TSCF - Predicción de trayectorias de usuario en el transporte público

Felipe Osimani y Leonardo Piñeyro
Docentes Matías Richart y Jorge Visca

Estructura de la presentación

- Introducción al proyecto
-
- Algoritmo trabajado
- Modelos de movilidad utilizados
- Código desarrollado
- Resultados
- Conclusiones y trabajo a futuro

Introducción al proyecto

- Recrear la investigación de Fengli Xu Et Al. sobre la recuperación de trayectorias de usuarios a través de datos agregados de torres de celulares
- Generar un ambiente simulado que logre recrear los datos reales
- Recuperar las trayectorias utilizando el algoritmo propuesto y comparar los resultados con el paper
- Utilizando lo aprendido en la investigación, aplicarlo al problema de recuperación de recorridos de usuarios en el sistema de transporte público
- Buscamos poder predecir recorridos de usuarios individuales a partir de datos de entradas/salidas de los ómnibus en cada parada

Introducción al trabajo de Fengli Xu Et Al.

- Utilizan datos agregados del sistema de redes celulares o aplicaciones móviles
- Estos datos los presentan como cantidad de usuarios por torre celular en cada unidad de tiempo
- Fundamentándose en que las trayectorias humanas son muy repetitivas y únicas dicen lograr recuperar trayectorias individuales
- La privacidad entonces no se conservaría luego de agregar los datos

Estructura de datos

Los datos se presentan de la siguiente forma

	Torre A	Torre B	Torre C
P^{t0}	5	3	2
P^{t1}	3	4	3
P^{t2}	1	7	2

Luego se construye una **distribución** los usuarios en una estructura L, para cada unidad de **tiempo**

$$L^{t_0} = [A_0, A_1, A_2, B_0, B_1, B_2, B_3, B_4, C_0, C_1]$$

En cada iteración del algoritmo se van formando las **trayectorias**

$$S^t = [s_1^t, s_2^t, \quad \dots \quad , s_N^t]$$

$$s_j^t = [q_j^1, q_j^2, \quad \dots \quad , q_j^t]$$

Algoritmo

El algoritmo va a iterar en las unidades de tiempo, asignando cada valor del L actual a un usuario particular. Lo hace de la siguiente forma:

- En el tiempo t + 1, tenemos la siguiente **distribución de posiciones**

$$L^{t+1} = [A_0, A_1, A_2, B_0, B_1, B_2, B_3, B_4, C_0, C_1]$$

- Calculamos el costo l_j^{t+1} de un usuario al elegir un elemento particular dentro de la distribución, moviéndose de donde está ahora a esa torre
- En el ejemplo, agregar B_0 a la trayectoria del usuario 1 se expresaría de la siguiente manera $s_1^t \rightarrow l_3^{t+1} = s_1^t \rightarrow B_0$ y calcularíamos el costo de dicho movimiento

Algoritmo (cont.)

Esta iteración de hace para todos los **usuarios**, y todos las **entradas de la distribución** en ese **tiempo** particular, logrando una matriz de costos:

$$C^t = \{c^t_{i,j}\}_{N \times N},\ N = \#users$$

$$C^t = \begin{bmatrix} c^t_{11} & c^t_{12} & \cdots & c^t_{1N} \\ \vdots & \ddots & & \\ c^t_{N1} & c^t_{N2} & \cdots & c^t_{NN} \end{bmatrix}$$

Algoritmo (cont.)

- Con dicha matriz de costos, vamos a calcular una matriz de decisión para elegir un nuevo elemento en cada traza de cada usuario para esta iteración.
- El problema se convierte entonces en un problema de minimización

$$Minimize \sum_{i=1}^{N} \sum_{j=1}^{N} c_{i,j}^{t} \times x_{i,j}^{t}$$

- Este tipo de problema ya es conocido como un **problema de asignación** y lo solucionaremos con el **algoritmo del Húngaro** que es el mejor algoritmo hallado para la resolución de estos problemas.

Cálculo de probabilidades

El único elemento faltante para describir la recuperación de trayectorias es el cálculo de costos de moverse de un lugar a otro. Para esto, los autores dividen el problema en dos escenarios, día y noche, y se basan en patrones de movimiento de los humanos para calcularlo

Cálculo de probabilidades - noche (0hs - 6hs)

Las personas tienden a **quedarse en un mismo lugar** durante la noche, por lo tanto vamos a predecir que el siguiente elemento de la ruta va a ser la última torre visitada

$$\tilde{l}_i^{t+1} \equiv q_i^t$$

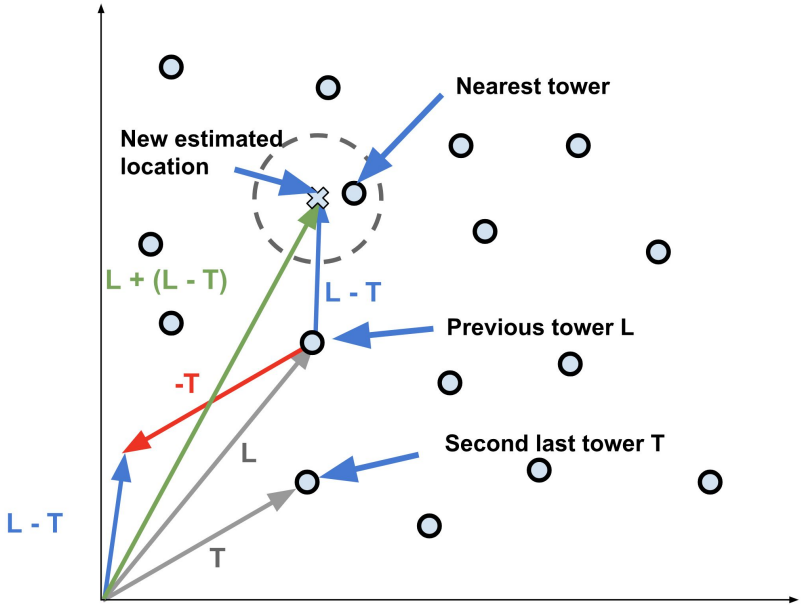
Con esto, construimos una matriz de costos con la distancia entre esta torre y todas las demás, haciendo que se tienda fuertemente a quedarse en el mismo lugar (costo 0)

Cálculo de probabilidades - día (6hs - 24hs)

Para el día, los autores asumen que los usuarios tienden a seguir su trayectoria con la misma velocidad, por lo tanto, para predecir la siguiente torre visitada, van a utilizar las **dos últimas torres.**

Cálculo de probabilidades - día (6hs - 24hs) (cont.)

Por lo tanto predecimos de la siguiente forma:



Asignación de trayectorias

- Una vez recuperadas todas las trayectorias de todos los días y noches, se enlazan una con otra para formar una trayectoria general de cada usuario
- Esta trayectoria se hace probando una con todas las otras, y quedándose con la que tenga menos *information gain*, en otras palabras, que tiene menos entropía (menos diferencia espacio/temporal)

Evaluación

Luego de finalizado el algoritmo, los autores mapean de forma greedy las trayectorias recuperadas con las reales. Para evaluar el resultado utilizan las siguientes tres métricas:

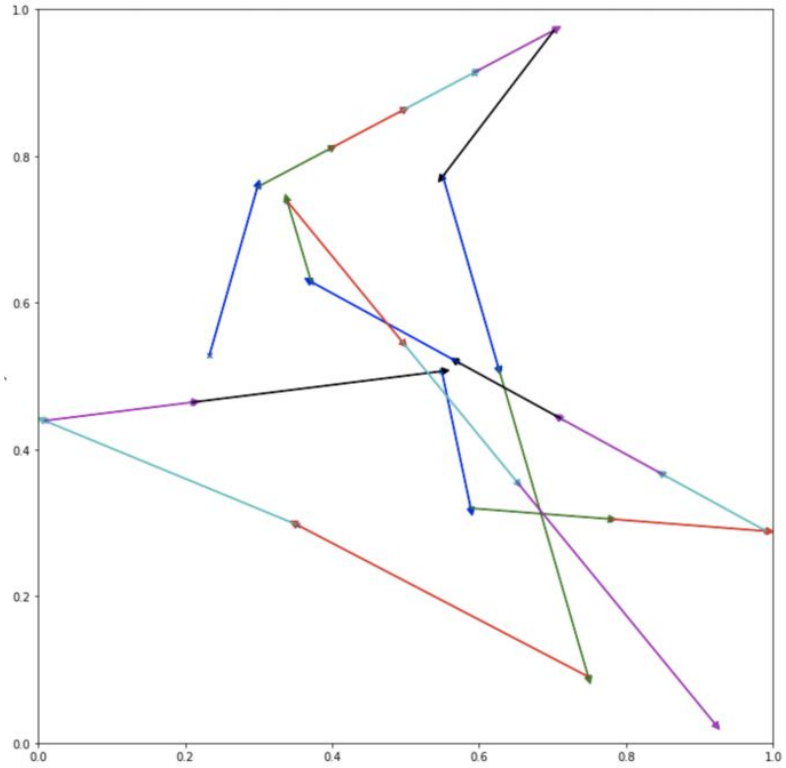
- **Accuracy:** número de puntos en los resultados espacio-temporales que coinciden con la trayectoria real asignada
- **Recovery error**: calcula el error geográfico entre la trayectoria recuperada y la real a la que se asignó. Para cada punto espacio-temporal, el error es la distancia entre ellos.
- **Uniqueness**: la utilizan para evaluar qué tan fácil se puede identificar a un usuario particular a partir de sus *k* lugares más visitados

Modelos de movilidad

Para simular las trayectorias utilizamos dos modelos de movilidad ya estandarizados (random waypoint y random direction) y creamos uno para el problema.

Modelos de movilidad (cont.)

Random waypoint



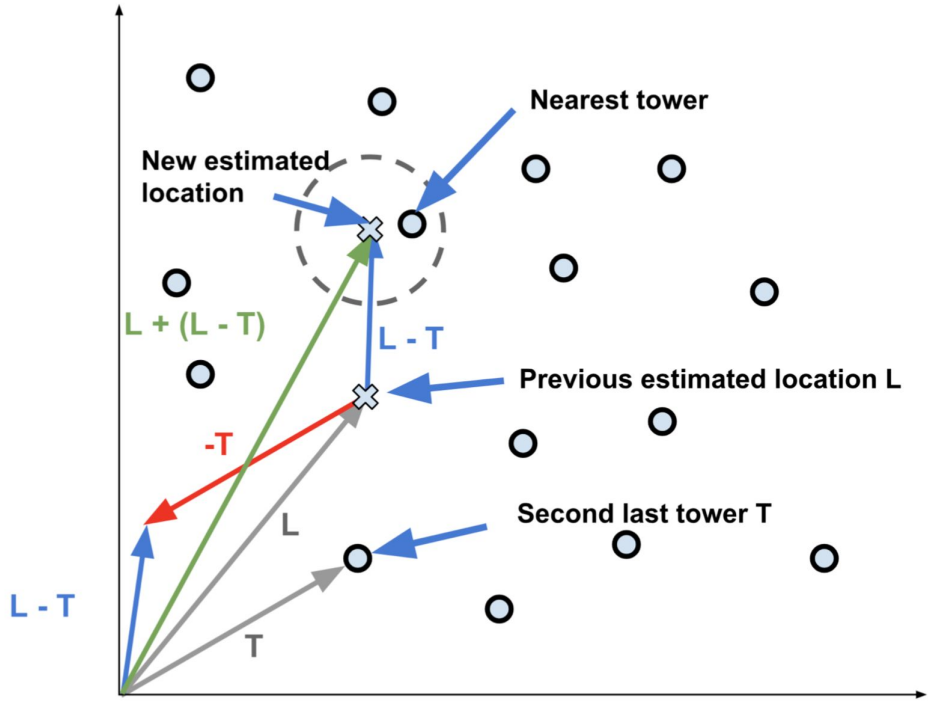
Modelos de movilidad (cont.)

También desarrollamos un modelo de movilidad que intenta mantener la misma dirección y velocidad, pero con probabilidad de cambiarla en algún momento.

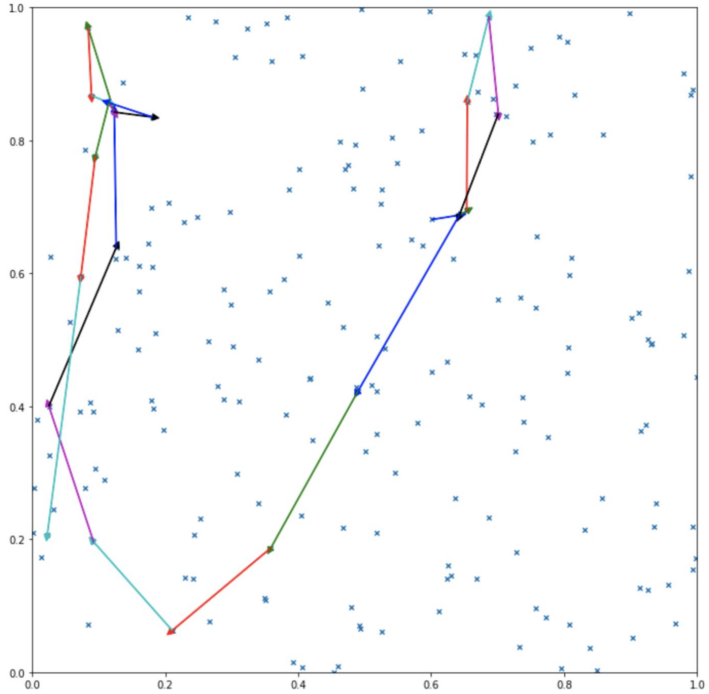
Nos basamos en la probabilidad de saltar de una torre a otra, asignándole una probabilidad a cada una, para luego elegir una basado en dichas probabilidades.

Para esto utilizamos una función expansora y softmax, logrando tener una sumatoria de probabilidades de 1.

Modelos de movilidad (cont.)



Modelos de movilidad (cont.)



Estructura del código

Dividimos el código en cuatro clases independientes, para facilitar la evaluación y la prueba con diferentes modelos de mobilidad

- TowersManager
- TraceSimulator
- MobilitySimulator
- TrajectoryRecovery

Resultados

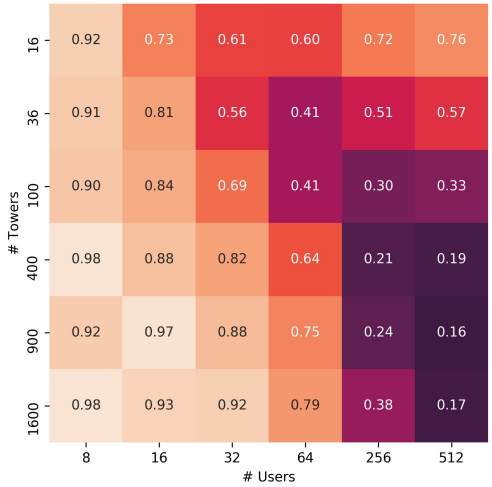
- Evaluamos el algoritmo repetidas veces en entornos de simulación con diferentes parámetros
- Dentro de los parámetros escogidos tenemos: cantidad de **usuarios**, cantidad de **torres**, número de **ciclos**, **velocidad** de los usuarios, **modelo de movilidad**.

Estructura de pruebas

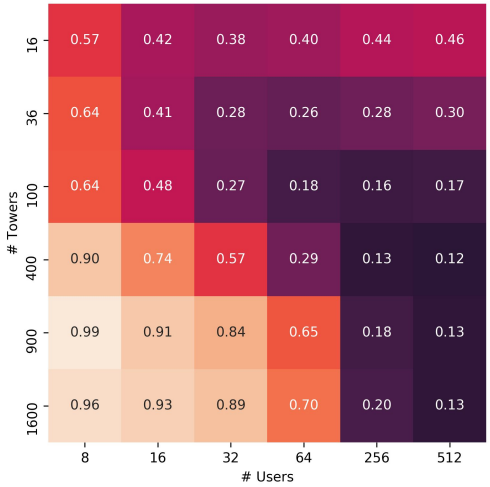
- Primero corremos la simulación una una combinación de parámetros, lo que genera las trazas de los usuarios.
- Luego agregamos los datos para obtener para cada torre la cantidad de usuarios en cada momento.
- Finalmente aplicamos el algoritmo de recuperación de trazas y analizamos su rendimiento.

Pruebas realizadas

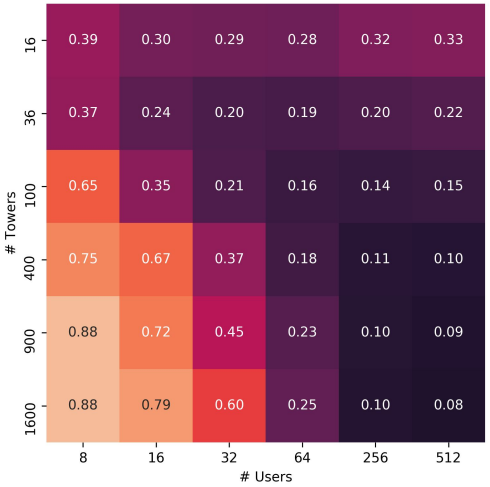
- **Accuracy** obtenida de pruebas con el modelo *Random Direction*



vel = 0.01



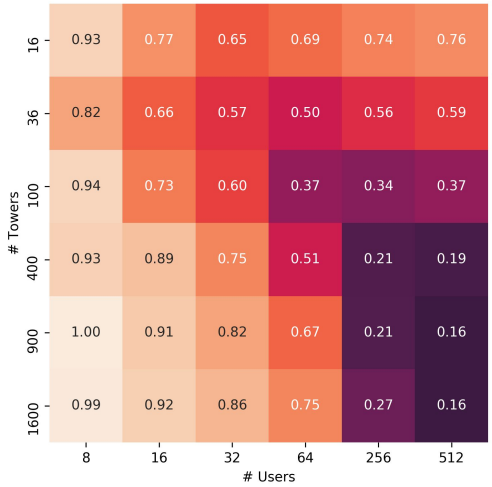
vel = 0.04



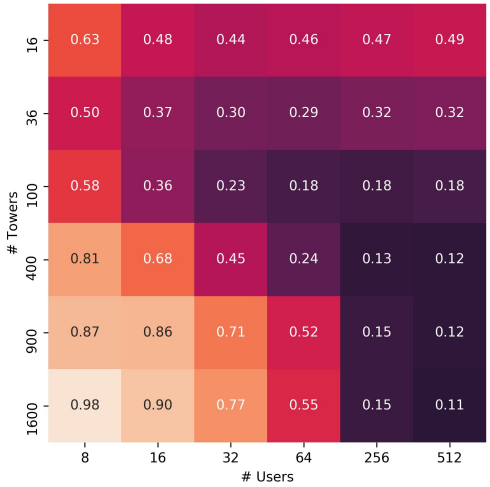
vel = 0.1

Pruebas realizadas (cont.)

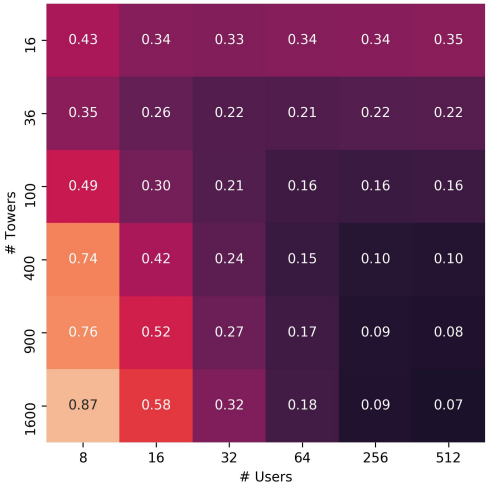
- **Accuracy** obtenida de pruebas con el modelo *Random Waypoint*



vel = 0.01



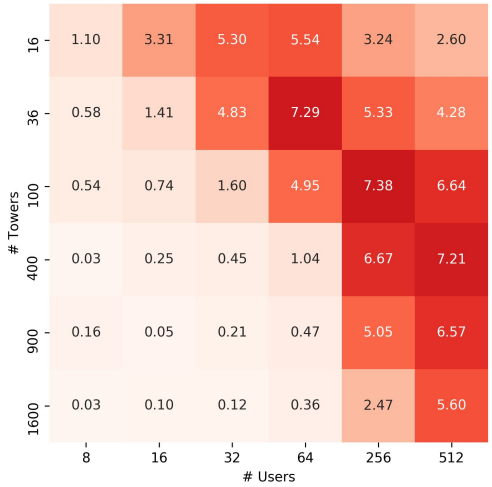
vel = 0.04



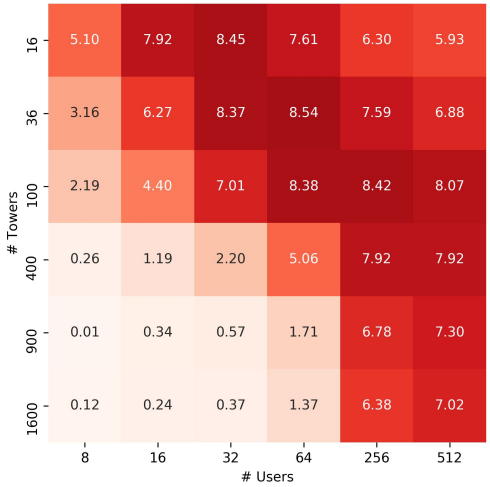
vel = 0.1

Pruebas realizadas (cont.)

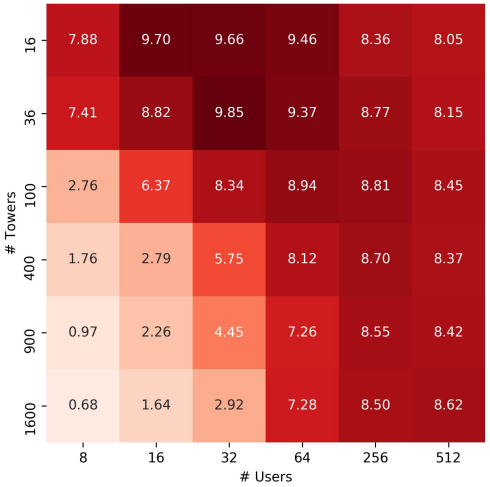
- **Recovery error** obtenido de pruebas con el modelo *Random Direction*



vel = 0.01



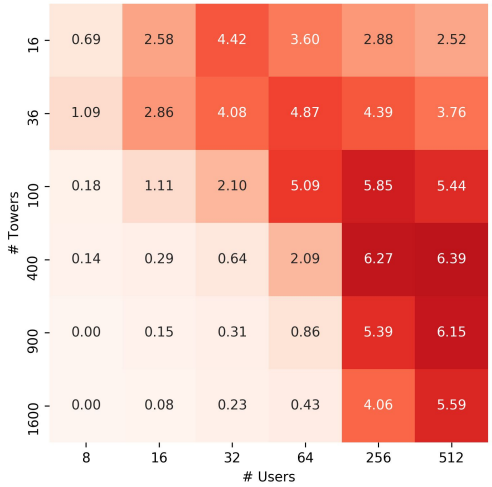
vel = 0.04



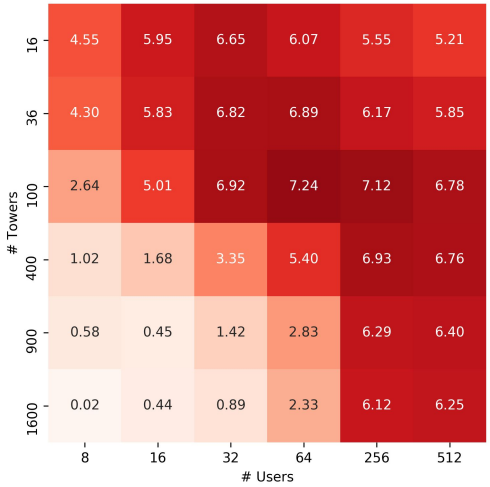
vel = 0.1

Pruebas realizadas (cont.)

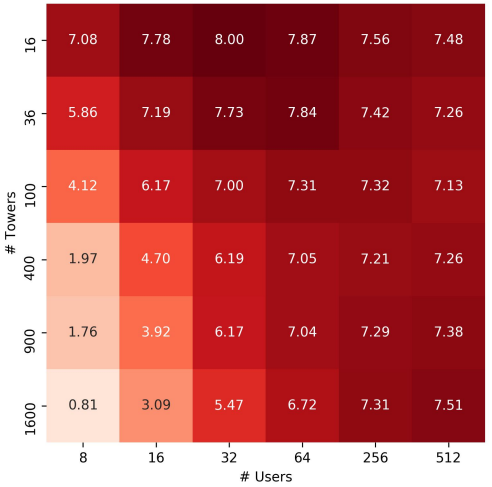
- **Recovery error** obtenido de pruebas con el modelo *Random Waypoint*



vel = 0.01



vel = 0.04



vel = 0.1

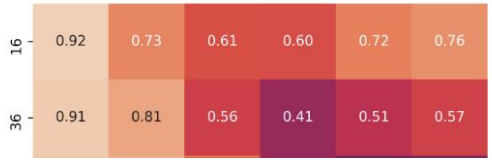
Conclusiones y trabajo futuro

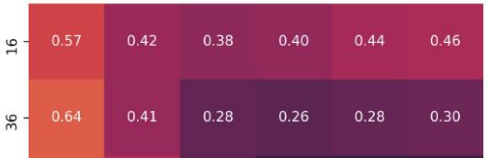
Presentamos una primer versión de las conclusiones que sacamos a partir de los resultados obtenidos:

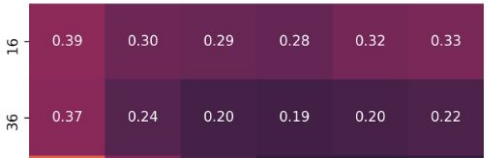
- Claramente, cuanto más usuarios y torres agreguemos, menos la accuracy y mayor el error
  - La cantidad de usuarios afecta más que la cantidad de torres
- Esto parece seguir los resultados obtenidos por los autores
  - En el paper, presentan una accuracy de 73% al 92% con torres agrupadas en “distritos” y números mucho peores para distritos más chicos
  - Esto es replicable a nuestra variación de cantidad de torres. 16 torres pueden ser vistas como 1600 torres en 10 distritos

Conclusiones y trabajo a futuro (cont.)

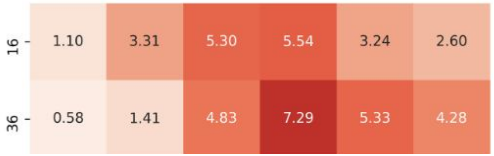
- Algo interesante a rescatar es que parecería ser que si bien el accuracy disminuye con la cantidad de usuarios, después de determinado punto parecería crecer, para poca cantidad de torres







- Lo mismo sucede con el error



Conclusiones y trabajo a futuro (cont.)

Quedan pendiente para el desarrollo del proyecto:

- Terminar de evaluar los resultados obtenidos
  - Con esto, tener una idea clara de cuáles son las variables que más afectan los resultados del algoritmo
- Crear un modelo para trabajar con el problema de las paradas
  - Utilizar las conclusiones recabadas para lograr el mejor modelo posible
  - Trabajar dicho modelo para evaluar los posibles resultados de un algoritmo similar, para predecir trayectorias de usuarios en los ómnibus