**Title:**

**Abstract**

Most recommender systems follow a request-response approach in which the recommendations are provided to the user upon his request. Recently a proactive recommender system - that pushes recommendations to the user when the current situation seems appropriate, without explicit user request - has been introduced in the research area of recommender systems. In this paper, a design of a context aware recommender system that recommends different types of items proactively under the Internet of Things paradigm is proposed. In this system, we have used a neural network that will do the reasoning of the context to determine whether to push a recommendation or not and what type of items to recommend.

1. **Introduction**

Traditional Recommendation systems usually requires user to submit query explicitly. In mobile environment, due to the limitation of mobile devices, data input and information browsing is inconvenient. A desirable solution is to proactively deliver relevant information to the user’s mobile device. A key challenge is to make information relevant within mobile user’s dynamic environment and automatically push interesting information to mobile user. Instead of asking users to input a query directly from their mobile device, the system filters and ranks new items based on user’s profile, location, available time, etc.

The proactive recommender system automatically delivers (i.e. pushes) recommendations to the user, without explicit request from him. The push model seems to be very effective in the applications where the availability of items changes often and rapidly, as it helps users timely receive their interested information. However, if the system pushes uninterested information to the user, or even pushes interested information to the user but at inappropriate context, the user’s acceptance of proactively delivered recommendations will decrease enormously. Hence for improving user’s acceptance in proactive recommender systems, determination of right context (situation assessment) and finding relevant items for the target user are very crucial.

The rest of the paper is organized as follows. Section 2 reviews the related work in this area. Section 3 presents…

1. **Related Works**
   1. **Push-Based Recommendations**

**News**

In [yeung2010proactive] presented a push-based news recommendation system named PPNews that recommend to mobile users personalized news articles in real time. The system proposed is based on user’s contextual information as well as news content. They developed a Bayesian network technique to predict user interest, as well as the Analytic Hierarchy Process (AHP) Model to determine the relevance of news articles, by using the algorithms of Content-based filtering (CBF) and Collaborative filtering (CF).

**Restaurants**

The paper presented in [bedi2012situation] proposes a Situation-Aware Proactive Recommendation System (SAPRS), which pushes relevant items to the user at the right context only. In a first phase, the system analyzes whether the current context is appropriate for making the recommendation. While in the second phase, the items are evaluated to be recommended to the user. The model proposed was developed to recommend restaurants to mobile users according to their preferences.

**Internet of Things**

In [salman2015proactive], the authors design a Context-Aware Recommendation System that recommends different types of items proactively under the Internet of Things (IoT) paradigm. On the one hand, they designed the Context Aware Management System (CAMS) to obtain, model and understand the data from the IoT. On the other hand, they used an artificial neural network (ANN) model to determine if the context is appropriate to recommend items to users and what type of recommendation to use.

**POI’s**

In automotive scenarios, the authors of [bader2010situation] proposed a proactive recommendation model that suggest Points-Of-Interest (POI), such as fuel stations, restaurants or parking places. Firstly, is identified the situation of the user from the data sensed about user, environment and device. Then, the presented model based on fuzzy logic is used to infer POIs to recommend by using the current situation and past situations of the user.

¿Qué no cubren estos trabajos que intentamos resolver nosotros?

* 1. **Trajectories-Based Recommendations**

¿Qué no cubren estos trabajos que intentamos resolver nosotros?

1. **Motivation**

El presente trabajo está inspirado en el Museo de Arte Moderna MoMA (*Museum of Modern Art*) de New York[[1]](#footnote-1), el cual está compuesto de 6 plantas. Cada planta contiene una sala con obras de arte, tales como painting, sculpture, printmaking, drawing, photography, architecture, design, film, etc., así como obstáculos como puertas para comunicarse entre las salas y escaleras para ir de una planta a otra. En el escenario de motivación los visitantes del museo cuentan con una aplicación móvil de recomendación que les sugiere de manera proactiva un itinerario de ruta a seguir para mirar, por ejemplo, cuadros y esculturas de su interés en un tiempo determinado.

Por ejemplo, supongamos que Bob quiere visitar el museo MOMA en 1 hora, que es el tiempo que dispone para ello. La aplicación de recomendación móvil, teniendo en cuenta el tiempo disponible de Bob, sus gustos en el pasado y su localización actual le recomienda automáticamente un conjunto de obras a mirar. Bob decide guiarse por las sugerencias de la aplicación y se dirige a la obra recomendada. Luego de apreciar la obra, Bob haciendo uso de la aplicación genera un voto a la obra con un valor de 4 en una escala de 1 a 5. La aplicación identifica que en la misma sala de Bob se encuentra otro visitante, al cual le puede propagar el voto de Bob automáticamente. De esta manera, el visitante sin esfuerzo alguno puede enriquecer el perfil almacenado en su móvil para futuras recomendaciones. A partir de este momento la aplicación del visitante es la responsable de propagar tanto sus votos como los recibidos por la aplicación en un tiempo determinado.

Luego de visitar varios cuadros de interés de una sala, Bob se dirige a la puerta de la sala para ir otra que no ha visitado aún. Después de generar la votación de varias obras visitadas en la sala actual, la aplicación móvil intenta propagar dichos votos, pero detecta que hay dos visitantes que no han recibido los votos de Bob. Dado el conflicto de intereses, la aplicación decide enviar la información de los ítems votados al visitante que se encuentra más distante de Bob. En lo adelante, la aplicación del visitante sería el responsable actual de propagar dichos votos lo más lejos posible. De manera recíproca Bob puede recibir votos de otros visitantes de la misma sala y enriquecer su perfil.

A pesar de gustarle las sugerencias de la aplicación móvil, Bob decide visitar un cuadro de interés que considera atractivo pero que no se encuentra dentro del itinerario recomendado por la aplicación. Automáticamente la aplicación detecta que Bob ha abandonado la ruta sugerida y actualiza la recomendación, generando una nueva ruta a seguir teniendo en cuenta su posición actual, preferencias en el pasado y el tiempo restante disponible. De esta manera, Bob recibe proactivamente recomendaciones de obras a visitar en un orden determinado hasta consumir el tiempo disponible para la visita.

1. **Simulator for the MOMA Museum**

Debido a la dificultad de probar un algoritmo de *Dynamic Context-Aware Recommendation* en un entorno real, hemos desarrollado una aplicación de simulación que permite simular un escenario de recomendación de obras de arte en el museo MOMA.

El simulador es una aplicación de escritorio desarrollada con el lenguaje de programación Java. Con el fin de facilitar su implementación, se utilizaron las siguientes API’s de Java:

* Jgraphx[[2]](#footnote-2)[[3]](#footnote-3): Facilitó la representación visual de las salas, pinturas, esculturas y usuarios del museo MOMA, mediante el uso de grafos.
* MOONRISE[[4]](#footnote-4): Framework que contiene un conjunto de modelos de Context-Aware Recommendation que facilitan el desarrollo de aplicaciones de recomendación para entornos móviles.
* Sqlite-jdbc: Permitió la conexión y gestión de las bases de datos de los usuarios.
  1. **Funcionalidades del simulador**

El simulador permite realizar un conjunto de funcionalidades, tales como:

* Load items: carga el fichero item.csv, que contiene información sobre los ítems. Dicha información es divida por en ficheros por planta
* Sacale and translate: escalar y/o trasladar el mapa cargado
* View floor 4: visualizar la estructura de la planta 4 con sus ítems correspondientes ubicados
* View floor 5: visualizar la estructura de la planta 5 con sus ítems correspondientes ubicados
* View combined floor: visualizar la estructura de ambas plantas
* Remove edges: elimina las paredes y pesos de las plantas, mostrando solo los puntos de las esquinas de cada sala
* Remove vertex labels: elimina las etiquetas de las esquinas de las salas
* Generate paths: genera un fichero con los paths de usuarios
* Start: inicia la simulación
* Stop: se detiene la simulación
* Pause: pausa la simulación
* Configuration: permite introducir o modificar parámetros de entrada
* Tooltips: permite visualizar información de un ítem seleccionado, por ejemplo: title, artist, type, room, location, etc.
  1. **Pre-procesamiento del dataset MOMA**

Para el proceso de simulación se utilizó el dataset MOMA[[5]](#footnote-5), que contiene solamente información de las obras de arte del Museo MoMA. El dataset está compuesto de 129 024 instancias y 29 atributos (e.g., title, artist, constituentID, artistBio, nationality, beginDate, endDate, gender, date, médium, dimensions, creditLine, accessionNumber, classification, department, dateAcquired, cataloged, objectID, URL, thumbnailURL, circumference (cm), depth (cm), diameter (cm), height (cm), length (cm), weight (kg), width (cm), seatHeight (cm), duration (sec.)).

Inicialmente el dataset de ítems fue pre-procesado para modificar algunos atributos, eliminar aquellos que no fueran de interés, así como para adicionar otros necesarios. Durante la fase de pre-procesamiento el dataset sufrió los siguientes cambios:

* Eliminación de las instancias donde los ítems no pertenecieran al departamento de "Painting & Sculture".
* Eliminación de los atributos URL y ThumbnailURL, al no ofrecer información útil para los objetivos del simulador.
* Eliminación de las instancias donde los artistas no pertenecieran a la planta 4 o 5
* Eliminación de los atributos que contiene información menor a un 6.67%, por ejemplo, los atributos: seatHeight y circumference, que no se tenía información de ninguna instancia; duration, que solo se tenía información de 16 instancias; y length, con información en 5 instancias.
* Los atributos que carecían de información se le dio valor null.
* Se hizo una binarización de los atributos:
* nationality: Spanish, French, Mexican, American, Dutch
* artist: AndyWarhol, EvaHesse, JacksonPollock, JasperJohns, LeeBontecou, RobertRauschenberg, RoyLichtenstein, ClaudeMonet, FridaKahlo, HenriMatisse, PabloPicasso, PaulCézanne, SalvadorDalí, VincentVanGogh
* classification: painting, sculpture
* Se discretizaron y binarizaron los atributos:
* date: EarlyCenturyXIX, MiddleCenturyXIX, LaterCenturyXIX, EarlyCenturyXX, MiddleCenturyXX, LaterCenturyXX, EarlyCenturyXXI, MiddleCenturyXXI, LaterCenturyXXI
* dateAcquired: LittleTime, LongTime
* Los valores del atributo Medium se unificaron en palabras más simples y se discretizaron luego con los siguientes atributos resultantes: Oil, Encaustic, SyntheticPolymer, Tempera, Metal, Casein, Silk, Gouache, SilkscreenInk, Enamel, Fabric, VariousMaterials, Felt, Steel, Bronze, Wood, Plaster, Glazed
* Incorporación del atributo Floor con los posibles valores: 4 y 5
* Incorporación del atributo EmotionTransmitted con los posibles valores: happy, sad, neutral
* Incorporación de los atributos item Location, cuyos valores fueron obtenidos mediante la aplicación Web WebPlotDigitizer[[6]](#footnote-6) y el generador de datos sintético DataGenCARS. El WebPlotDigitizer facilitó la obtención de las localizaciones de las esquinas de cada sala a partir de una imagen de una planta del museo. Por otra parte, el generador DataGenCARS fue extendido para incorporar un nuevo generador de atributos (ItemLocationAtrributeGenerator), que dado las localizaciones de las esquinas de una sala posibilite generar de manera uniforme las localizaciones de ítems en las paredes de la sala. Para ello, se aplicó el siguiente criterio: mientras quede espacio en las paredes se incorporan localizaciones con un separador por defecto de 20 pixeles. Cuando queden ítems por ubicar en las paredes y se haya acabado el espacio en las paredes, entonces serán ubicados aleatoriamente dentro de la sala.
* Incorporación del atributo Room con los valores de 1-14 para la planta 5, y de 15-26 para la planta 4

En el proceso de binarización el valor 1 significa la presencia de ese atributo en dicha instancia, y 0 la ausencia.

Como resultado de la fase de pre-procesamiento se obtuvo un total de 240 instancias de ítems. Por otra parte, se generaron 100 usuarios, 200 contextos, y 24 000 ratings. El número de ratings generado representa que cada usuario votó todos los ítems disponibles. La información de contexto y de ratings fue incorporada al dataset fue completado de manera offline para obtener un dataset acorde a un escenario de simulación.

Mediante el generador de datos sintéticos DataGenCARS[[7]](#footnote-7) [[8]](#footnote-8) se generaron los siguientes atributos de contexto (con sus posibles valores):

* mood: {happy, sad, neutral}
* temperatureRoom: {warm, hot, cold}
* numberPeopleRoom: {much, little, none}
* noiseLevel: {high, low, medium}

El atributo del ítem EmotionTransmitted también fue generado mediante DataGenCARS.

Además, se generaron los ratings de los usuarios. Para ello, se crearon 10 perfiles de usuarios teniendo en cuenta los siguientes atributos de ítems y de contexto:

|  |  |
| --- | --- |
| Ítems | Contexto |
| ArtistAndyWarhol  ArtistEvaHesse  ArtistJacksonPollock  ArtistJasperJohns  ArtistLeeBontecou  ArtistRobertRauschenberg  ArtistRoyLichtenstein  ArtistClaudeMonet  ArtistFridaKahlo  ArtistHenriMatisse  ArtistPabloPicasso  ArtistPaulCézanne  ArtistSalvadorDalí  ArtistVincentVanGogh  NationalitySpanish  NationalityFrench  NationalityMexican  NationalityAmerican  NationalityDutch  Gender  DateEarlyCenturyXIX  DateMiddleCenturyXIX  DateLaterCenturyXIX  DateEarlyCenturyXX  DateMiddleCenturyXX  DateLaterCenturyXX  DateEarlyCenturyXXI  DateMiddleCenturyXXI  DateLaterCenturyXXI  MediumOil  MediumEncaustic  MediumSyntheticPolymer  MediumTempera  MediumMetal  MediumCasein  MediumSilk  MediumGouache  MediumSilkscreenInk  MediumEnamel  MediumFabric  MediumVariousMaterials  MediumFelt  MediumSteel  MediumBronze  MediumWood  MediumPlaster  MediumGlazed  ClassificationPainting  ClassificationSculpture  DateAcquiredLittleTime  DateAcquiredLongTime | TemperatureRoom  NumberPeopleRoom  NoiseLevel |

Los 10 perfiles de usuario que se generaron son:

1. Pintura española de óleo, preferentemente de Pablo Picasso, inicios del siglo XX y que lleve tiempo en el museo. Le interesa que no haya personas en la sala y la temperatura de la sala sea agradable.

* ArtistPabloPicasso
* NationalitySpanish
* DateEarlyCenturyXX
* MediumOil
* ClassificationPainting
* DateAcquiredLongTime
* NumberPeopleRoom
* TemperatureRoom

1. Escultura española de bronce, preferentemente de Pablo Picasso, mitad del siglo XX, y que lleve tiempo en el museo. Le interesa que la temperatura de la sala sea agradable.

* ArtistPabloPicasso
* NationalitySpanish
* DateMiddleCenturyXX
* MediumBronze
* ClassificationSculpture
* DateAcquiredLongTime
* TemperatureRoom

1. Pintura francesa de óleo, preferentemente de Henri Matisse, inicios del siglo XX y que lleve tiempo en el museo. Le interesa que el nivel de ruido de la sala sea bajo.

* ArtistHenriMatisse
* NationalityFrench
* DateEarlyCenturyXX
* MediumOil
* ClassificationPainting
* DateAcquiredLongTime
* NoiseLevelRoom

1. Escultura americana de polímero sintético, preferentemente de Andy Warhol, mitad del siglo XX y que lleve tiempo en el museo. Le interesa que el nivel de ruido de la sala sea bajo.

* ArtistAndyWarhol
* NationalityAmerican
* DateMiddleCenturyXX
* MediumSyntheticPolymer
* ClassificationSculpture
* DateAcquiredLongTime
* NoiseLevelRoom

1. Pintura americana de óleo, preferentemente de Jackson Pollock, mitad del siglo XX y que lleve tiempo en el museo. Le interesa que no haya personas en la sala y el nivel de ruido de la sala sea bajo.

* ArtistJacksonPollock
* NationalityAmerican
* DateMiddleCenturyXX
* MediumOil
* ClassificationPainting
* DateAcquiredLongTime
* NumberPeopleRoom
* NoiseLevelRoom

1. Pintura americana de óleo, preferentemente de Robert Rauschenberg, mitad del siglo XX y que lleve poco tiempo en el museo. Le interesa que no haya personas en la sala.

* ArtistRobertRauschenberg
* NationalityAmerican
* DateMiddleCenturyXX
* MediumOil
* ClassificationPainting
* DateAcquiredLittleTime
* NumberPeopleRoom

1. Escultura americana de tela, preferentemente de Robert Rauschenberg, finales del siglo XX y que lleve poco tiempo en el museo. Le interesa que no haya personas en la sala.

* ArtistRobertRauschenberg
* NationalityAmerican
* DateLaterCenturyXX
* MediumFabric
* ClassificationSculpture
* DateAcquiredLittleTime
* NumberPeopleRoom

1. Pintura americana de óleo, de cualquier género, mitad del siglo XX y que lleve tiempo en el museo. Le interesa que no haya personas en la sala y la temperatura de la sala sea agradable.

* NationalityAmerican
* Gender
* DateMiddleCenturyXX
* MediumOil
* ClassificationPainting
* DateAcquiredLongTime
* TemperatureRoom
* NumberPeopleRoom

1. Pintura española de óleo, de cualquier género, mitad del siglo XX y que lleve tiempo en el museo. Le interesa que el nivel de ruido de la sala sea bajo.

* NationalitySpanish
* Gender
* DateMiddleCenturyXX
* MediumOil
* ClassificationPainting
* DateAcquiredLongTime
* NoiseLevelRoom

1. Escultura francesa de bronce, de cualquier género, inicios del siglo XX y que lleve tiempo en el museo. Le interesa que el nivel de ruido de la sala sea bajo.

* NationalityFrench
* Gender
* DateEarlyCenturyXX
* MediumBronze
* ClassificationSculpture
* DateAcquiredLongTime
* NoiseLevelRoom

De los 100 usuarios existentes, cada uno de ellos pertenece a uno de los 10 perfiles existentes. Por ejemplo, el usuario del 1 al 10 pertenecen al perfil de usuario 1, y así sucesivamente.

Con el fin de ajustar el rating generado a partir de los perfiles de usuario, se extendió DataGenCARS. Para ello, se implementó otra función de ajuste de rating para ser aplicado al rating resultante, teniendo en cuenta el atributo de contexto Mood y de ítem EmotionTransmitted. El criterio de la función es el siguiente:

(el peso puede ser + o -)

|  |  |  |
| --- | --- | --- |
| Context Attribute: Mood | Item Attribute: EmotionTransmitted | Weight |
| happy | happy | +1.0 |
| sad | sad | +1.0 |
| neutral | neutral | +1.0 |
| sad | happy | +0.5 |
| happy | sad | -0.5 |
| neutral | happy | +0.25 |
| neutral | sad | -0.25 |
| happy | neutral | -0.25 |
| sad | neutral | +0.25 |

* 1. **Generation of user paths**

La generación de los user paths se realizaron mediante el algoritmo **Nearest Point Of Interest Recommender (NPOI)**:

1. Para un number of users se generarán sus paths correspondientes.
   1. Por cada usuario, mientras no se consuma el time available for the users, el usuario estará caminando por las salas del museo observando ítems.
      1. La posición inicial del usuario es generada aleatoriamente, es decir, el usuario inicialmente puede aparecer en cualquiera de las salas.
      2. Se identifica la sala del usuario, así como los ítems candidatos a visitar (no vistos) dentro de ella.
      3. De la lista de ítems candidatos, se escoge el ítem próximo a visitar (el más cercano a la posición actual del usuario). Dicho ítem a visitar puede ser una pintura, escultura o puerta.

En el caso de que el ítem a visitar sea una puerta, entonces el usuario debe salir de la sala actual (aunque queden ítems pendientes por visitar) y dirigirse a otra sala. Si la sala a la cual se dirige ha sido visitada en su totalidad, entonces deberá dirigirse a otra puerta (de otra sala), donde queden ítems disponibles para visitar. En el caso de que el usuario haya visitado todos los ítems pertenecientes a las salas que se encuentran conectadas a la sala actual, entonces se escoge aleatoriamente una puerta para salir.

* + 1. El tiempo empleado para moverse de un ítem a otro (currentTime) es determinado de la siguiente manera:

currentTime+=time(item1, item2);

Donde, time=distance(item1, item2)/userVelocity();. No obstante, si se mueve de una puerta a otra, pero pasando por una escalera, entonces: time= timeOnTheStairs();

Si el ítem visitado es una painting o sculpture, entonces al currentTime se le adiciona además el delay observing painting or sculpture, quedando de la siguiente manera:

currentTime+=time(item1,item2)+ delayObserving();

Los valores de las siguientes variables son introducidos como parámetros de configuración:

* Number of users
* Time available for the users
* Delay observing painting or sculpture
* User velocity
* Time on the stairs
  1. **Dynamic Context-Aware Recommendation Approach**

El algoritmo **Trajectory-based Context-Aware Recommendation Postfiltering (T&CARP)** recomienda ítems (pinturas y esculturas), teniendo en cuenta la localización, las preferencias y el time available for the current user visitar el museo. Este enfoque, está basado en el paradigma **Context-Aware Recommendation Postfiltering**. Ambos, se encuentran implementados en el Framework MOONRISE (MObile cONtext-aware RecommendatIon SystEm).

**Trajectory-based Context-Aware Recommendation Postfiltering algorithm**:

1. Mientras que haya time available for the current user visitar el museo.
   1. Se aplica un traditional recommendation algorithm para la obtención de los K items to recommend (ítems candidatos) al current user.
   2. Los ítems candidatos son filtrados a partir de un minimum threshold of recommendation. En el caso de que ninguno de los ítems candidatos supere el minimum threshold of recommendation, entonces los ítems candidatos no son filtrados (items filtrados = items candidatos).
   3. Los ítems filtrados son ordenados teniendo en cuenta la localización del current user de la siguiente manera:
      1. Se genera un grafo con todos los ítems (pinturas y esculturas) existentes en el museo.
      2. Mediante el grafo, se determina el camino mínimo entre todas las combinaciones posibles de pares de ítems filtrados. El peso de las aristas sería la distancia euclidiana entre los pares de ítems filtrados.
      3. Se genera un nuevo grafo (llamado subgrafo) completo con los ítems filtrados solamente. El peso de cada arista del subgrafo será la longitud de cada camino mínimo obtenido entre los pares de ítems filtrados.
      4. Se determina el camino mínimo del subgrafo pasando por todos los ítems filtrados, fijando el ítem de partida. Para esto, se utiliza la teoría de un grafo Hamiltoniano.
      5. Los ítems resultantes, se recomiendan al current user.

Los valores de las siguientes variables son introducidos como parámetros de configuración:

* Current user ID
* Time available for the current user
* Traditional recommendation algorithm
* K items to recommend
  1. **Proceso de simulación:**

Aspectos a tener en cuenta sobre el algoritmo de simulación **Trajectory and User-Based Collaborative Filtering (T&UBCF)**:

* Previo a la ejecución de la simulación, los **paths de los usuarios** son generados de manera offline y almacenados en un fichero. La aplicación facilita la opción de utilizar el mismo fichero para diversas simulaciones. No obstante, lospathscontenidos en dicho fichero pueden actualizarse (o volverse a generar) cuantas veces sean necesarias mediante la aplicación.
* El **path del current user** es generado y actualizado en tiempo de simulación (ver **T&UBCF algorithm**).
* Aplicamos una **recomendación híbrida** del tipo **Switching** para atacar el **comienzo frío**. Alternamos entre los algoritmos de recomendación **NPOI** y **T&CARP**, en dependencia de la información existente en la base de datos del current user.
* El algoritmo **T&CARP** aplicaun **User-based Collaborative Filtering** basado en la **User Neighborhood**, aplicando la **Pearson Correlation** como medida de similitud.
* Inicialmente, en la base de datos del current user: todos los usuarios (incluyendo el current user) tienen información del ítem ficticio=400 con un rating=5.
* El current user inicialmente partirá de la puerta de entrada 241, que está en la sala 5 del museo.

**T&UBCF algorithm:**

1. Por cada iteración del hilo de ejecución.
   1. Se **actualiza la posición** de todos los usuarios:
      1. Por cada usuario:
         1. Mientras que el (realTimePerIteration>0) & (currentTime < timeAvailableForTheCurrentUser):
            1. Se obtiene el userPath:

* Si el usuario no es el current user, entonces se obtiene el userPath del fichero generado offline.
* Si el usuario es el current user, entonces se obtiene userPath mediante un **algoritmo de recomendación**:
* Si no hay información suficiente en la base de datos, entonces se aplica el algoritmo de recomendación **NPOI** pero solo para el current user.
* Si hay información suficiente en la base de datos, entonces se aplica el algoritmo de recomendación **T&CARP.**
  + - * 1. Si el userPath no ha llegado a su fin.
* Se obtiene del userPath la artista actual (ítem partida y ítem destino).
* Si el usuario es el current user y el ítem destino es una puerta, entonces se **actualiza la recomendación**.
* Si el usuario (incluyendo el current user) estaba mirando un ítem y le queda tiempo en la iteración para votar, entonces:
* Se almacena el rating generado para ese ítem en una base de datos centralizada.
* Si el usuario es el current user, entonces se incrementa el currentTime+=time(item1,item2)+ delayObserving();
* Si el usuario (incluyendo el current user) no estaba mirando un ítem, entonces:
* Si llego al ítem destino y el usuario es el current user, entonces se incrementa el currentTime+=time(item1, item2);
* Si aún no ha llegado al ítem destino, entonces se obtiene la próxima posición del usuario.
  + - * 1. Si el userPath ha llegado a su fin y el usuario es el current user, entonces se **actualiza la recomendación**.
      1. Se detiene el hilo de ejecución.
  1. Se **actualiza la recomendación** del usuario actual, si se cumple con la siguiente condición:

Si (cantidad de ratings recibidos en la base de datos del usuario actual >= number of ratings received) & (currentTime >= minimum time to update recommendation)

* 1. Se **pinta la posición actual** de todos los usuarios en la pantalla de la aplicación.
  2. Se **duerme el hilo** durante un screen refresh time.

1. **Experiments**
   1. **Configuration parameters**

|  |  |
| --- | --- |
| **Parameter** | **Default value** |
| Number of users | 100 |
| Time available for the users | 1 hour |
| User velocity | 3 Km/h |
| Delay observing painting or sculpture | 30 seconds |
| Time on the stairs | 60 seconds |
| Current user ID | 100 |
| Time available for the current user | 1 hour |
| Traditional recommendation algorithm | User-based Collaborative Filtering |
| K items to recommend | 10 |
| Minimum threshold of recommendation | 2.5 |
| Real time per iteration | 30 seconds |
| Iteration time/Screen refresh time | 1 second |
| Simulation resolution | 1 second |
| 1Km represents | 6597 pixels |
| Number of ratings received | 40 |
| Minimum time to update recommendation | 30 seconds |

**Explicación de los valores de configuración:**

* Number of users: La cantidad de usuarios a los cuales se les generará un path (o que es lo mismo que el número de paths a generar).
* Time available for the users: Tiempo disponible para los usuarios visitar el museo.
* Delay observing painting or sculpture: El tiempo que se demora un usuario en observar una pintura.
* User velocity: Velocidad promedio de un usuario caminando.
* Time on the stairs: Tiempo que se demora en ir de una puerta a otra pasando por una escalera.
* Current user ID: Identificador del usuario actual, al cual se le recomendarán ítems a visitar.
* Time available for the current user: Tiempo disponible para el usuario actual visitar el museo.
* Traditional recommendation algorithm: Algoritmo de recomendación tradicional (User-based Collaborative Filtering, SVD or Random).
* K items to recommend: El número máximo de ítems a recomendar al usuario actual.
* Minimum threshold of recommendation: Umbral mínimo de recomendación.
* Real time per iteration: Tiempo real que se simula en una iteración.
* Iteration time/Screen refresh time: Tiempo de refrescamiento de la pantalla.
* Simulation resolution: Resolución de la simulación.
* 1Km represents: Cuántos píxeles representan 1Km. Para simular las distancias del museo MOMA se ha realizado un escalado de a , asumiendo que una planta del museo tiene .
* Number of ratings received: Número de ratings recibidos en la base de datos del usuario actual. Este parámetro es utilizado para activar el algoritmo de recomendación.

Por ejemplo:

* Minimum time to update recommendation: Tiempo mínimo transcurrido para actualizar la recomendación.

Ejemplo de utilización de los 2 últimos parámetros de configuración:

Si (cantidad de ratings recibidos en la base de datos del usuario actual >= number of ratings received) & (currentTime >= minimum time to update recommendation), entonces la recomendación es actualizada.

* 1. **Baselines:**

Los siguientes baselines se compararon con el **algoritmo de simulación T&UBCF**. De igual manera al algoritmo **T&UBCF**, los baselines parten de la puerta de entrada 241 de la sala 5.

* **Random recommender (RAND):**

1. Mientras que el (currentTime < timeAvailableForTheCurrentUser):
   1. Se identifica la sala donde está la puerta de entraday se obtienen los ítems disponibles (pintura, escultura o puerta)a visitar en dicha sala.
   2. Si aún quedan ítems disponibles en la sala actual:
      1. Se escoge aleatoriamente un ítem a visitar.
      2. Si el ítem a visitar es una pintura o una escultura, entonces es visitado y se determina el currentTime+=time(item1,item2)+ delayObserving();
      3. Si el ítem a visitar es una puerta, entonces debe dirigirse a otra sala y se determina el currentTime+=time(item1,item2);
   3. Si no quedan ítems disponibles en la sala actual, entonces se escoge aleatoriamente otra puerta para irse a otra sala.

* **Exhaustive visit recommender (ALL):** visita todos ítems de cada sala, empezando por la sala 1 hasta que el currentTime > timeAvailableForTheCurrentUser.
* **Nearest Point Of Interest Recommender (NPOI):** fue explicado previamente
* **Know-It-All recommender (Know-It-All):** se lleva a cabo el **algoritmo de simulación T&UBCF** descrito anteriormente,pero la base de datos del current user:
* Contiene toda la información (items votados) por cada usuario.
* Inicialmente no se tiene ninguna información (items votados) del current user.
* **K-Ideal Recommender (K-Ideal):**
* Utiliza la BD con todo los ratings de los usuarios (menos el del especial, que no tiene nada inicialmente).
* Algoritmo:

1. Los ítems no vistos por el usuario especial se ordenan por el voto (siendo el de mayor ranking el de mejor voto)
2. De los items ordenados por voto, se escogen los K items, sin filtrarse por umbral.
3. Los K items resultantes son ordenados, teniendo en cuenta la trayectoria a seguir.
4. El path resultante (los K items ordenados por trayectoria) es el utilizado para la simulación (de 1 hora). Dicho path será actualizado (con la idea previamente descrita) cuando:

* El ítem a visitar sea una puerta.
* Se acabe el path y aún quede tiempo disponible para la visita.

1. **Conclusions**

In this paper we presented a …

1. **Bibliographic**

**Experimentos para MOMM 2017**

* db\_museum (BD generada por DataGenCARS): contiene 176 usuarios, 240 ítems, 9 contextos y 380160 ratings (176 usuarios X 240 ítems X 9 contextos).

**Baselines:**

* **Random recommender (RAND):**
* db\_user (BD del usuario especial -176-): inicialmente no contendrá nada. No obstante, para que no esté vacía se pondrá una instancia ficticia (ejemplo, 176, 400, 9, 5, null, 1).
* Algoritmo de recomendación:

1. Se obtiene aleatoriamente de db\_museum una primera lista con todos los ítems del usuario especial 176.
2. Se obtiene de db\_user una segunda lista con los ítems vistos por el usuario especial 176.
3. Se obtiene una lista resultante con los K ítems de la primera lista pero que no se encuentren en la segunda lista.

* Notas: La lista resultante no será ordenada por trayectoria, ni será filtrada por umbral.
* **Exhaustive visit recommender (ALL):** visita todos ítems de cada sala, empezando por la sala 1 hasta que el currentTime > timeAvailableForTheCurrentUser.
* **Nearest Point Of Interest Recommender (NPOI):**

1. Se identifica la sala del usuario especial, así como los ítems candidatos a visitar (no vistos) dentro de ella.
2. De la lista de ítems candidatos, se escoge el ítem próximo a visitar (el más cercano a la posición actual del usuario). Dicho ítem a visitar puede ser una pintura, escultura o puerta.

En el caso de que el ítem a visitar sea una puerta, entonces el usuario debe salir de la sala actual (aunque queden ítems pendientes por visitar) y dirigirse a otra sala. Si la sala a la cual se dirige ha sido visitada en su totalidad, entonces deberá dirigirse a otra puerta (de otra sala), donde queden ítems disponibles para visitar. En el caso de que el usuario haya visitado todos los ítems pertenecientes a las salas que se encuentran conectadas a la sala actual, entonces se escoge aleatoriamente una puerta para salir.

* **Know-It-All recommender (Know-It-All):**
* **K-Ideal Recommender (K-Ideal):**

1. <https://www.moma.org/> [↑](#footnote-ref-1)
2. <https://github.com/jgraph/jgraphx> [↑](#footnote-ref-2)
3. <https://jgraph.github.io/mxgraph/docs/manual_javavis.html#1.3> [↑](#footnote-ref-3)
4. <http://webdiis.unizar.es/~maria/?page_id=250> [↑](#footnote-ref-4)
5. <https://github.com/MuseumofModernArt/collection> [↑](#footnote-ref-5)
6. <http://arohatgi.info/WebPlotDigitizer/app/> [↑](#footnote-ref-6)
7. <http://www.sciencedirect.com/science/article/pii/S157411921630270X> [↑](#footnote-ref-7)
8. <http://webdiis.unizar.es/~maria/?page_id=70> [↑](#footnote-ref-8)