



Algoritmo de recomendaciones: Programa que recomienda plantas como regalos

Estuardo Ureta 17010

Oliver Graf 17190

Kristen Brandt 171482

I. Índice

Índice

II. Investigación de algoritmos existentes:

III. Design Thinking:

Definición del problema:

Propuesta de ideas de solución del problema

Prototipos de baja fidelidad

Pseudocódigo

Base de datos inicial

Referencias

II. Investigación de algoritmos existentes:

Hoy en día los algoritmos de recomendación son utilizados en todas las áreas del mercado mundial, ya sea para publicidad o para la optimización de la experiencia que las personas tengan al utilizar un producto. Estos algoritmos son hechos a base de diferentes filtros que permiten que el algoritmo pueda sugerirle al usuario productos o temas de su interés subjetivamente, con el fin de poder optimizar la información presentada al usuario. Existen varios tipos de algoritmos de recomendación: basados en la opinión de muchas personas o tendencias, basado en la actividad de un único usuario y deep learning, por nombrar algunos ejemplos (Gorakala, 2016).

Los algoritmos de recomendación basados en los patrones de uso de todos sus usuarios y de las tendencias recientes aparecen principalmente en servicios como tiendas en línea, redes sociales o aplicaciones como Netflix. Muchas veces ni siquiera es necesario que el usuario ingrese su opinión explícitamente en el algoritmo, ya que en algunos ejemplos como Waze o Google Maps, los algoritmos trabajan en conjunto con herramientas de minería de datos, para calcular cuales rutas son las menos transitadas en un momento dado. Por otro lado, existen algoritmos, por ejemplo algoritmos que utilizan deep learning, que monitorean la actividad de un solo usuario y pueden no depender de tener una conexión al internet. Este tipo de algoritmos se pueden encontrar en algunos de los teléfonos inteligentes de ésta época, que pueden modificar su funcionamiento según como el usuario utiliza el dispositivo. Una desventaja de este tipo de algoritmos, es que si el usuario no interactúa lo suficiente con el algoritmo, éste no proporcionará buenas recomendaciones (Gorakala, 2016).

Estos algoritmos están en todas partes y hacen posibles muchas funciones que hoy en día todos los usuarios del internet utilizan diariamente. Por ejemplo, en la página web para subir y ver videos conocida como Youtube, el usuario tiene la capacidad de suscribirse a los canales que más les guste. En esta página, se pueden apreciar dos tipos de algoritmos de recomendación diferentes, los cuales ambos proporcionan videos recomendados al usuario en dos diferentes tabs. Uno de estos tabs es llamado el tab de tendencias y aparte está la página de inicio; en ambos aparecen videos de todo tipo para que el usuario tenga de donde escoger. La diferencia entre ambos tabs, es que en la página de inicio, aparecen videos relacionados de alguna manera al usuario, mientras que en la página de tendencias aparecen, valga la redundancia, las tendencias de todos los usuarios. En el tab de la página de inicio pueden aparecer videos de canales que el usuario ha visitado, sin necesidad de haberse suscrito a este, o incluso pueden salir videos relacionados a

otros videos que el usuario ha visto. Detrás de esta página de inicio hay un algoritmo de recomendaciones que se enfoca principalmente en los gustos personales del usuario y en posiblemente busca vínculos o patrones entre otros usuarios que miran videos parecidos para determinar cuáles videos recomendar. Por el otro lado, existe la página de tendencias, donde existe la posibilidad de que como usuario uno no encuentre ningún video que le llame la atención, pero a Youtube no le interesa que en esta página aparezcan videos personalizados. En la página de tendencias aparecen todos aquellos videos que están de moda y son elegidos y filtrados según la cantidad de visitas y likes que tengan. El algoritmo detrás de esta página probablemente sea menos complejo y por lo tanto menos efectivo, que el que se utiliza en la página de inicio, pero aún así sigue siendo un algoritmo de recomendaciones. La diferencia entre ambas páginas y sus respectivos algoritmos es el enfoque que tienen. Uno busca optimizar la experiencia personal del usuario y el otro busca vender videos (o mejor dicho vistas de videos, ya que Youtube hace dinero de la publicidad que muestra en sus videos) de manera fácil y rápida (Ridwan, 2019).

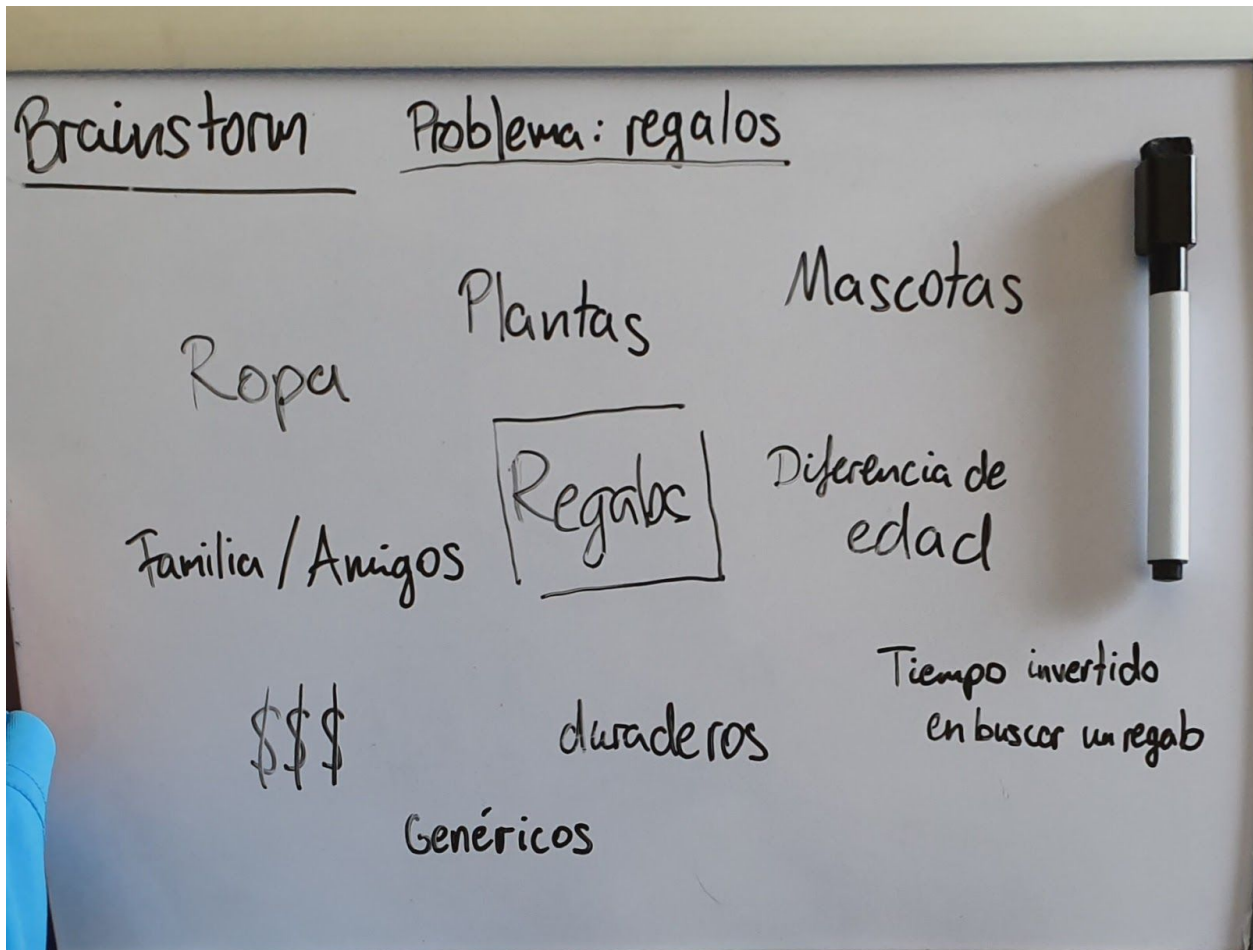
La manera en la que todos estos algoritmos funcionan es similar y dependen de herramientas matemáticas y de la capacidad de almacenar información. Dichas herramientas matemáticas y dicha información puede variar según el programa o la aplicación. En conclusión, los algoritmos de recomendaciones se pueden encontrar en todos lados y son muy versátiles en cuanto a su aplicación. En el caso nuestro, donde el objetivo es recomendar regalos al usuario, será necesaria la implementación de un algoritmo como el de la página de inicio de Youtube, que busca patrones para acercar al usuario los artículos que puedan interesarle y para alejar los que no. Será necesaria la implementación de alguna herramienta matemática, como por ejemplo, un índice de similitud para calcular las relaciones entre los usuarios y los artículos en venta (Ridwan, 2019).

II. Design Thinking:

Definición del problema:

El acto de elegir un regalo para alguien es muy complejo y existen muchas variables que tienen que tomarse en cuenta cuando lo que se busca es recomendar un regalo, por ejemplo, la importancia que le ponen el recipiente o la persona que está dando el regalo al regalo, la ocasión por la que se está regalando algo, el monto de dinero que la persona está dispuesta a gastar por el regalo, etc. Quisimos enfocarnos solo en un tipo de regalos para nuestro recommendation engine, por lo que se realizó una lluvia de ideas, seguido de una etapa de empatía, para determinar un tema adecuado.

Lluvia de ideas:



Entrevistas:

Se buscó más que una entrevista, generar un diálogo con los entrevistados sobre el problema mencionada anteriormente. Esto con el objetivo de generar insight, las cuales nos ayudan a ponernos en el lugar del usuario y ver más a profundidad de sus palabras. La población entrevistada fue de personas de 16-30 años con dinero y necesidad de dar regalos a sus seres queridos. A continuación se muestran a los entrevistados y la información recopilada con cada uno de ellos:

Entrevistado	Resumen comentarios	Insights
 Nombre: Natalia Estrada Edad: 20	Ella nos platicaba sobre como no le regalaría un regalo muy elaborado a una persona que no conoce muy bien, pero para su novio si le regalaría un regalo con mayor calidad.	La calidad del regalo a regalarse depende de la cantidad de tiempo que lleva la persona de conocer al recipiente.
 Nombre: Paul Packmohr Edad: 21	Paul nos cuenta que le regala muchas rosas a su novia pero que no regalaría una rosa a algún amigo porque representan amor para el.	Al momento de regalar una flor o planta el usuario elige dependiendo de la ocasión de la que se trata. Ej. Romance, Rosas, Ramos de flores Ej2. Funeral, Lirios o arreglos florales Ej3. Amistad, Cactus, suculentas



Nombre: Erick Hernandez
Edad:21

Nos comenta que le regaló un gran arreglo de flores a su mama que le salió tanto caro pero no le molesto gastar tanto ya que siente que su mamá lo merece.

El usuario toma en cuenta el tiempo que lleva conociendo a una persona o tal vez su cariño hacia ella y en base a eso elige el precio a la hora de escoger un regalo.



Nombre: Antonio
Edad:21

Esta persona nos habla que nunca le regalaría una planta a su hermana debido a que no es muy responsable y cree que ella no podría mantenerla viva

El usuario puede llegar a dudar de comprar un regalo el cual necesita cuidado ya que se siente como un compromiso, pero a ciertos individuos les gusta ese compromiso.



Nombre: Javier Palomo
Edad: 18

EL joven recién ingresado nos comenta que se siente incómodo con la idea de recibir flores como regalo ya que solamente las mujeres deberían recibir dichos regalos según Javier.


Normalmente el individuo de sexo femenino prefiere recibir flores a recibir plantas a diferencia del sexo masculino. Algunas personas están altamente arraigadas a ideas erróneas de lo que debe o no debe hacer un hombre para encajar en la sociedad.



Nombre: Ricardo Palco
Edad: 21

Ricardo nos dice que prefiere que le regalen un cactus ya que duran más y es más fácil cuidarlo ya que el se denomina una persona despistada.

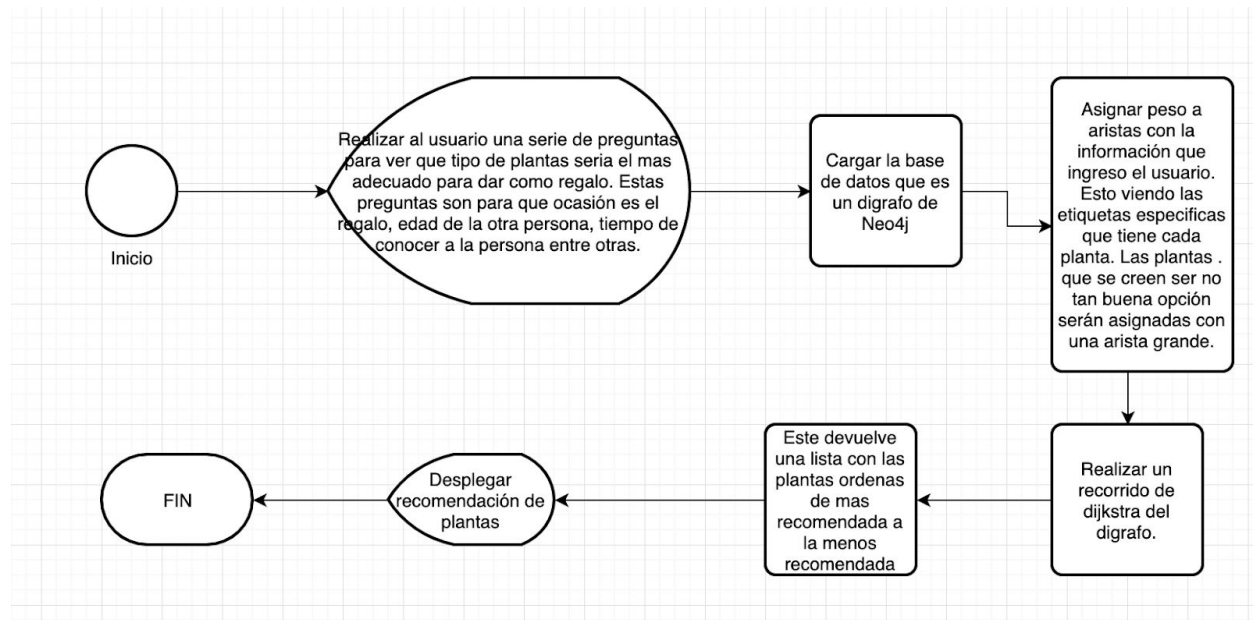
La mayoría de hombres que reciben plantas de regalo, terminan sin proporcionar los cuidados necesarios para la misma. Prefieren plantas de poco cuidado, por ejemplo, cactus.

 <p>Nombre: Marcos Sarti Edad: 22</p>	<p>Marcos no comenta de lo difícil que es para él encontrar regalos debido al poco tiempo que posee para pensar en qué regalar. El trabaja y va a la universidad todos los días y desearía que alguien más hiciera las tareas triviales por el.</p>	<p>Encontrar el tiempo para comprar un regalo puede llegar a ser un desafío.</p> <p>El usuario en ocasiones se presiona a sí mismo respecto al hecho de que su regalo sea perfecto.</p>
<p>Nombre: Edad:</p>	<p>Con ella platicamos de regalos y nos cuenta que cree que las nuevas generaciones no aprecian los arreglos florales tanto como personas de mayor edad.</p>	<p>La edad de la persona que recibe el regalo es importante para el usuario a la hora de elegir un regalo.</p>

Necesidades:

1. El compromiso que se le presenta al recipiente del regalo-planta puede ser muy grande, dependiendo del precio y cuidado necesario de la planta.
2. Determinar que planta es considerada un regalo demasiado grande puede resultar difícil.
3. La planta a elegir depende mucho de la ocasión y también depende de la cantidad de tiempo que lleva la persona de conocer al recipiente; determinar la planta ideal es problemático.

Pseudocódigo /Diagrama de flujo:



Base de datos inicial:

El diseño de la base de datos surgió de la idea de usar el algoritmo de Dijkstra para determinar qué artículos son más recomendables para el usuario. El grafo guardado en la base de datos contiene varios clusters de nodos, donde los nodos representan una sola planta. Los clusters estarán ordenados por las características que tengan en común las plantas, por ejemplo, la ocasión por la que se entrega ese tipo de plantas, el cuidado que tenga que recibir la planta, etc. El peso de las aristas se modificará ya dentro del programa que se escribirá en python. Al modificar el peso de las aristas esencialmente lo que se está haciendo con los nodos es alejarlos o acercarlos, según su recomendabilidad.

Explicación/Evidencia de que el prototipo funciona:

Para que el prototipo funcione dos cosas deben pasar: que en las plantas más recomendables el peso de sus aristas disminuyan y que las plantas menos recomendables incrementen el peso de sus aristas. Luego, lo demás funciona con el algoritmo de Dijkstra como se muestra a continuación en el siguiente código.

Digamos que el tulipán no es recomendable para el usuario, lo siguiente pasará:

```
3 import networkx as nx
4 G = nx.DiGraph()
5
6 # agregar nodos
7 G.add_node("Inicio")
8 G.add_node("Rosa")
9 G.add_node("Tulipan")
10 G.add_node("Lirio")
11 G.add_node("Girasol")
12 G.add_node("Cactus")
13 G.add_node("Suculenta")
14 G.add_node("Agapanto")
15
16 print ("Nodos: ", G.nodes())
17
18 # agregar aristas
19 G.add_edge("Inicio", "Rosa", weight=1)
20 G.add_edge("Inicio", "Tulipan", weight=10)
21 G.add_edge("Inicio", "Lirio", weight=2)
22 G.add_edge("Rosa", "Tulipan", weight=10)
23 G.add_edge("Rosa", "Girasol", weight=5)
24 G.add_edge("Lirio", "Girasol", weight=5)
25 G.add_edge("Girasol", "Tulipan", weight=5)
26 G.add_edge("Girasol", "Cactus", weight=5)
27 G.add_edge("Cactus", "Girasol", weight=5)
28 G.add_edge("Cactus", "Agapanto", weight=5)
29 G.add_edge("Suculenta", "Agapanto", weight=5)
30
31 print ("Aristas: ", G.edges())
32
33 # single source shortest path with Dijkstra
34
35 print ()
36 print ("Ruta mas corta con Dijkstra:")
37 print (nx.single_source_dijkstra_path(G, "Inicio"))
38
39 print
40 print ("Longitud de la ruta mas corta:")
41 print (nx.single_source_dijkstra_path_length(G, "Inicio"))
```

```
Python 3.7.1 (default, Dec 10 2018, 22:54:23) [MSC v.1915 64 bit (AMD64)]
Type "copyright", "credits" or "license" for more information.

IPython 7.2.0 -- An enhanced Interactive Python.

In [1]: runfile('C:/Users/Odi/Downloads/Dijkstra.py', wdir='C:/Users/Odi/Downloads')
Nodos: ['Inicio', 'Rosa', 'Tulipan', 'Lirio', 'Girasol', 'Cactus', 'Suculenta', 'Agapanto']
Aristas: [('Inicio', 'Rosa'), ('Inicio', 'Tulipan'), ('Inicio', 'Lirio'), ('Rosa', 'Tulipan'), ('Lirio', 'Girasol'), ('Girasol', 'Tulipan'), ('Girasol', 'Cactus'), ('Cactus', 'Girasol'), ('Cactus', 'Agapanto'), ('Suculenta', 'Agapanto')]

Ruta mas corta con Dijkstra:
{'Inicio': ['Inicio'], 'Rosa': ['Inicio', 'Rosa'], 'Tulipan': ['Inicio', 'Rosa', 'Girasol', 'Tulipan'], 'Lirio': ['Inicio', 'Lirio'], 'Girasol': ['Inicio', 'Rosa', 'Girasol'], 'Cactus': ['Inicio', 'Rosa', 'Girasol', 'Cactus'], 'Agapanto': ['Inicio', 'Rosa', 'Girasol', 'Cactus', 'Agapanto']}
Longitud de la ruta mas corta:
{'Inicio': 0, 'Lirio': 2, 'Rosa': 4, 'Girasol': 5, 'Cactus': 5, 'Tulipan': 8, 'Agapanto': 9}

In [2]:
```

Pero en cambio, si el tulipán sí es recomendable, el peso de sus aristas cambiará y este subirá en la lista de prioridad:

```
3 import networkx as nx
4 G = nx.DiGraph()
5
6 # agregar nodos
7 G.add_node("Inicio")
8 G.add_node("Rosa")
9 G.add_node("Tulipan")
10 G.add_node("Lirio")
11 G.add_node("Girasol")
12 G.add_node("Cactus")
13 G.add_node("Suculenta")
14 G.add_node("Agapanto")
15
16 print ("Nodos: ", G.nodes())
17
18 # agregar aristas
19 G.add_edge("Inicio", "Rosa", weight=1)
20 G.add_edge("Inicio", "Tulipan", weight=3)
21 G.add_edge("Inicio", "Lirio", weight=2)
22 G.add_edge("Rosa", "Tulipan", weight=10)
23 G.add_edge("Rosa", "Girasol", weight=5)
24 G.add_edge("Lirio", "Girasol", weight=5)
25 G.add_edge("Girasol", "Tulipan", weight=5)
26 G.add_edge("Girasol", "Cactus", weight=5)
27 G.add_edge("Cactus", "Girasol", weight=5)
28 G.add_edge("Cactus", "Agapanto", weight=5)
29 G.add_edge("Suculenta", "Agapanto", weight=5)
30
31 print ("Aristas: ", G.edges())
32
33 # single source shortest path with Dijkstra
34
35 print ()
36 print ("Ruta mas corta con Dijkstra:")
37 print (nx.single_source_dijkstra_path(G, "Inicio"))
38
39 print
40 print ("Longitud de la ruta mas corta:")
41 print (nx.single_source_dijkstra_path_length(G, "Inicio"))
```

```
Python 3.7.1 (default, Dec 10 2018, 22:54:23) [MSC v.1915 64 bit (AMD64)]
Type "copyright", "credits" or "license" for more information.

IPython 7.2.0 -- An enhanced Interactive Python.

In [1]: runfile('C:/Users/Odi/Downloads/Dijkstra.py', wdir='C:/Users/Odi/Downloads')
Nodos: ['Inicio', 'Rosa', 'Tulipan', 'Lirio', 'Girasol', 'Cactus', 'Suculenta', 'Agapanto']
Aristas: [('Inicio', 'Rosa'), ('Inicio', 'Tulipan'), ('Inicio', 'Lirio'), ('Rosa', 'Tulipan'), ('Lirio', 'Girasol'), ('Girasol', 'Tulipan'), ('Girasol', 'Cactus'), ('Cactus', 'Girasol'), ('Cactus', 'Agapanto'), ('Suculenta', 'Agapanto')]

Ruta mas corta con Dijkstra:
{'Inicio': ['Inicio'], 'Rosa': ['Inicio', 'Rosa'], 'Tulipan': ['Inicio', 'Tulipan'], 'Lirio': ['Inicio', 'Lirio'], 'Girasol': ['Inicio', 'Rosa', 'Girasol'], 'Cactus': ['Inicio', 'Rosa', 'Girasol', 'Cactus'], 'Agapanto': ['Inicio', 'Rosa', 'Girasol', 'Cactus', 'Agapanto']}
Longitud de la ruta mas corta:
{'Inicio': 0, 'Lirio': 2, 'Tulipan': 3, 'Rosa': 4, 'Girasol': 5, 'Cactus': 5, 'Agapanto': 9}

In [3]:
```

Link de Github:

<https://github.com/KristenBrandt/Proyecto2>

Referencias:

- Bailey, D. A. (2007). *Java Structures: Data Structures in Java for the Principled Programmer*. Boston: McGraw-Hill
- Gorakala, S. K. (2016). *Building Recommendation Engines*. Birmingham: Packt Publishing.

- Ridwan, M (2019). Predicting Likes: Inside A Simple Recommendation Engine's algorithms. Recuperado de:
<https://www.toptal.com/algorithms/predicting-likes-inside-a-simple-recommendation-engine>