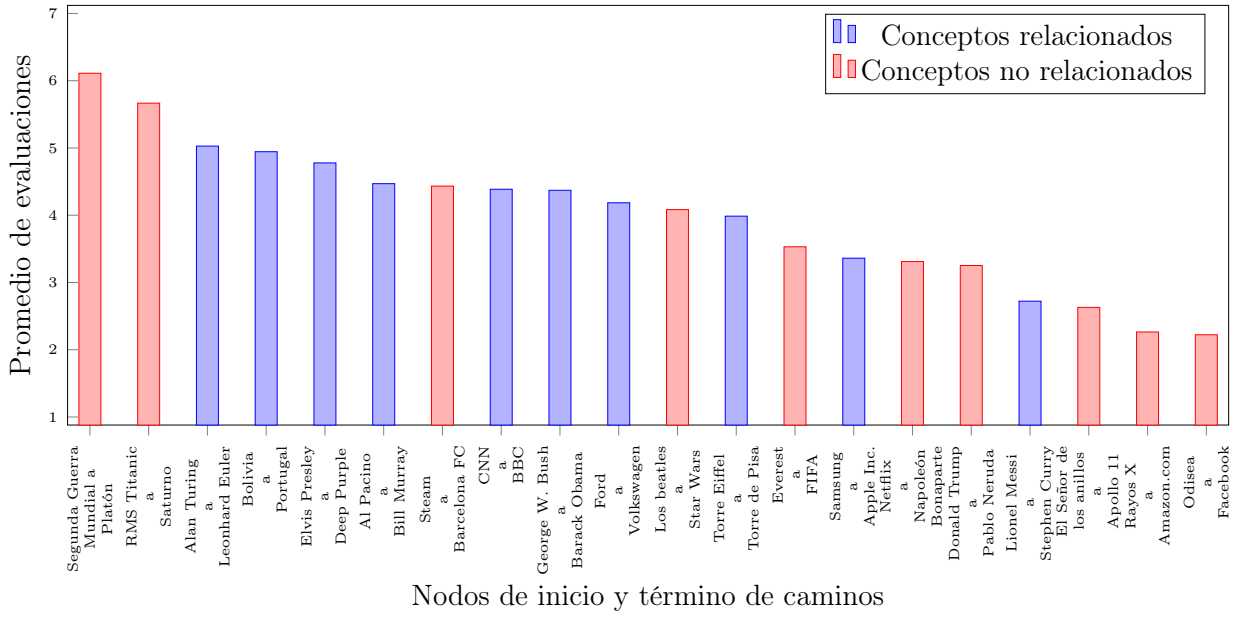


Figura 5.12: Promedio de evaluaciones de caminos entre todas las ponderaciones del grafo.

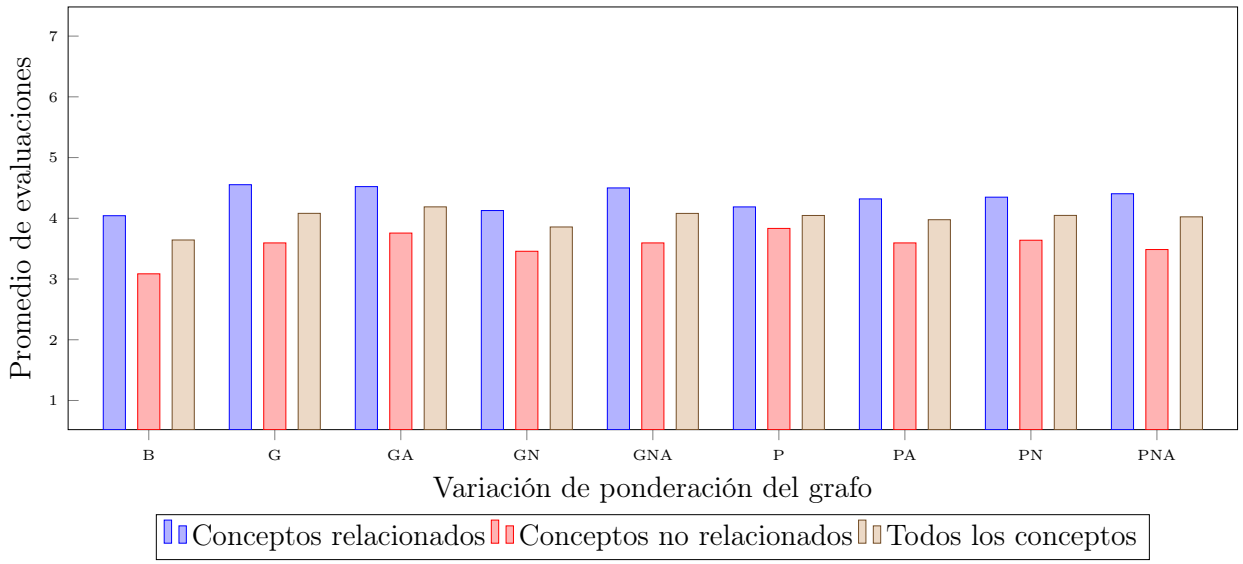
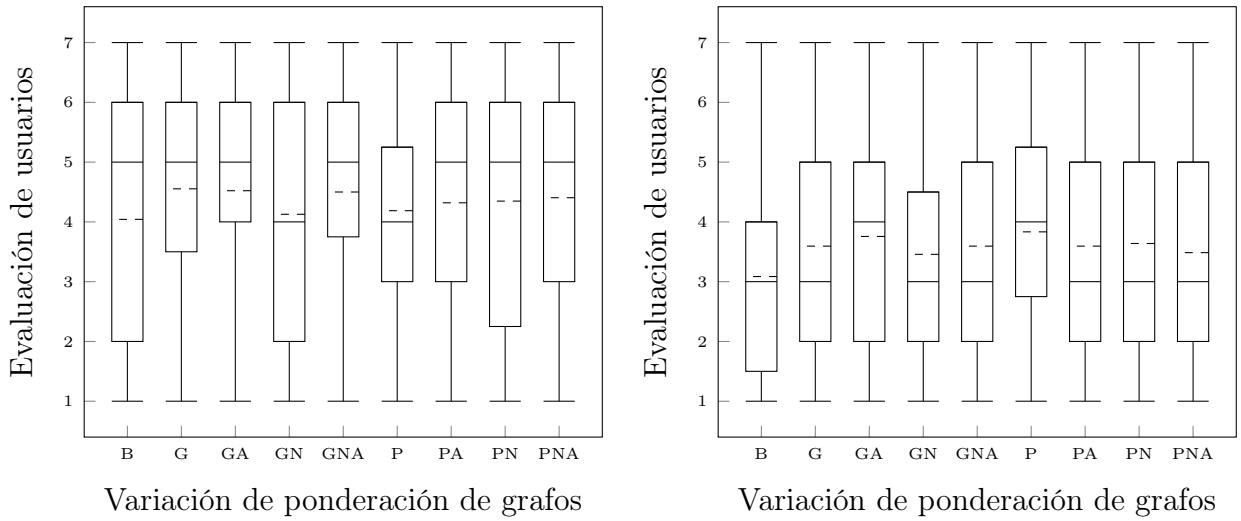


Figura 5.13: Promedio de evaluaciones entre todos los caminos por ponderación del grafo.

grupos las evaluaciones en todas las ponderaciones se distribuyeron entre 1 y 7. De la figura 5.14a se interpreta que la ponderación por grado de los nodos tiene el mejor promedio de evaluaciones y concentra 50 % de las evaluaciones entre 4 y 6. De la figura 5.14b se interpreta que la distribución de las evaluaciones en general es similar independiente de la ponderación del grafo.



(a) Grupo de pares de conceptos relacionados. (b) Grupo de pares de conceptos no relacionados.

Figura 5.14: Evaluación de usuarios por ponderación de grafo.

Evaluación por largo de los caminos

Otra característica que presentan los caminos, además de la relación de los nodos de inicio y término de estos, es su largo. En la figura 5.15 se pueden observar los promedios de las evaluaciones con respecto al largo de los caminos. Aunque valida la idea de que un camino más corto no implica ser un camino más relevante para un usuario (como es la hipótesis central de esta memoria), este resultado es sorprendente por el hecho de que se esperaba una preferencia más fuerte de caminos más cortos. Si bien no hay una correspondencia clara entre el largo de los caminos y el puntaje obtenido en la evaluación, se tiene que los caminos de largo menor o igual a 6 reciben una mejor evaluación que los caminos con largo sobre 6.

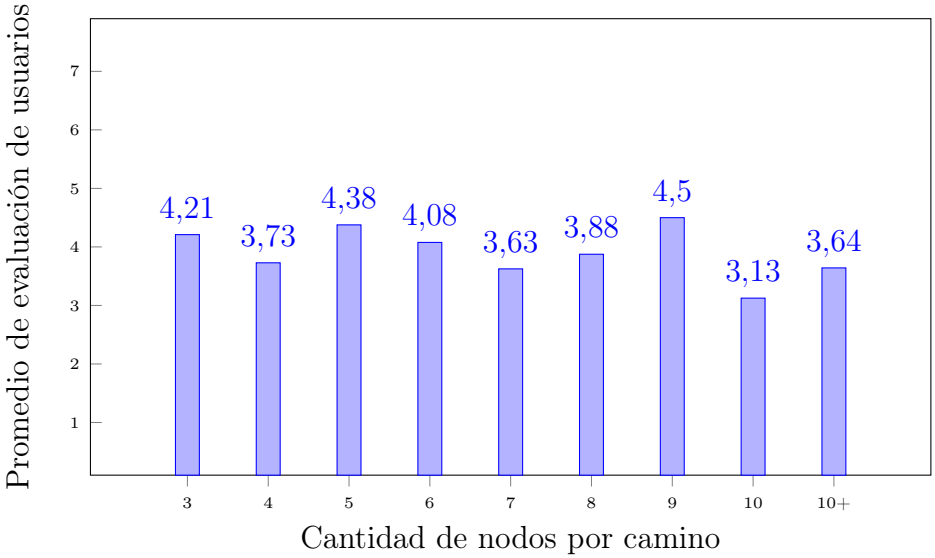


Figura 5.15: Evaluación de usuarios por largo de los caminos resultantes.

5.3. Análisis de resultados

A partir de los resultados obtenidos de los experimentos y evaluación realizadas, las principales respuestas a las interrogantes propuestas al comienzo de la sección son las siguientes:

¿Cuáles estrategias de ponderación de grafos producen caminos más relevantes en la estimación de los usuarios?

A pesar de que los resultados de la evaluación con usuarios según la variación de ponderación del grafo utilizada, figura 5.13, no son completamente determinantes al señalar la mejor estrategia para la obtención de caminos relevantes es importante notar que, tanto los métodos que se basan en el grado de los nodos, como los métodos basados en la utilización de PageRank obtienen una mejor evaluación que el caso base (caso que busca minimizar el largo de los caminos).

¿Cómo varían los caminos entre las estrategias? ¿Cuáles características (por ejemplo, largo del camino) tienen? ¿Cómo varía el largo de los caminos con distintas muestras de datos?

Los caminos resultantes entre cada variación de ponderación del grafo varían considerablemente como se observa en la figura 5.7. La única excepción son los caminos obtenidos al utilizar la variación grado y grado con peso en aristas debido a la diferencia entre los valores asignados a las aristas y nodos como se explicó en la sección 5.1.6.
La principal característica medida en los experimentos realizados corresponde al largo de los caminos resultantes. En la figura 5.5 se observa que los caminos obtenidos utilizando las variaciones grado, grado con peso en aristas y PageRank presentan un largo considerablemente mayor al resto de las variaciones como se explicó en la sección 5.1.5.
Al utilizar distintas escalas para el tamaño del grafo se obtuvieron caminos con largos similares como se observa en la figura 5.6. En su mayoría, los cambios en los largos de los caminos se producen debido al cambio de los grados y PageRank de los nodos al aumentar el tamaño del grafo.

¿Cómo es el rendimiento del algoritmo de Dijkstra? ¿Cómo varía el rendimiento con respecto al tamaño de los datos? ¿Cómo varía el rendimiento entre las distintas estrategias de ponderación de grafos?

El rendimiento del algoritmo de Dijkstra utilizado para la búsqueda de caminos fue el esperado, es decir, los tiempos de búsqueda crecieron de manera similar a la cantidad de nodos en las distintas escalas de grafos utilizados como se observa en la figura 5.4.
Para las distintas variaciones de ponderación del grafo los tiempos de búsqueda (figura 5.2) y la cantidad de nodos visitados en cada búsqueda (figura 5.3) explican la diferencia de rendimiento entre las distintas estrategias como se explicó en la sección 5.1.4.

Capítulo 6

Conclusión

A continuación se presentan las conclusiones obtenidas del trabajo de memoria realizado.

6.1. Objetivos logrados

En este trabajo de memoria se implementó un algoritmo de búsqueda de caminos relevantes entre dos nodos de un grafo RDF. Para esto se construyó un sistema para representar información guardada en formato RDF en grafos. Además, con el fin de encontrar caminos relevantes y no basar la búsqueda solo en que la propiedad a minimizar sea el largo del camino, se crearon métodos de ponderación de grafos según el grado de los nodos, etiqueta de las aristas y la utilización de PageRank. Así se utilizó el algoritmo de Dijkstra de búsqueda de caminos de tal manera que encontrara los caminos con mínimo peso de nodos y aristas. Finalmente, se creó una aplicación con el fin de visualizar los caminos resultantes.

6.2. Resultados obtenidos

Como resultado de las pruebas realizadas se encontró que el rendimiento de la búsqueda de caminos medido se ajusta a la complejidad del algoritmo. Por otro lado se obtuvieron diversos caminos a consecuencia de la utilización de las distintas variaciones de ponderación de grafos diseñadas. Además, se realizó una evaluación con usuarios que entrego resultados que, si bien no son definitorios, ayudan a la compresión de qué caminos son considerados relevantes para los usuarios. Los principales resultados obtenidos a partir de las pruebas y evaluación relazadas son las siguientes:

- Experimentos:
  (a) El rendimiento del algoritmo de Dijkstra fue el esperado. Los tiempos de búsqueda de caminos para pares de nodos de mayor importancia, es decir con menor ID, fueron menores debido a una menor cantidad de nodos visitados. Además, se obtu-

vieron tiempos de búsqueda similares para las distintas variaciones de ponderación de grafos.

(b) Los resultados al comparar el largo de los caminos obtenidos al utilizar las distintas variaciones de ponderación del grafo fueron similares a excepción de las variaciones grado, grado con peso en aristas y PageRank. Al comparar el largo de los caminos con las distintas escalas de datos se observaron mínimas variaciones.

(c) Los caminos resultantes fueron distintos al utilizar distintas variaciones de ponderación de grafos.

- Evaluación con usuarios:

(d) Las búsquedas de pares de conceptos relacionados obtuvieron una mejor evaluación que las búsquedas de pares no relacionados, independiente de la ponderación del grafo.

(e) La variación de ponderación correspondiente al caso base (caminos con largo mínimo) obtuvo la peor evaluación.

(f) Los caminos con menor largo obtuvieron una evaluación levemente mayor.

6.3. Conclusiones de resultados

De estos resultados se puede deducir que, si bien los caminos cortos pueden ser interesantes para algunos casos, los caminos más cortos no necesariamente son siempre los de mayor relevancia para el usuario. En general, parece difícil encontrar una medida de caminos que se correlacione exactamente con lo que los usuarios estiman relevante. La mejor selección de la ponderación del grafo para la búsqueda de caminos relevantes probablemente varíe según el tipo de nodos de inicio y fin del camino y, en general, del propósito de la aplicación. Con respecto al tema del rendimiento, si bien el resultado de las pruebas de los tiempos de ejecución del algoritmo son los esperados, este tiene tiempos de búsqueda razonable hasta grafos entre 3.200.000 y 6.400.000 con un promedio de 10s. Para grafos más grandes es necesario hacer modificaciones al algoritmo implementado o encontrar otra aproximación al problema para asegurar tiempos razonables para un sistema interactivo.

6.4. Trabajo futuro

Como trabajo futuro se plantea encontrar variaciones al algoritmo de búsqueda a través de la utilización de heurísticas que ayuden a mejorar su rendimiento. Otro aspecto a considerar para un trabajo futuro es modificar o encontrar otros métodos para la ponderación de caminos según el área o función que se desea cubrir. Finalmente, parece importante lograr un mejor entendimiento de lo que significa un "camino relevante" en este contexto, lo que probablemente es sensible a la aplicación o al usuario en particular.

Bibliografía

[1] Renzo Angles, Marcelo Arenas, Pablo Barceló, Aidan Hogan, Juan L. Reutter, and Domagoj Vrgoc. Foundations of modern graph query languages. *CoRR*, abs/1610.06264, 2016.

[2] P. E. Hart, N. J. Nilsson, and B. Raphael. A formal basis for the heuristic determination of minimum cost paths. *IEEE Transactions on Systems Science and Cybernetics*, 4(2):100–107, July 1968.

[3] Philipp Heim, Sebastian Hellmann, Jens Lehmann, Steffen Lohmann, and Timo Stegemann. Relfinder: Revealing relationships in rdf knowledge bases. In *Proceedings of the 4th International Conference on Semantic and Digital Media Technologies (SAMT 2009)*, pages 182–187, Berlin/Heidelberg, 2009. Springer.

[4] Philipp Heim, Jürgen Ziegler, and Steffen Lohmann. gFacet: A browser for the web of data. In *Proceedings of the International Workshop on Interacting with Multimedia Content in the Social Semantic Web (IMC-SSW 2008)*, volume 417 of *CEUR-WS*, pages 49–58, 2008.

[5] Donald E. Knuth. A generalization of Dijkstra's algorithm. *Information Processing Letters*, 6(1):1 – 5, 1977.

[6] Lawrence Page, Sergey Brin, Rajeev Motwani, and Terry Winograd. The pagerank citation ranking: Bringing order to the web. Technical Report 1999-66, Stanford InfoLab, November 1999. Previous number = SIDL-WP-1999-0120.

[7] Guus Schreiber and Yves Raimond. RDF 1.1 Primer. W3C Working Group Note, June 2014. `http://www.w3.org/TR/rdf11-primer/`.

[8] Denny Vrandecic and Markus Krötzsch. Wikidata: a free collaborative knowledgebase. *Commun. ACM*, 57(10):78–85, 2014.