ISSN 1870-4069

Reentrenamiento continuo de un algoritmo LSTM con un sistema multiagente que implementa el modelo BDI para la predicción de viajes en bicicleta

Ramon A. Briseño, Juan C. López, Rocio Maciel Arellano,
Víctor M. Larios, Raul J. Beltrán, J. Antonio Orizaga T.

Universidad de Guadalajara,
Centro Universitario de Ciencias Económico Administrativas,
Doctorado en Tecnologías de Información,
Centro de Innovación en Ciudades Inteligentes,
México

{alejandro.bmartinez, juan.lopez6936}@alumnos.udg.mx,
{raul.beltran, jose.orizaga}@academicos.udg.mx,
{rmaciel, vmlarios}@cucea.udg.mx

Abstract. En este documento se describe el desarrollo un sistema multiagente, con un modelo BDI encargado de mantener entrenado continuamente un algoritmo de redes neuronales recurrentes de corta y larga memoria, el cual predice por día la cantidad de viajes en sistemas de bicicleta pública para el desarrollo del proyecto de la Smart Citiy de la Zona Metropolitana de Guadalajara. Para los algoritmos de aprendizaje automático, es necesario someterlos a reentrenamiento cada que el conjunto de datos aumenta de manera considerable o en tareas en las que se requiere análisis de datos para realizar predicciones en tiempo real. El proceso consiste en la obtención, depuración y estructuración de nueva información, reentrenamiento del algoritmo de aprendizaje automático con este nuevo conjunto de datos y el reporte de resultados. Con algunas adecuaciones al modelo Belief-Desire-Intention (BDI) mediante la plataforma multiagentes SPADE, se implementó un agente que tiene como propósito ejecutar el proceso de aprendizaje automático del algoritmo para la predicción. Este agente calibra de manera automática la longitud de la secuencia de entrada que requiere el algoritmo. Utilizando el modelo BDI, el agente predictor consigue que el algoritmo mantenga de manera sistemática una precisión mayor al 85% en la exactitud, contando con la capacidad de aumentar su eficacia en un 5% con un módulo en otro agente planificador basado BDI. Este sistema multigente, demostró ser modular, esaclable, reutilizable y con la autonomía suficiente para efectuar una predicción confiable sin necesidad de agregar más desarrollo de software después de su implementación. Además, demostró que el modelo BDI en los sistemas multiagentes, es una herramienta poderosa para interactuar con algoritmos de aprendizaje automático.

Keywords: Datos Masivos, Minería de Datos y Analíticos, Red Neuronal LSTM, Predicción, Sistema Multiagente con Modelo BDI, Movilidad Sustentable, Smart Cities, Tecnologías de Información, E-World.

1. Introducción

Con la urbanización acelerada, uno de los problemas más palpables está asociado al incremento de tráfico de automóviles privados saturando las vialidades y con ello un incremento en la contaminación ambiental. Una estrategia para combatir este problema, es incentivar movilidad limpia en lo posible, a través de ifraestructura de ciclovías y con una red de estaciones en un sistema de bicicleta pública. Este es el caso de la Zona Metropolitana de Guadaljara (ZMG). Sin embargo, la adopción nos es fácil, sobre todo si las ciclovías no son seguras y en Guadalajara se ha sufrido un incremento de muertes de ciclistas por atropellamiento aumentando este índice en un 54% comparando el año 2018 con el 2019 [8].

Actualmente, la ZMG cuenta con casi 100 Km de Ciclovías y para que esta solución se integre al proyecto de Smart City de la región [9], es necesario que se puedan simular diferentes scenarios para que el sistema sea más seguro y atractivo. El programa MiBici es como se denomina la red de estacioens de bicicletas púbicas ZMG. Las bicicletas están disponibles para su renta por día o por año mediante una membresía. Se realizaron alrededor de 4,658,208 viajes en el 2019, con un total de 85,143 usuarios registrados en el programa hasta junio de 2020 [7]. El programa MiBici publica en su sitio web, datos abiertos sobre la cantidad de viajes que se efectúan por día, por mes y por año. El programa MiBici se toma como objeto de estudio de esta investigación, dada su importancia en el uso de medios de transporte sustentable.

Este trabajo propone escalar este sistema de bicicleta pública en una primera etapa para poder hacer predicciones de operación que lo haga más eficiente y responsivo a las necesidades de los ciudadanos. Para esto, integraremos una solución de simulación con multiagentes que se entrenarán a través de los datos reales producidos en la red de bicicletas públicas para generar optimizaciones. Un agente es una unidad de software autónoma [2], la toma de decisiones y la comunicación son algunas de sus principales características, también se pueden organizar con otras unidades de software, para realizar tareas conjuntas.

Por lo que, un sistema multiagente (MAS) es un entorno donde más de un agente convive y trabaja de manera coordinada para resolver un objetivo en común [3]. Los sistemas MAS han incorporado el aprendizaje automático y al analisis de la movilidad, durante la revisión del estado del arte, destaca un sistema multiagente en el programa de sistema de bicicleta compartida en salamanca [1], este predice la demanda de bicicletas y prevé una visualización en tiempo real de los viajes en curso.

El sistema MAS descrito en este trabajo se encarga de predecir la cantidad de viajes por día que cada estación tendrá como punto de finalización e inicio de un viaje, esta cantidad de viajes resultante de la predicción será analizada para una posterior identificación e inclusión de variables a un modelo de predicción de accidentes de movilidad para ciclo vías de la ZMG. El MAS desarrollado obtiene y estructura información del portal MiBici, reentrena un algoritmo de redes neuronales recurrentes de corta y larga memoria (LSTM) y finalmente postea resultados de la predicción de viajes.

Este sistema utiliza el modelo BDI (Believe, desire and intentions) para dotarlo de informarción que le permite calibrar la longitud de la secuencia de entrada de manera

autónoma, con el fin de que el sistema ofrezca de manera sistemática una exactitud mayor al 85% en su precisión a lo largo del tiempo y sin la intervención del diseñador del software.

2. Propuesta para la predicción de viajes en bicicleta

Este presente trabajo de investigación describe, la automatización de un proceso de aprendizaje automático para un algoritmo de predicción y la interacción que tiene el MAS con el algoritmo (LSTM) de red neuronal recurrente de corta y larga memoria al ajustar la longitud de la secuencia de entrada, cada agente está diseñado para tomar decisiones de acuerdo a lo establecido en su modelo BDI.

En primer término, el sistema obtiene la información necesaria mediante *scraping*[1], despúes ejecuta tareas de normalización y depuración de datos no relevantes para generar un conjunto de datos estructurados, este conjunto de datos es sometida a entrenamiento de un algoritmo, el resultado es la predicción de la cantidad de viajes por día que tendrán las estaciones del programa MiBici de la ZMG, el sistema multiagente calibra la longitud de la secuencia de entrada del algoritmo de aprendizaje automático y algunos parámetros del algoritmo como la cantidad de neuronas y épocas, esto se logra a partir del modelo BDI de su arquitectura, La autonomía de un agente generalmente significa que un agente opera sin ntervención u orientación humana directa (u otra) [12], este modelo hace que la toma de decisiones de los agentes sea más cercana a la que tomaría el diseñador del sistema, ya que este plasma su actuar en los planes del modelo [4], lo que le da una mayor sensibilidad al sistema.

Para la predicción el MAS, utiliza un algoritmo de red neuronal recurrente de corta y larga memoria (LSTM), utilizado generalmente para el análisis y predicción de series temporales donde se observa un mejor desempeño en la precisión comparado con otros algoritmos de aprendizaje automático [10]. El experimento contempló una serie de tiempo de una sola variable “la cantidad de viajes por día”. Para la predicción, el algoritmo recibe una secuencia de valores de viajes por día como entrada y arrojará un solo valor de salida, mismo que representa la cantidad de viajes que tendrá el día posterior a los días introducidos como entrada.

El MAS se desarrolló en la plataforma SPADE [5], un entorno de desarrollo en Python que además permite escalar de forma distribuida conectando nodos de procesamiento según sea necesario. El modelo BDI desarrollado funciona para la plataforma SPADE 3 y Python 3, miestras que el algoritmo LSTM es ejecutado con Keras y Tesorflow.

2.1. Arquitectura del sistema multiagente

El sistema multiagente consta de 5 agentes, un agente para la recolección de datos, uno para la depuración, uno para el reporteo y dos para gestionar la predicción mediante

[1] *Scraping*, o web scraping es una técnica para obtener código html de paginas web de manera automática, se procesan los documentos html obtenidos, se navegan por las etiquetas y se guardar la información requerida.

el modelo BDI. El agente scraper, el agente depurador, el agente predictor y el agente reporteador siempre están vivos a la espera que suceda un evento o de recibir una notificación para realizar su tarea.

La secuencia de eventos se describe a cotinuación:

1. El agente scraper monitorea el portal web del programa Mibici de manera periódica (una vez al día), una vez que encuentra nuevos datos abiertos de viajes, los descarga a una carpeta del servidor de donde se ejecutó MAS. Cada que el agente scraper descarga nueva información notifica al agente depurador.
2. Después de recibir la notificación del agente scraper, el agente depurador se encarga de formar un conjunto de datos con la información en la estructura que el algoritmo LSTM necesita. Si ya existe un conjunto de datos el agente depurador solo depura la información nueva y la agrega al conjunto de datos ya existente. Una vez que el conjunto de datos contiene la información nueva el agente depurador notifica al agente predictor.
3. Después de recibir notificación del agente depurador, el agente predictor crea al agente BDI.
   (a) El agente BDI se encarga de entrenar y encontrar la longitude de la secuancia de entrada indicada para obtener una buena predicción. Una vez entrenado el algoritmo y encontrado la longitud de la secuencia de entrada el agente BDI escribe los resultados de la predicción en diferentes archivos en la carpeta del sistema, después notifica al agente reporteador y muere, el agente predictor sigue a la espera de nuevas notificaciones.
4. Después de recibir notificación del agente BDI, el agente reporteador muestra los resultados de la predicción en un sitio web, donde se muestran entre otra información estadística, la gráfica de validación con la que se calculó el MAPE y la predicción correspondiente al día posterior a los datos iniciales.

En la figura 1 se observa el diagrama de funcionamiento del MAS. A continuación, se detalla el funcionamiento de cada agente.

Agente scraper: Los navegadores web son útiles para ejecutar código javascript, mostrar imágenes y presentar la información en un formato de fácil lectura para las personas, sin embargo, los web scrapers son excelentes para obetener y procesar largas cantidad de información rápidamente, es decir, que en lugar de estar revisando una página web a la vez, es posible estar visualizando los datos de miles de páginas web al mismo tiempo [11]. La búsqueda de información se realiza de manera periódica a la web informativa del programa MiBici, dentro de su sección de blog se publican en ligas de descarga archivos csv con los datos organizados por año y meses. El agente scraper ejecuta una función de consulta a esa web en busca de nueva información. Para obtener las ligas de decarga se utiliza un complemento html para la separación del documento en sus etiquetas.

Si el agente encuentra uno o más posts con datos de viajes que aún no se encuentran en la carpeta datasets del sistema multiagente, los descarga y los guarda. La

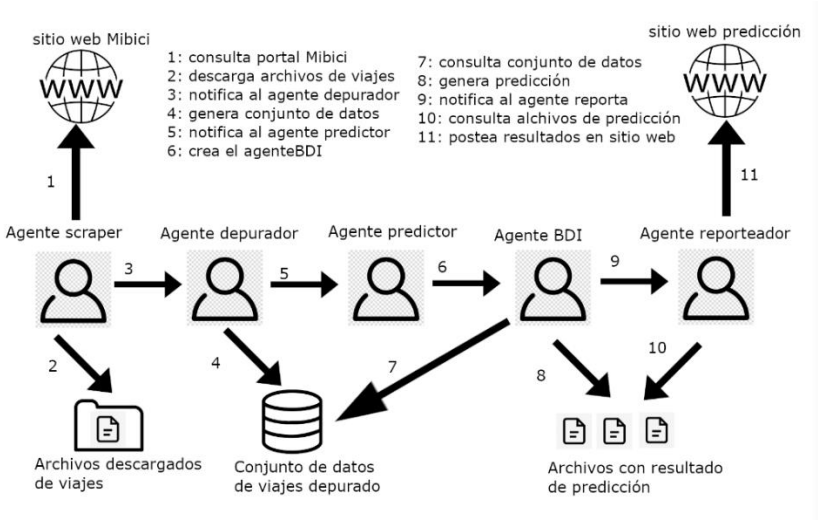

Fig. 1. Ilustración que representa el funcionamiento y la arquitectura del MAS.

información se organiza en carpetas por año y por meses, finalmente los archivos descargados se renombran y se guardan en formato csv.

Agente depurador: Este agente se encuentra todo el tiempo esperando la notificación del agente scraper. Una vez notificado, el agente revisa si hay un conjunto de datos depurado, si este no existe, el agente lo genera con todos los archivos descargados por el agente scraper. Si ya existe un conjunto de datos depurado, el agente verifica si hay archivos nuevos descargados por el agente scraper que no estén incluidos en el conjunto de datos, si se encuentran archivos nuevos, el agente los depura y agrega dicha información al conjunto de datos unificado y depurado. La generación de los archivos finales de inicio y finalización de viajes por estación consiste en realizar un ciclo que revisa cada fila y paralelamente tiene un contador de viajes por cada día y generar un archivo con dos columnas, la primera columna con la fecha en tipo fecha y la segunda de tipo entero misma que contendrá la cantidad de viajes por cada día. El conjunto de datos se guarda como archivo en formato csv con los nombres de columnas fecha y viajes.

Agente predictor: Al igual que el anterior, este agente en todo momento se encuentra a espera de la notificación del agente depurador. Una vez recibida la notificación de que el conjunto de datos está listo para que el algoritmo de aprendizaje automático se entrene, el agente predictor crea al agente BDI. El agente predictor se mantiene en espera hasta que el agente BDI termina su tarea. Cuando el agente BDI termina su tarea también termina su ciclo de vida, por lo que el agente predictor regresa a la espera de una nueva notificación.

Agente BDI: Este es el único agente del sistema MAS que implemente el modelo BDI, de allí fue nombrado “agente BDI”. Las tareas de los demás agentes del sistema pueden

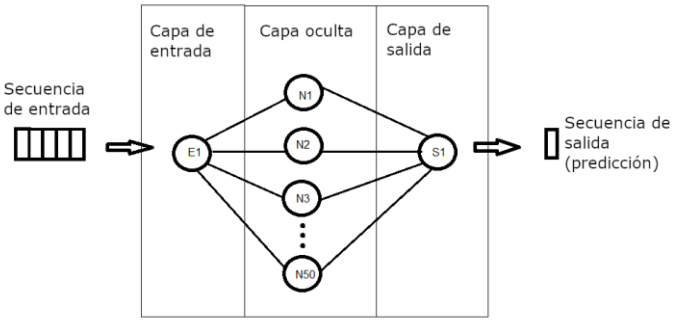

Fig. 2. Diagrama de bloques del algoritmo LSTM utilizado.

ser resueltas fácilmente con los tipos de comportamientos que ofrece la plataforma SPADE, como el comportamiento cíclico, el periódico, el comportamiento de una sola ejecución y el comportamiento de máquina de estados. Sin embargo, este agente necesita un planificador más inteligente para lograr el objetivo de entrenar el algoritmo de aprendizaje automático y encontrar la longitud de la secuencia de entrada indicada para que la predicción tenga un porcentaje de exactitud confiable.

En el modelo BDI implementado contiene 4 conceptos fundamentales. (1) **Belief**: el cual es un conocimiento que el agente tiene sobre su entorno. Se puede agregar, eliminar y consultar beliefs en el momento que se requiera. (2) **Goals**: es el objetivo que el agente perseguirá. En el caso del agente BDI el objetivo es encontrar la longitud de la secuencia de entrada indicada para que la predicción tenga un porcentaje de exactitud mayor al 85%. Originalmente el modelo BDI de la versión de SPADE 2.0 con python 2 [6] acepta múltiples objetivos. (3) **Services**: los servicios son pequeñas tareas que van agregando Beliefs a la base de conocimientos del sistema MAS. (4) **Plans**: estos son planes en los que se define una cantidad de servicios con los que se pretende llegar a cumplir el objetivo. Los servicios se agregan al plan en el orden que se desea que estos ejecuten su tarea. El modelo original pretendía que la tarea ejecutada por el último servicio fuera la que cumpliera con el objetivo. Sin embargo, en la adaptación para este trabajo se flexibilizó el sistema para que cualquier servicio pudiese cumplir el objetivo y omitir los servicios pendientes de un plan.

El sistema BDI implementado funciona ejecutando un cicló el cual busca los planes y ejecuta los que tienen como finalidad el objetivo del agente. Una vez encontrado un plan este ejecuta todos los servicios que el plan contiene, dichos servicios van agregando beliefs a la base de conocimientos. Si alguno de los servicios de un plan alcanza el objetivo definido en el goal, el sistema para y el agente BDI notifica al algente reporteador. Si ningún servicio de un plan logra el objetivo, el agente continúa con la ejecución de los servicios del siguiente plan encontrado, los servicios de los planes en ejecución utilizan los conocimientos que dejaron los servicios de los planes que fallaron. De esta forma cada plan que falla ayuda a que el siguiente plan tenga mayor probabilidad de llegar al objetivo y así de manera sistemática. El modelo BDI

adaptado para este trabajo se configuro para que en el momento que todos los planes fallen, se identifique el servicio obtuvo el mejor acercamiento al objetivo en términos porcentuales.

La primera implementación del algoritmo de aprendizaje automático, se definieron como parámetros los 11 últimos meses del conjunto de datos para validación, 20 épocas de procesamiento, 50 neuronas activas en una capa oculta, MAPE como métrica para medir confiabilidad de la predicción y una secuencia de valores de entrada con tamaño entre 1 a 50 valores, donde la primera secuencia utilizada fue de tamaño 5. En el agente BDI se consideraron 4 planes, 13 servicios y un objetivo. Con el objetivo de alcanzar un MAPE menor al 15%, la descripción de los planes, servicios y el objetivo quedaron de la siguiente forma:

- Goal: El objetivo es que al tener datos nuevos y reentrenar el algoritmo LSTM el MAPE sea menor al 15%.
- Plans: El agente cuenta con 4 planes. El primer plan cuenta con un servicio, los tres posteriores cuentan con 4 servicios cada uno. Al ser un agente con un solo objetivo todos los planes tienen el mismo objetivo en común.
- El primer plan consiste en entrenar el algoritmo con la longitud de la secuencia de entrada que se tiene definida por el diseñador del sistema. La longitud de entrada definida solo cambiará si este plan falla y se comienza con la búsqueda de un nuevo tamaño de secuencia de entrada. Este plan solo cuenta con un servicio. El servicio entrena el algoritmo con la longitud de la secuencia de entrada definido y obtiene la métrica MAPE. Si el MAPE obtenido es menor a 15% se alcanza el objetivo y termina la ejecución del agente BDI. Si el objetivo no es alcanzado, el servicio guarda los valores de MAPE obtenido y tamaño utilizado, así como un belief que confirma que el servicio se ejecutó con éxito a pesar de no cumplir el objetivo.
- El segundo plan consiste en entrenar el algoritmo aumentando y disminuyendo longitud de la secuencia de entrada definida por el diseñador del sistema. Este plan es clave, ya que en caso de no conseguir el objetivo dará información vital a los planes posteriores. El plan cuenta con 4 servicios. Los sercivios consisten en entrenar el algoritmo con la longitud de la secuencia de entrada definida aumentada y disminuida en 10 y 20 unidades y obtieninedo la métrica MAPE. Si el MAPE obtenido por algun servicio es menor al 15% se alcanza el objetivo y termina la ejecución del agente DBI. Si el objetivo no es alcanzado y los valores de MAPE y longitud de la secuencia de entrada de alguno de estos servicios dieron mejor resultado que el MAPE y longitud de la secuencia de entrada de los servicios anteriores estos se guardan. Proceso realizado de manera sistematica con cada servicio.
- Tanto el tercero como el cuarto plan consisten en entrenar el algoritmo con la mejor longitud de la secuencia de entrada hasta el momento, aumentada y disminuida en 6, 3, 2 y 1 unidades. Al igual que en el Segundo plan cada servicio revisa si cumplio el objetivo y en caso de no hacerlo, verifica si sus resultados son los mejores hasta el momento con el fin de que estos sean usados por los servicios posteriores. Si ninguno de los servicios ofrecidos por los 4 planes logra

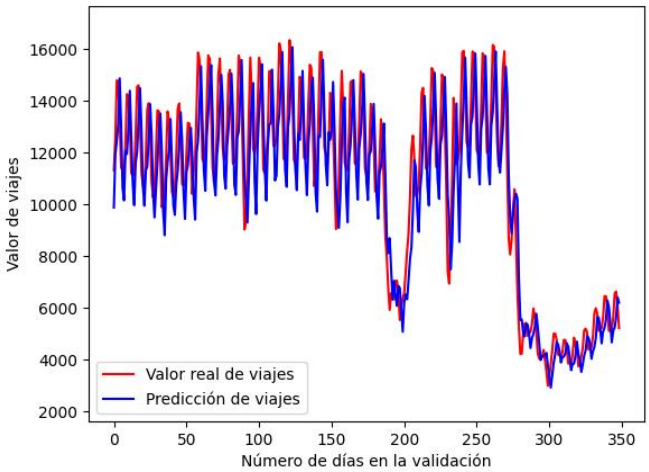

Fig. 3. Gráfica de los valores de predicción en contraste con los meses de validación del conjunto de datos (obtención de 11% en MAPE).

el objetivo de obtener un MAPE menor al 15% el agente entrena el algoritmo con la longitud de la secuencia de entrada que dio el resultado más acercado al objetivo y termina su ejecución. Cada que el agente termina su ejecución el valor de la longitude de la secuencia de entrada definida por el diseñador del sistema cambió al valor que alcanzo el objetivo o al que dio el mejor resultado, lo que provocará un incremento sistemático en la aproximación al objetivo para el siguiente agente BDI.

Agente reporteador: Este agente se encuentra en todo momento a espera de la notificación del agente predictor. Una vez recibida la notificación de que la predicción ha sido realizada ejecuta su tarea. La tarea del agente reporteador es presentar en una página web la predicción de viajes por estación y las gráficas generadas por el agente BDI donde se ilustra la confiabilidad de la predicción.

3. Pruebas y resultados

El conjunto de datos está integrado de 1917 registros correspondientes al periodo del 1 de diciembre del 2014 al 31 de mayo del 2020. Para disminuir el impacto negativo en la predicción causada por la dispersión y los datos atípicos que presenta la serie de tiempo, se le aplicó un suavizado con una media móvil con un tamaño de ventana en 3 valores. Se utilizó el 83% de los datos para el entrenamiento y el 17% de los datos para pruebas y validaciones. Para la reproducción del experiemento, tanto el código fuento del modelo BDI creado y la arquitectura multiagente se encuentra en un repositorio Git donde los investigadores interesados pueden pedir acceso al mismo contactando a cualquier autor de este artículo.

Para encontrar la longitud de la secuencia de entrada definida con el que comenzara a correr el MAS en producción, el agente BDI se ejecutó 11 veces con longitudes de secuencias de entradas predefinidas con valores de 20 hasta 30 unidades. La longitud de la secuencia de entrada que arrojo la mejor métrica fue 5, obteniendo un MAPE del 11%.

4. Conclusión y trabajos futuros

El sistema multiagente demostró ser eficiente en la automatización del proceso de predicción de la cantidad de viajes por estación del programa MiBici. Aunado a eso, el MAS complementado con el modelo BDI tuvo la capacidad de interactuar con uno de los parámetros del algoritmo LSTM para buscar una predicción estable a lo largo del tiempo, obteniendo una exactitud en la precisión del 89% en las pruebas realizadas. El MAS, con la autonomía de cada uno de sus agentes, facilitó la ejecución continua de un proceso complejo sin la intervención del diseñador del sistema después de su implementación.

Al calibrar la longitud de la secuencia de entrada el agente BDI aumenta su posibilidad de alcanzar el objetivo de tener un MAPE menor al 15% en la precisión de la predicción, Sin embargo, no asegura que el objetivo sea alcanzado. Por otra parte, es muy útil, ya que en las pruebas realizadas para encontrar la longitud de la secuencia de entrada que tendría definida el MAS en su puesta en producción, se observó que de entre 1 a 50 unidades que pudo tomar la longitud de la secuencia de entrada, la predicción tiene una variación promedio del 5% en el MAPE, por lo que se concluye que si el valor definido de la longitud de la secuencia de entrada en algún momento no llegase a cumplir el objetivo, existe una ventana de aproximadamente el 5% de mejoramiento del MAPE ejecutando los planes del agente BDI.

Cabe remarcar que esta es la primera etapa de este proyecto para que el sistema público de bicicletas de las ZMG converja con otras soluciones del proeycto de Smart City logrando identificar las estaciones de la red y predecir como será su carga de uso. Esto puede servir para eficientar la logística de balanceo de cargas en estaciones para que siempre haya bicicletas disponibles. También para predecir densidad de uso de ciclovías y tener medidas para resguardar la seguridad en ellas cuando estas tienen más demanda mediante mayor supervisión de cámaras de video y agentes de tránsito. La vida de los ciudadanos no tiene costo y poder predecir es una contribución. Si bien estos dos ejemplos se pueden resolver con algoritmos de gráfos y optimización tradicional, la solución propuesta se implementa más rápido y el aprendizaje automático de los agentes genera una solución más simple y de menor complejidad computacional com lo describe [13].

En cuanto a la técnica desarrollada, como trabajo futuro se pretende ampliar este trabajo donde el modelo BDI utilizado calibre de manera automática más parámetros del algoritmo de aprendizaje automático. Aunado a ello, se pretende utilizar una ontología que describa del funcionamiento del algoritmo de aprendizaje automático LSTM. Se busca que el agente BDI utilice dicha ontología para que las tareas que

realizan los servicios estén fundamentadas en el conocimiento del algoritmo otorgado por la ontología y se puedan cumplir objetivos de manera más eficiente e inteligente.

Referencias

1. Lozano, A., et al.: Multi-agent system for demand prediction and trip visualization in bike sharing systems. Applied Sciences 8.1, 67 (2018)
2. Oja, M.,Tamm, B.,Taveter, K.: Agent-based software design. Proc. Estonian Acad. Sci. Eng, 7(1), pp. 5−21 (2001)
3. Lee, L.C., et al.: The stability, scalability and performance of multi-agent systems. BT Technology Journal, 16(3), pp. 94−103 (1998)
4. Araiza-Illan, D., Pipe, T., Eder, K.: Model-based testing, using belief-desire-intentions agents, of control code for robots in collaborative human-robot interactions. arXiv preprint arXiv:1603.00656 (2016)
5. SPADE: SPADE, Smart Python Agent Development Environment. https://spade-mas.readthedocs.io/en/latest/readme.html (2020)
6. Aranda, G., Palanca, J., Escriva, M., Criado, N., Garcia-Pardo, J.A.: SPADE User's Manual. http://www.javierpalanca.com/spade/manual/ (2020)
7. MiBici: Datos abiertos. Explora, experimenta y comparte los datos de uso de MIBICI. https://www.mibici.net/es/datos-abiertos/ (2020)
8. Carapia, F.: Periódico MURAL, Matan a más ciclistas pese a las ciclo vías. https://www.mural.com/matan-a-mas-ciclistas-pese-a-las-ciclovias/ar1861611?referer=--7d616165662f3a3a6262623b727a7a7279703b767a783a-- (2020)
9. Gobierno de Zapopan: Presenta Pablo Lemus estrategia de ciclovías emergentes en sesión de Junta de Coordinación Metropolitana. https://www.Guadalajara.gob.mx/v3/noticias/presenta-pablo-lemusestrategia-de-ciclovias-emergentes-en-sesion-de-junta-de-coordinacion (2020)
10. Han, Z., et al.: A review of deep learning models for time series prediction. IEEE Sensors Journal (2019)
11. Mitchell, R.: Web scraping with Python: Collecting more data from the modern web. O'Reilly Media, Inc. (2018)
12. Wooldridge, M., Jennings, N.R.: Agent theories, architectures, and languages: a survey. In International Workshop on Agent Theories, Architectures, and Languages, Springer, Berlin, Heidelberg, pp. 1−39 (1994)
13. Chiariotti, F., et al.: A dynamic approach to rebalancing bike-sharing systems. Sensors, vol. 18(2), pp. 512 (2018)