UNIVERSIDAD AUTÓNOMA DE MADRID
ESCUELA POLITÉCNICA SUPERIOR

Grado en Ingeniería informática

TRABAJO FIN DE GRADO

Ayuda al diagnóstico de incidencias en grandes infraestructuras de TI

Autor: Rodrigo De Pool
Tutor: Javier Aracil

julio 2020

Rodrigo De Pool
Ayuda al diagnóstico de incidencias en grandes infraestructuras de TI

Rodrigo De Pool
C\ Francisco Tomás y Valiente Nº 11

IMPRESO EN ESPAÑA – PRINTED IN SPAIN

Resumen

La monitorización de red es una componente esencial para garantizar el rendimiento y disponibilidad de grandes centros de datos. Sin embargo, mientras las redes de ordenadores crecen y se tornan más complicadas, la inspección tradicional basada en umbrales fijos e inspección manual, resulta cada vez más costosa. Este trabajo examina la posibilidad de un sistema que detecte y notifique automáticamente de incidencias en grandes redes de ordenadores, aligerando la carga de un gestor de red y agilizando la subsanación de incidencias.

Los datos de red que se recogen en la comunicación entre un servidor y sus clientes, contienen una amplia información sobre el funcionamiento del propio servidor. Utilizando estos datos, el proyecto explorará dos modelos: Uno basado en redes LSTM (*Long short-term memory*) y otro basado en métodos auto-regresivos. Dado un servidor concreto, los modelos propuestos buscarán distinguir cuándo las características de red obedecen a un comportamiento usual del servidor y cuándo a un comportamiento sospechoso de incidencia. El objetivo sería construir un modelo que, integrándose en herramientas para la gestión de red, identifiquen incidencias de manera autónoma y posibles causas de dichas incidencias, facilitando así el manejo de grandes volúmenes de servidores.

El trabajo abarcará desde el diseño e implementación de los modelos, hasta la evaluación de los mismos. Para ello, se dispone de datos reales extraídos de un centro de servidores de una gran compañía logística de hidrocarburos, esto nos permitirá determinar su eficacia y viabilidad en un entorno real.

Palabras clave

Centro de datos, características de red, LSTM, métodos auto-regresivos

Abstract

Network monitoring is an essential component to guarantee availability and high performance in data centers. However, as computer networks enlarge and turn more complicated, it becomes infeasible to monitor them through traditional methods, such as fixed thresholds and human inspection of the network. This project studies the possibility of a system that detects and notifies autonomously of abnormal functioning of a server.

The network data collected in the communication between a server and its clients, contains vast information about the functioning of the server. Using this data, the project proposes two models: the first one, based on LSTM (Long short-term memory) neural networks; the second one, based on traditional autoregressive methods. The proposed models will try to distinguish between normal functioning of the server and suspicious network data. The main objective would be to integrate such models in tools for network monitoring, detecting abnormal behaviour automatically and, therefore, simplifying the management of large volume data centers.

This project will start by defining rigorously the problem, proposing and implementing the models and, finally, evaluating the viability of such models. To accomplish this goal, a dataset collected from a big oil company has been provided. This will enable us to simulate and validate our system in a real-life enviroment.

Keywords

network monitoring, datacenter, dataset, LSTM, autoressive models

Índice

Listas

Lista de ecuaciones

Lista de figuras

Lista de tablas

1

Introducción

La gestión de red consiste en analizar datos recogidos en la comunicación entre un conjunto de servidores y sus clientes. Los datos recolectados contienen información sobre el funcionamiento del propio servidor. Por ejemplo, una subida repentina del número de clientes puede indicar un ataque de denegación de servicio, o la caída del número de bits enviados puede estar señalando el mal funcionamiento de un servidor. La gestión de red en centros de datos se ocupa de identificar estas incidencias, notificarlas y subsanarlas, garantizando un alto rendimiento y disponibilidad de los servicios.

El volumen de servidores gestionados en centros de datos aumenta cada vez más. Por un lado, esto se debe a la digitalización de los sistemas. Por otro, ha habido un creciente interés en la migración de servicios a la nube. Por ello, las redes a gestionar se tornan cada vez más complejas y los métodos tradicionales de inspección manual resultan inviables.

En este trabajo se implementan y evalúan modelos que detectan y notifican incidencias en redes de ordenadores de manera autónoma. El sistema propuesto será alimentado con las series temporales que forman las características de red de un servidor. Luego, utilizando esta información, buscará distinguir el comportamiento de red 'usual' del servidor, del comportamiento 'anómalo'. Remitiéndonos al ejemplo anterior, si el sistema detectase una repentina subida del número de flujos, notificaría a un gestor para investigar la posible incidencia. De este modo, liberamos al gestor de la tarea de identificar incidencias, facilitando la gestión de un mayor volumen de servidores.

En este trabajo se proponen y comparan dos modelos para clasificar incidencias. Ambos modelos se basan en el uso de estimadores de series temporales, el primero utiliza redes neuronales LSTM para esta tarea, mientras que el segundo utiliza métodos auto regresivos. Desarrollaremos más adelante el funcionamiento de estos sistemas.

Adicionalmente, se dispone para este trabajo de un conjunto de datos recogido en un centro de servidores de una gran compañía logística de hidrocarburos. Esto permitirá que evaluemos los modelos propuestos en un entorno lo más parecido a circunstancias reales.

En cuanto a la bibliografía consultada, las propuestas de modelos para clasificar incidencias en redes de ordenadores fueron extraídas del artículo [1]. Por su parte, el funcionamiento general de

las redes LSTM fue consultada en [2], y su aplicación en la clasificación de series temporales fue consultada en [3]. Además, algunas ideas referentes a la detección de incidencias fueron consultadas en [4]. Por último, el trabajo fue realizado en colaboración con la empresa Naudit que aportó la guía de expertos en materia de redes de comunicación.

1.1. Objetivos

En este trabajo se diseña, implementa y evalúa un sistema capaz de detectar incidencias en redes de ordenadores de manera autónoma. El objetivo final es desarrollar un modelo que, utilizándose en el ámbito de la gestión de red, facilite el manejo de un gran número de servidores.

1.2. Colaboración con Naudit HPCN

El proyecto presentado en este documento se enmarca en un convenio de colaboración entre la Universidad Autónoma de Madrid y la empresa de base tecnológica Naudit HPCN. La investigación y desarrollo realizados en este trabajo servirán de base para la implementación real en clientes de Naudit HPCN.

1.3. Estructura del documento

El documento se compone de tres capítulos principales y las conclusiones. El capítulo 2 introduce algunas de las tecnologías que servirán como punto de partida para el desarrollo de este proyecto. Posteriormente, el capítulo 3 define con mayor rigor las condiciones de funcionamiento de los modelos, discute las especificaciones y su implementación. Ya explicado el sistema, en el capítulo 4 se realizan varios experimentos para evaluar y comparar el funcionamiento de los modelos. Por último, en las conclusiones discutimos brevemente la viabilidad del modelo implementado y posibles ámbitos de desarrollo futuro.

2

Tecnologías utilizadas

En este capítulo daremos una idea general de algunas de las tecnologías que se utilizarán como punto de partida para desarrollar los modelos de clasificación de incidencias. Si bien el estudio minucioso de estas tecnologías esta fuera del alcance del trabajo, en este capítulo las introducimos y esbozamos su funcionamiento.

En concreto, introducimos dos estimadores de series temporales que serán necesarios para definir el modelo de clasificación. El primero utiliza redes recurrentes LSTM (*long short-term memory*) para la regresión de series y el segundo se basa en métodos auto-regresivos (VAR).

2.1. Redes LSTM

Las redes neuronales LSTM son redes recurrentes que permiten que información de pasos anteriores persista y forme parte de la predicción de nuevas entradas. Gracias a la propiedad de identificar interdependencias en las entradas, las redes recurrentes se han aplicado con éxito en la predicción de series temporales, el reconocimiento de voz, traducción y modelado de lenguajes, ámbitos donde la información de contexto es necesaria para lograr el resultado esperado.

2.1.1. Funcionamiento general

En la figura 2.1 se ilustra la estructura de las celdas recurrentes de una red LSTM. Los vectores X_t, C_t y h_t, representan respectivamente la entrada de a la red, su estado y su salida en el instante t.

El estado C_t de la red se encarga de almacenar la información de contexto que se utilizará en futuras predicciones. Por su parte, la red se compone de tres 'puertas':

1.– La *forget gate* se encarga de determinar qué información del estado persistirá al siguiente estado.

2.– La *input gate* se encarga de actualizar el estado de la red para incluir nueva información respecto a la entrada actual.

3.– La *output gate* determina la salida de la red en función de los valores del estado (C_t), la entrada (X_t) y la salida del paso anterior (h_{t-1}).

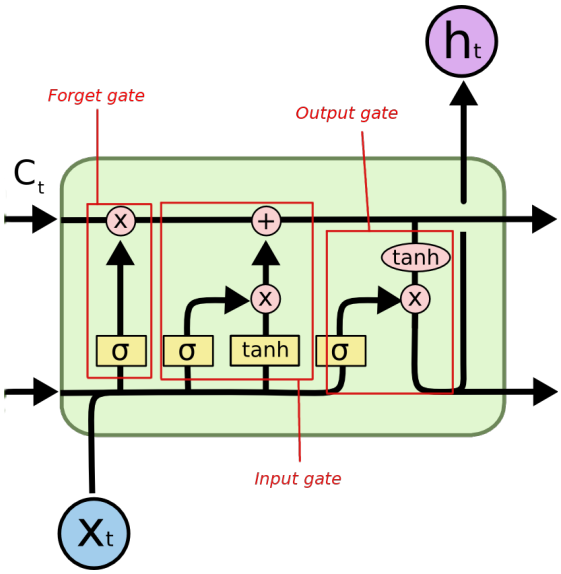


Figura 2.1: Red LSTM.

Cada una de las 'puertas' de la red funciona como una red *feed-forward* usual. Mientras que los círculos rojos representan las operaciones de multiplicación y suma bit a bit.

Utilizando el aprendizaje de descenso por gradiente en redes recurrentes (*Backpropagation through time*), cada puerta aprende a determinar su salida para optimizar la predicción, esto es: la *forget gate* aprenderá qué es relevante que persista en el estado de la red en función de X_t y h_{t-1}; la *input gate* aprenderá qué información debe incluirse en el estado en función de los mismos parámetros; y, por último, la *output gate* aprenderá cómo componer la salida de la red en función de la entrada, el estado y la salida anterior. Finalmente, siguiendo la estructura cíclica de las redes recurrentes, la salida y su estado serán alimentados a la red junto a la próxima entrada.

2.1.2. Estimación de características en redes de ordenadores

Las redes LSTM son eficaces estimadores de series temporales, resultando incluso robustas frente a datos ruidosos, con dependencias a largo plazo y con rasgos poco relevantes o aleatorios ([2]). Aprovechando estas propiedades, en este trabajo utilizaremos las LSTM para estimar la serie de características de red de un servidor.

Las características de red (normalizadas) forman un vector de 14 dimensiones que se recoge en intervalos de 5 minutos. Para predecir el vector de valores de red, se utilizará una LSTM que recibe como entrada una ventana de una hora de vectores previos.

Ejemplo de predicción

Partiendo del conjunto de datos que se dispone para este trabajo seleccionamos un servidor concreto. Utilizamos el 70 % de los datos para entrenar la red y el 30 % restante se usa para evaluar la predicción. En la figura 2.2 se ilustran los resultados de la predicción proyectados a una única dimensión. En este caso, se muestran los bits por segundo medios recibidos por el servidor. Cada unidad del eje x representa un intervalo de 5 minutos.

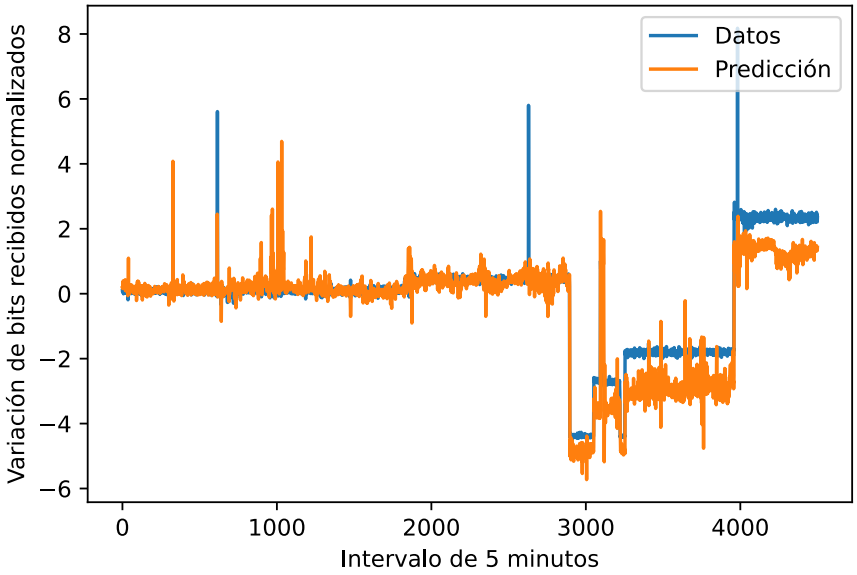


Figura 2.2: Ejemplo de predicción de serie temporal.

Observación: Nótese que los valores que se utilizan están normalizados, hablaremos más al respecto en la sección 3.1.2.

En el siguiente capítulo detallaremos como la estimación de esta serie temporal se enmarca en la clasificación de incidencias en redes de ordenadores.

2.2. Modelo vectorial auto regresivo

El modelo vectorial auto regresivo (VAR, por sus siglas en inglés) es un modelo de estimación de series temporales estacionarias. VAR se utiliza para describir series temporales cuyas variables interactúan linealmente entre sí. En el marco de este trabajo, utilizaremos el modelo VAR para estimar el valor de características de red en centros de datos.

2.2.1. Funcionamiento general

Consideremos una matriz de dimensiones $K \times T$ que representa una serie temporal multidimensional, donde K es el número de variables y T el número de observaciones. El modelo VAR estima el valor de la serie temporal en el instante t, como una función lineal de los anteriores n valores más un error u_t:

$$Y_t = \nu + A_1 Y_{t-1} + \ldots + A_n Y_{t-n} + u_t$$

En la ecuación se tiene que: ν es un vector de R^k, A_i es una matriz constante $K \times K$ y u_t, el error de la estimación, es una variable aleatoria con distribución Normal$(0, \Sigma_u)$.

El elemento $(A_s)_{(i,j)}$ determina el efecto en la coordenada i-ésima de un cambio unitario en la coordenada j-ésima retardada s intervalos, esto es, el efecto de $(Y_t)_j$ sobre $(Y_{t+s})_i$. De este mismo modo, la columna i-ésima de la matriz A_s determina el efecto de Y_t sobre $(Y_{t+s})_i$. El error u_t corresponde con la variación del vector Y_t que no puede ser estimada a partir de sus n observaciones anteriores.

Para estimar las variables A_i y ν del modelo, se minimiza el error cuadrático medio de las observaciones que se dispongan. Supongamos que se dispone de T observaciones de la serie temporal, entonces se considera el error:

$$\text{Error} = \sum_{i=n}^{T} (Y_i - \hat{Y}_i)^2$$

Donde los estimadores $\hat{Y}_i$ se determinan con las variables $A_1, \ldots, A_n$ y ν. De este modo, el Error es función precisamente de dichas variables. Se utilizan entonces como estimadores del modelo VAR aquellos valores de $A_1, \ldots, A_n$ y ν que minimizan la función Error.

2.2.2. Hipótesis, diferenciación de datos y ámbito de aplicación

Para que los estimadores por mínimos cuadrados del modelo VAR resulten en un modelo *consistente* [1] se necesita que la serie temporal sea estacionaria y que la variable de error u_t sea independiente de las variables explicativas.

En este proyecto utilizaremos el modelo VAR para predecir series temporales de características de red. Los valores de estas series no suelen tener comportamientos estacionarios. Por ello, en vez de predecir directamente los valores de la serie, estimaremos la variación de estos valores. Podemos observar en la figura 2.3 que la variación de los bits por segundos enviados presentan un comportamiento estacionario, a pesar de que la serie de los bits no lo presente.

En la figura 2.4 observamos un ejemplo de predicción de serie temporal proyectado a una única

[1] Que el estimador tienda en probabilidad al valor real de la serie.

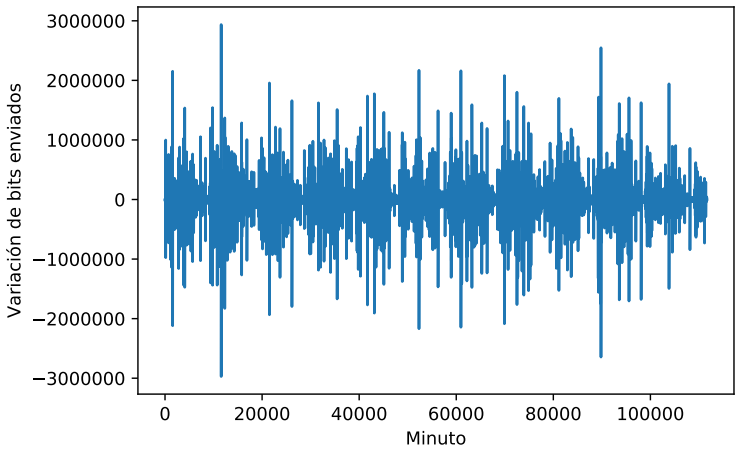


Figura 2.3: Ejemplo de estacionaridad en variación de datos

variable. En este caso, se ilustra la predicción de la variación de los bits recibidos. Recordamos que, al igual que para el regresor LSTM, los datos serán normalizados antes de introducirlos en el modelo (y antes de calcular la variación).

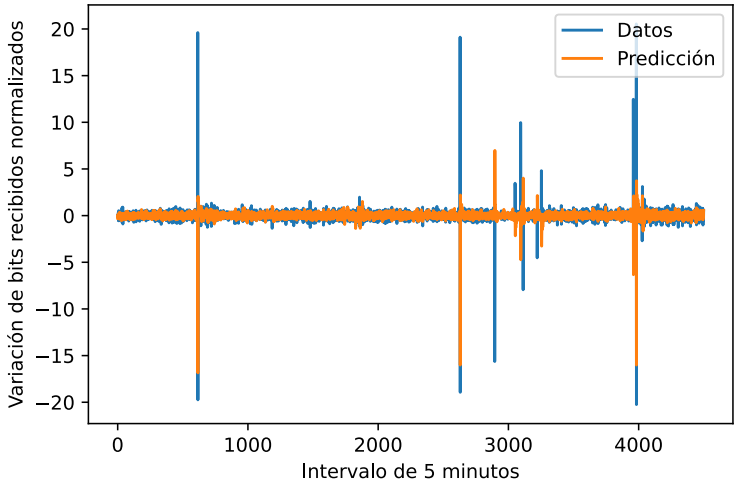


Figura 2.4: Predicción de la variación de bits recibidos con VAR

3

Diseño e implementación

En este capítulo detallaremos con más rigor la arquitectura que se estudia y los datos que se disponen para la detección de incidencias. Luego, utilizando los regresores del capítulo 2, definiremos los modelos de clasificación de incidencias y expondremos la información que aportan acerca del centro de datos. Por último, hablaremos sobre la generación de incidencias artificiales, herramienta que nos será útil para evaluar los sistemas en el siguiente capítulo.

3.1. Especificación del problema

3.1.1. Sobre la arquitectura

En primer lugar, para proceder a analizar datos de red, requerimos de una infraestructura que recolecte estos datos. En la figura 3.1 se muestra la arquitectura que es objeto de estudio en este trabajo. Observamos que entre el centro de datos y la red, hay una sonda que tiene acceso a los datos de la comunicación. Esta sonda se encarga de recoger las características de red que serán luego analizadas por un gestor de red o, en el marco de este trabajo, analizadas por un sistema.

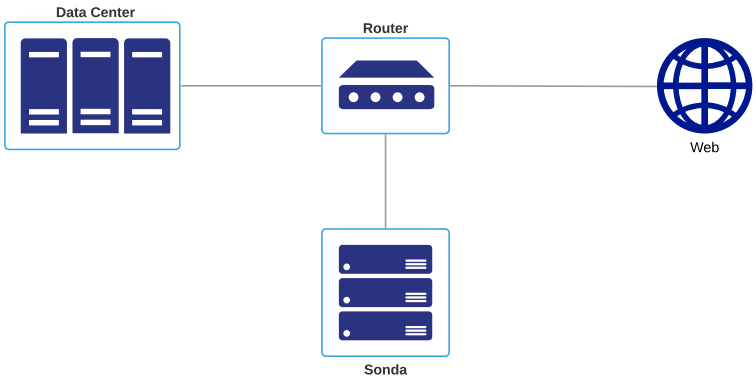


Figura 3.1: Arquitectura de centro de datos

La sonda que se observa en la figura recopilará características de red del centro de servidores.

Los datos se almacenarán agregados en intervalos de 5 minutos y a nivel de servidor. Esto es, cada 5 minutos se recogerá un vector de métricas que resume la información de red de cada servidor. Estos vectores conformarán las series temporales que luego estudiaremos para detectar incidencias. Recalcamos que los datos son agregados en la sonda, por lo que no se dispone de una información completa de la comunicación de red. Aunque sería tentador trabajar con otro nivel de agregación (a nivel de flujo, por ejemplo), en infraestructuras reales encontramos que la agregación se hace a nivel de servidor por limitaciones de almacenamiento [1].

3.1.2. Datos del problema

Como hemos mencionado, para este trabajo se dispone de un conjunto de datos recogido en un centro de servidores de una empresa logística de hidrocarburos. Los datos cubren un intervalo de tres meses y contienen características de red de unos 300 servidores. A partir de ahora, haremos referencia directa a este conjunto para los ejemplos que se propongan. Sin embargo, quedará patente que todo el proyecto podría replicarse en cualquier centro de datos con una arquitectura similar a la mostrada en la sección anterior.

En la tabla 3.1 mostramos un ejemplo de registros de red de un servidor real.

Fecha	bps enviados	bps recibidos	...	RTT medio
08-07-2019 19:35	49918	256093		0.0271
08-07-2019 19:40	588324	80718		0.0003
...			...	
25-09-2019 01:50	31479	89977		0.0300

Tabla 3.1: Registros de características de red de un servidor

Cada fila de la tabla se compone de un vector de métricas que resume la comunicación de red del servidor en un intervalo de 5 minutos. De este modo, la tabla completa conforma una serie temporal multidimensional. Notemos que cada servidor tendrá su propia tabla y, con ello, su propia serie temporal. Además, el centro de datos albergará servidores que ofrecen servicios distintos, por lo que el comportamiento 'usual' de cada servidor puede diferir.

En cada intervalo de 5 minutos se recogen 14 métricas sobre la comunicación de red. Las métricas son seleccionadas atendiendo a aquellas que suelen ser más informativas durante la gestión. En la tabla 3.2 recogemos cada una de las características de red recopiladas en la sonda.

Ejemplo de anomalía de red

Veamos un ejemplo de cómo las características de red pueden revelar información sobre los servidores

[1] Este aspecto consta en el *dataset* dispuesto por la compañía colaboradora.

Bps enviados medios	Bps recibidos medios
RTT medio por conexión	Paquetes Ack duplicados
Paquetes sin respuesta del cliente	Paquetes sin respuesta del servidor
Número de conexiones	Ventanas cero
Paquetes por segundo enviados medios	Paquetes por segundo recibidos medios
Paquetes reset del servidor	Paquetes reset del cliente
Número de retransmisiones	Paquetes SYN

Tabla 3.2: Características de red almacenadas

monitorizados:

En la figura 3.2 se muestra la serie temporal de los bps enviados medios de un servidor. Fijémonos que en el recuadro rojo se enmarca una repentina subida de los bits enviados. Este tipo de comportamientos indican la posible existencia de una incidencia. Conformando esto un claro ejemplo de comportamiento anómalo, un modelo funcional debería identificar el momento en que se inicia el pico en los bps y notificar de ello a un gestor de red para que lo inspeccione.

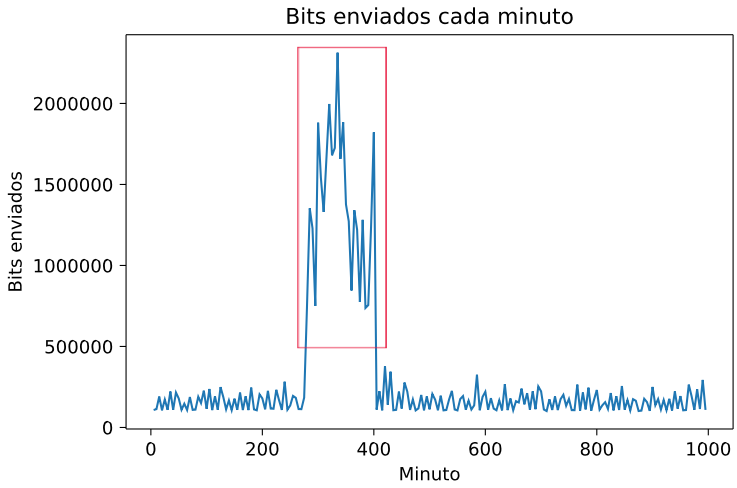


Figura 3.2: Ejemplo de comportamiento anómalo en un servidor.

Este ejemplo ilustra el funcionamiento del sistema en circunstancias exageradas. Sin embargo, se pretende que los modelos propuestos sean capaces de identificar fluctuaciones más sutiles (pero anómalas) en las características de red.

Selección de servidores

Los datos dispuestos cuentan con alrededor de 300 servidores. En este trabajo nos limitamos a presentar los resultados con una selección de 10 de ellos. Por un lado, esto facilita la legibilidad de los resultados. Por otro, los servidores se seleccionan considerando aquellos con mayor nivel de

agregación (mayor número de clientes o conexiones).

Por lo general, los servidores con mayor agregación tienen un comportamiento más estable, con lo que los modelos tendrán un mejor desempeño. Por el contrario, los servidores con pocos usuarios suelen tener series temporales volátiles y difíciles de predecir, por lo que los modelos propuestos tendrían un peor rendimiento. Por tanto, la selección centra el estudio en aquellos servidores cuya naturaleza predecible facilita la automatización de la gestión de red.

Elección de servidores
Siguiendo esa línea de razonamiento, los servicios se seleccionaron buscando servidores con comportamientos estables. De los 300 servidores se eligieron los 10 con mayor agregación atendiendo al siguiente algoritmo:

1.– Primero, se seleccionan del total de servidores la mitad que tuviese mayor número medio de conexiones abiertas. De los obtenidos, se restringe a los 10 servidores que tuviesen una menor varianza del número de conexiones.

2.– En segundo lugar, se utiliza el mismo procedimiento que en el paso anterior para seleccionar otros 10 servidores, en esta ocasión, considerando el número de IPs de clientes [2].

3.– Por último, de los 20 servidores resultantes, se eligen los 10 que tuviesen la menor variación del ancho de banda.

Dado este método de selección, uno podría preguntarse por qué los servidores no se seleccionaron basándose directamente en la variación del ancho de banda. Aunque se podría haber hecho, quizás supondría una sobre simplificación. El método propuesto pretende que la estabilidad de los servidores venga inducida por la agregación de muchos datos, en contraposición a que el servidor específico ofrezca servicios que induzcan comportamientos de red estable.

Normalización de datos

En el capítulo anterior introdujimos los regresores basados en LSTM y VAR. Recordamos que ambos modelos requieren que los datos manejados estén normalizados para dar resultados satisfactorios.

Antes de alimentar los valores en nuestros clasificadores de incidencias, las características de red serán normalizadas. La normalización se realiza de manera estándar, a cada valor de red le restamos la media de la característica en cuestión y lo dividimos entre su desviación estándar. Para todos los experimentos, los datos se dividen en un 70 % de entrenamiento y un 30 % de validación. De modo que, la normalización de los datos de entrenamiento se realiza con sus propias estadísticas, mientras que la normalización de los datos de validación se realiza con las estadísticas de los datos de entrenamiento.

A partir de ahora, aunque no hagamos referencia explícita a ello, asumiremos que los datos que manejan los modelos son valores normalizados.

[2] Nótese que el número de conexiones y el de IPs de clientes, son valores distintos. Un mismo servidor puede tener varias conexiones abiertas con un solo cliente

Sobre las incidencias

En un contexto real, cuando un gestor de red se encuentra con una incidencia, la identifica y la subsana. Sin embargo, no se lleva un registro riguroso y sistematizado de las incidencias que han ocurrido en el pasado. Por tanto, el *dataset* que se dispone no está etiquetado, esto es, para cada registro no sabemos si el registro obedece a un comportamiento normal de la red o a una anomalía. Esto nos obliga a proponer modelos de entrenamiento no supervisado, como veremos en la siguiente sección.

Es importante notar que, aunque los registros no estén etiquetados, los centros de datos suelen funcionar la mayor parte del tiempo bajo normalidad. Aprovecharemos este hecho para evaluar los modelos. Asumiremos que todos los datos que se disponen corresponden a un comportamiento normal y, como mostraremos en la sección 3.3, induciremos sobre estos datos incidencias de manera artificial. Luego, con los datos modificados, buscaremos evaluar la eficacia de nuestros modelos en identificar las incidencias fabricadas.

Sobre la evaluación de modelos

Recordamos en esta sección algunas métricas ampliamente utilizadas en el contexto de clasificación y detección de incidencias.

1.– La sensibilidad: Hace referencia al porcentaje de verdaderos positivos frente al total de positivos del problema. En el caso concreto de estudio:

$$\text{sensibilidad} = \frac{\text{incidencias detectadas correctamente}}{\text{total de incidencias reales}}$$

2.– La precisión: Hace referencia al porcentaje de verdaderos positivos frente al total de positivos dados por el clasificador. En nuestro caso:

$$\text{precisión} = \frac{\text{incidencias detectadas correctamente}}{\text{incidencias detectadas}}$$

3.– *F-score*: Es una métrica que aglutina la información de la sensibilidad y la precisión. Se calcula como la media armónica entre la sensibilidad (s) y la precisión (p):

$$F = \frac{s * p}{s + p}$$

Alternativamente, se puede considerar el F_β-*score*:

$$F_\beta = \frac{(1 + \beta^2) s * p}{s + p * \beta^2}$$

El parámetro β es un factor de relevancia que indica cuántas veces se considera más importante la sensibilidad que la precisión. Para el experimento en la sección 4.1 utilizaremos el F_2-*score* como métrica para evaluar las clasificaciones, aunque realmente se puede establecer una correspondencia con otras métricas.

Por último, en cuanto a la evaluación de los modelos, cabe mencionar que las incidencias en redes de ordenadores se suelen dar por ráfagas. En la figura 3.2 vemos este fenómeno, una incidencia ocurre

en un momento dado y se extiende por un intervalo de tiempo. A efectos de la gestión de red en un contexto real, sólo es relevante que se identifique el momento en que se inicia la incidencia. Por ello, cuando evaluemos los modelos, daremos un intervalo por bien clasificado a partir del primer momento en que se detecte la inci dhencia, sin tener en cuenta las clasificaciones del resto del intervalo.

3.2. Modelos propuestos

En términos generales, el funcionamiento del clasificador de incidencias es el siguiente:

Se dispone de un regresor de series temporales que, utilizando valores anteriores, es capaz de predecir el vector de red actual que se esperaría bajo circunstancias 'normales' del propio servidor. A su vez, en ese instante recibimos un vector con las métricas de red que, en efecto, se observan. Finalmente, si tenemos que el vector de métricas observado se aleja demasiado del vector de métricas esperado, entonces el modelo notificará que el comportamiento de red del servidor es sospechoso de incidencia.

En esta sección detallaremos con mayor rigor los dos modelos propuestos para clasificar las incidencias. La diferencia entre los modelos radica únicamente en la regresión de la serie temporal: uno utilizará redes LSTM para la tarea y el otro usará VAR. Veremos que el regresor capaz de extrapolar el comportamiento 'usual' del servidor, resultará en un mejor clasificador.

3.2.1. Definición del modelo

A la serie temporal que define las características de red la denotaremos por $\{X_k\}_{k=1}^{\infty}$, donde

$$X_k = (X_k^1, ..., X_k^{14})$$

es el vector de características de red en el intervalo de tiempo k-ésimo.

Supongamos ahora que estamos en el intervalo i-ésimo de la red y queremos comprobar el estado del servidor en cuestión:

El primer paso será utilizar el regresor de series temporales para determinar el vector de red esperado, a este vector lo denotaremos por $\hat{X}_i$. En la figura 3.3 se indica cómo se utilizaría una red LSTM para determinar el valor esperado actual en función de una ventana de una hora de valores anteriores (recordamos que cada intervalo son 5 minutos, con lo que 18 intervalos son una hora).

Debido a que el regresor es entrenado principalmente con datos libres de incidencia, el vector de red estimado $\hat{X}_i$ se asemejará al comportamiento del servidor en circunstancias normales. De este modo, si la observación real se aleja de lo predicho, entonces ese comportamiento sería sospechoso de incidencia. Para calcular la distancia entre X_i y $\hat{X}_i$, utilizamos el error cuadrático medio y lo

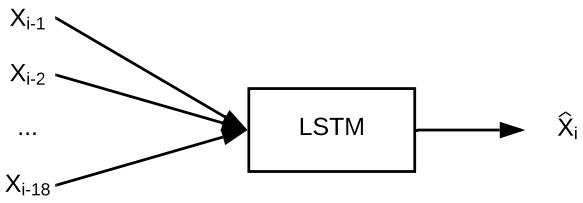


Figura 3.3: Esquema de entradas y salidas de una LSTM (análogo para VAR).

comparamos con un umbral precomputado:

$$\frac{1}{n}||\hat{X}_i - X_i||^2 > \mu \tag{3.1}$$

De satisfacerse la inecuación 3.1, diremos que el comportamiento del servidor en el intervalo i-ésimo es sospechoso de incidencia. Por lo que el sistema dispararía una alarma para notificar la incidencia al gestor de red.

La figura 3.4 ilustra cómo funcionaría el clasificador si el vector de características de red fuese unidimensional.

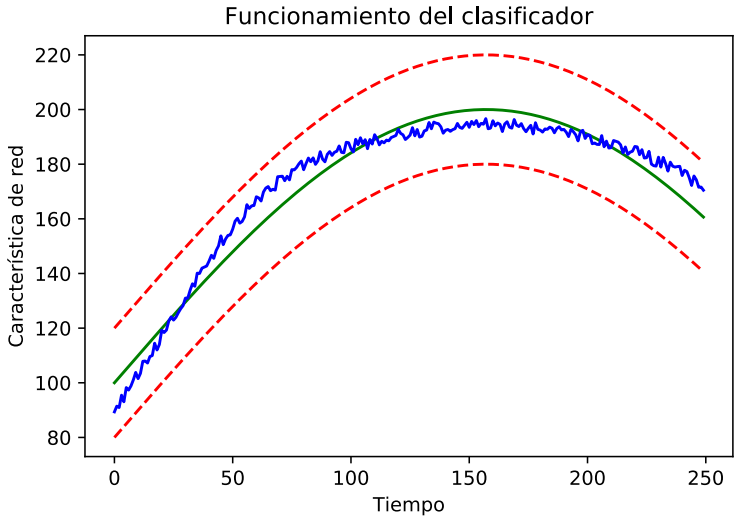


Figura 3.4: Ejemplo de funcionamiento del clasificador: en azul la serie observada, en verde la predicción y con líneas discontinuas rojas los límites fuera de los cuales el comportamiento se considera anómalo.

Observación: Recordamos que la única diferencia entre los dos modelos es el algoritmo para estimar la serie temporal. De la formulación dada se sigue que, mientras el algoritmo reproduzca más fielmente

la serie temporal sin incidencias, el modelo será más certero en la clasificación.

3.2.2. Sobre el umbral

Definido el modelo, fijémonos que ambos dependen: por una parte, de su capacidad para predecir acertadamente la serie temporal; por otra, de un umbral μ que determinará la clasificación. El umbral determina los límites de lo que se considera, o no, una incidencia (véase 3.1).

El umbral es un parámetro del modelo que establece un compromiso entre la sensibilidad y precisión: un umbral alto mejorará la precisión del modelo en detrimento de su sensibilidad (notifica solo las alarmas más extremas); por el contrario, un umbral bajo aumentará la sensibilidad del modelo, disminuyendo su precisión (detecta incidencias más sutiles). Por lo general, el umbral será un parámetro que fijará el gestor de acuerdo a las necesidades que se tenga con respecto a la sensibilidad y precisión del sistema.

En cuanto a las unidades del umbral, si asumimos que las variables presentan una distribución constante, de la ecuación 3.1 deducimos que el umbral equivale al número de desviaciones típicas al cuadrado que se desvía la predicción del vector observado. Por ejemplo, un umbral de 25 nos indica una separación de 5 desviaciones típicas entre el vector observado y el esperado. Aunque la hipótesis de la distribución es fuerte, esto nos da una idea de las unidades a esperar en los valores umbrales.

Más adelante, en la sección 4.1, damos un método para determinar el umbral. En ese experimento se establece, para un servidor dado, una correspondencia aproximada entre umbrales y *F-scores*. Luego, el gestor de red puede seleccionarlo de acuerdo a sus exigencias.

3.2.3. Información de la clasificación

Ya vimos cómo el modelo propuesto distingue si un registro de red X_i es o no una incidencia. En esta sección discutimos qué otra información acerca del registro podemos extraer del modelo:

Características más relevantes

Una vez hemos clasificado un registro X_i como incidencia, podríamos preguntarnos por qué el modelo considera que este vector corresponde a un comportamiento anómalo. Esa información podría ser crucial para que un gestor de red pudiese subsanar la incidencia. Veamos cómo el modelo propuesto nos permite establecer qué características de red fueron más relevantes en la clasificación.

Consideremos el módulo

$$||\hat{X}_i - X_i||^2$$

Este valor determina la clasificación. Si el valor supera el umbral, nos gustaría saber qué peso tuvo cada característica de red en la clasificación. Para esto, basta con considerar el porcentaje que aporta cada componente del vector $(\hat{X}_i - X_i)$ a su módulo. La aportación de la coordenada j-ésima sería:

$$r_i^j = \frac{(\hat{X}_i^j - X_i^j)^2}{||\hat{X}_i - X_i||^2}$$

El factor r_i^j nos permite ordenar las características de red según cuánto se han desviado de su valor esperado. Con ello, tenemos que aquellas características con mayor factor r tendrán comportamientos más inusuales y, probablemente, sean más descriptivos en cuanto a la naturaleza de la incidencia.

De este modo, el modelo no solo nos permitirá indicar si un servidor presenta un comportamiento anómalo, sino que también podrá guiar al gestor de red hacia aquellas características que sean más relevantes para desvelar las causas de la incidencia. De esta forma, el modelo ahorra tiempo tanto en la identificación de incidencias, como en la investigación necesaria para subsanarla.

Prioridad entre incidencias

En esta sección veremos que el modelo nos permite definir un orden de prioridad entre las incidencias detectadas.

Un centro de datos puede componerse de miles de servidores que son monitorizados por un único gestor de red, por tanto, establecer prioridades entre las incidencias permite al gestor centrarse en aquellas alarmas que sean más urgentes o presenten comportamiento más inusuales.

Definimos la prioridad de una incidencia como el valor p:

$$p = \frac{\frac{1}{n}||\hat{X}_i - X_i||^2}{\mu}$$

El factor nos indica 'cuántos umbrales' el valor observado se ha desviado del valor esperado. Claramente, mientras mayor sea el factor p, más radical es el comportamiento de la incidencia, y más relevante la alarma.

En definitiva, los factores p y r permiten a un gestor de red identificar inmediatamente qué servidores están siendo más afectados y en qué características de red, facilitando así el manejo de grandes centros de datos.

3.3. Etiquetado de incidencias

Como ya mencionamos antes, uno de los inconvenientes en los registros de red es que no están etiquetados, con lo que no sabemos cuando un vector X_i corresponde con una incidencia. Aunque el sistema propuesto no requiera de este etiquetado para la clasificación, sería conveniente tenerlo para

poder evaluar la eficacia del modelo antes de implantarlo en un entorno real.

En esta sección exponemos una idea extraída del artículo [1], que resuelve esta problemática induciendo artificialmente las incidencias en los datos.

3.3.1. Generando incidencias

Como primera simplificación, asumiremos que todos los registros de red iniciales están libres de incidencias. Aunque puede parecer una hipótesis fuerte, realmente los centros de datos suelen funcionar con normalidad la mayor parte del tiempo, por lo que si asumimos que no hay incidencias, estaremos acertando en la amplia mayoría de los casos.

Partiendo de esto, diseñaremos incidencias que luego induciremos en los datos del servidor. Las incidencias se diseñan atendiendo a tres parámetros:

1.– El intervalo de tiempo en el que aplica.

2.– El porcentaje de clientes del servidor a los que afecta la incidencia.

3.– Un factor multiplicativo que indica la intensidad de la incidencia en el porcentaje de clientes afectados.

Las incidencias inducidas se pueden solapar entre sí, afectando de este modo a varias características de red a la vez. Combinando distintas ternas de parámetros, se pueden generar incidencias que repliquen a grosso modo circunstancias reales. Por ejemplo, para simular un ataque de denegación de servicio, bastaría con disparar el número de conexiones y bits recibidos en un intervalo y servidor concreto. En el apéndice A discutimos algunos paquetes de incidencias que se utilizarán más adelante para simular distintos tipos de anomalías de red.

Utilizando el generador de incidencias convertimos la evaluación de los modelos en un problema supervisado. Por un lado, todo registro que no ha sido alterado se cataloga como libre de incidencias. Mientras que aquellos en los que se indujo una variación se les clasifica como sospechosos. La clasificación que induce este generador nos permitirá evaluar, en el siguiente capítulo, el desempeño de nuestros modelos en detectar incidencias.

Un ejemplo del generador de incidencias

En la figura 3.5 tenemos un ejemplo de incidencia generada sobre datos reales. En azul tenemos los datos originales, en rojo tenemos el intervalo sobre el que fue inducida la incidencia.

Un modelo eficaz debería detectar el intervalo en rojo y notificar al gestor.

Nota: En este ejemplo ilustramos la incidencia sobre una única característica. Sin embargo, en los experimentos del siguiente capítulo consideraremos incidencias que alteren varias características simultáneamente.

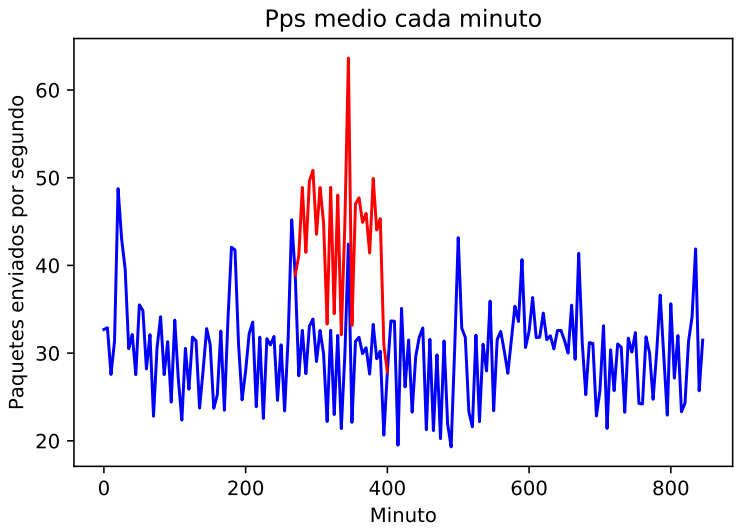


Figura 3.5: En azul la serie temporal original. En rojo el intervalo donde se induce una incidencia.

4

Experimentos

Ya propuesto el modelo de clasificación de incidencias en el capítulo anterior, procedemos a someterlo a diferentes experimentos para probar su eficacia y viabilidad. Realizaremos pruebas tanto para el modelo basado en redes LSTM, como para el modelo auto-regresivo, con la finalidad de comparar los resultados.

Como primera prueba, induciremos incidencias artificiales en el *dataset* de características de red (utilizaremos el generador mencionado de la sección 3.3) y comprobaremos si el modelo es capaz de detectar las anomalías fabricadas. Además, con este experimento expondremos un método para determinar el parámetro 'umbral' del modelo.

En segundo lugar, evaluaremos los modelos en un contexto real. Simularemos lo que hubiese ocurrido si hubiésemos desplegado el modelo durante 15 días en el centro de datos estudiado en el *dataset*. El objetivo es evaluar el número de alarmas disparadas, con el fin de estimar la carga que el sistema trasladaría a un gestor. Adicionalmente, inspeccionaremos manualmente algunas de las alarmas disparadas para comprobar si el sistema detecta en los datos comportamiento que, en efecto, son anómalos.

Por último, haremos un breve estudio sobre la capacidad de cómputo requerida por el modelo.

4.1. Evaluando los modelos con incidencias artificiales

En esta sección estudiamos cómo se desenvuelven los modelos identificando incidencias inducidas artificialmente en los datos.

Tras alterar los registros de red, utilizaremos el modelo para clasificar los intervalos de la serie temporal. Luego, usando la clasificación inducida por el generador de incidencias, calcularemos el F_2-*score* del clasificador para un rango de umbrales, y determinaremos si se obtienen clasificaciones satisfactorias para alguno de los umbrales probados. Esta prueba no solo nos permitirá determinar si los modelos son capaces de predecir eficazmente las incidencias artificiales, sino que también nos ayudará determinar el umbral óptimo a utilizar en cada servidor.

Durante el experimento pondremos a prueba el modelo induciendo distintos tipos de incidencias, con esto nos aseguramos de que los modelos son capaces de detectar comportamientos anómalos independientemente de la naturaleza de la incidencia. Las incidencias fueron diseñadas de modo que reproduzcan en la medida de lo posible circunstancias reales. Para más detalle sobre los paquetes de incidencias consulte el apéndice A.

Nota: En este experimento el 70 % de los datos iniciales fue utilizado para entrenar el modelo, fue en el 30 % restante en el que se indujeron las incidencias y sobre el que se realizó la clasificación.

4.1.1. Prueba con un único servidor

Como primera aproximación, presentamos los resultados de este experimento en un único servidor. El servidor fue elegido al azar entre los 10 servidores seleccionados en la sección 3.1.2. El experimento consistió en:

1.– Seleccionar el 30 % final de los datos (recordamos que ya están normalizados) e inducir en ellos un paquete de incidencias.

2.– Sobre estos datos ejecutamos ambos modelos de clasificación de incidencias.

3.– Para valores umbrales del rango $[1, 50]$, calculamos el F_2-*score* comparando la clasificación obtenida por el clasificador con la clasificación real inducida por el generador de incidencias.

4.– Por último, seleccionamos el umbral con el mejor F_2-*score* y analizamos su idoneidad.

El mismo procedimiento se repite para cada uno de los 5 paquetes de incidencias detallados en el apéndice A.

Resultados con un único servidor

Utilizamos el clasificador basado en redes LSTM para ejecutar el experimento con cada paquete de incidencias. Los resultados los recogemos en la tabla 4.1. Notemos que todos los paquetes de incidencias obtienen un F_2-*scores* notablemente alto. Podemos concluir que, para este servidor concreto, el clasificador basado en redes LSTM clasifica satisfactoriamente.

Paquete de incidencias	Umbral	F_2	Precisión	Sensibilidad
1	25	0.832	92.54 %	81.15 %
2	9	0.9331	88.13 %	94.7 %
3	11	0.8767	59.31 %	99.58 %
4	28	0.9524	80.0 %	100.0 %
5	30	0.8532	90.07 %	84.21 %

Tabla 4.1: Información desglosada de los umbrales óptimos (LSTMs)

Repetimos el experimento utilizando el modelo basado en VAR. En la tabla 4.2 recogemos los re-

sultados de la clasificación. Podemos comprobar que en este caso también se obtienen clasificaciones satisfactorias. De hecho, el método auto regresivo obtiene mejores resultados que el basado en redes neuronales (para este servidor concreto).

Paquete de incidencias	Umbral	F_2	Precisión	Sensibilidad
1	30	0.8396	97.46 %	81.15 %
2	10	0.9499	96.18 %	94.7 %
3	7	0.9192	70.29 %	99.58 %
4	18	0.9309	72.93 %	100.0 %
5	18	0.9717	87.3 %	100.0 %

Tabla 4.2: Información desglosada de los umbrales óptimos(VAR)

4.1.2. Prueba con varios servidores

En aras de facilitar la legibilidad, en la sección anterior se presentaron los resultados de un único servidor. Nos gustaría contrastar si se obtienen resultados similares en otros servidores. En esta sección repetimos el experimento anterior en los 10 servidores seleccionados en 3.1.2.

Al igual que en la prueba con un servidor, inducimos las incidencias y calculamos los F_2-*score* por cada umbral. Luego, para presentar los resultados, calcularemos el porcentaje de servidores cuya de clasificación es 'satisfactoria'. Esto nos permitirá evaluar si los modelos funcionan consistentemente en otros servidores de alta agregación.

Consideraremos una clasificación como 'satisfactoria' si obtiene un F_2-*score* superior a 0.8, utilizando el umbral que maximiza esta métrica.

Para esta sección presentaremos los resultados para dos paquetes de incidencias (el 1 y el 5). El primero afecta únicamente a una característica de red, mientras que el quinto simula una circunstancia más realista en la que varias series temporales se ven afectadas simultáneamente.

Resultado con varios servidores

Presentamos los resultados para cada paquete de incidencias por separado:

1.– **El paquete de incidencias 1:** Se obtuvo con LSTM una clasificación satisfactoria en 6 de los 10 servidores y para el modelo basado en VAR se obtuvo en 8 de 10.

2.– **El paquete de incidencias 5:** Se obtuvo con LSTM una clasificación satisfactoria en 6 de los 10 servidores y para el modelo basado en VAR se obtuvo en 10 de 10.

Para ambos modelos tenemos que la detección de incidencias artificiales es satisfactoria en más de la mitad de los servidores estudiados, independientemente del paquete de incidencias aplicado. Sin

embargo, el paquete de incidencias 5 obtiene una mejor clasificación que el 1. Esto se debe probablemente a que al alterar varias características a la vez, las métricas de red se alejan más del vector estimado y como consecuencia se facilita la clasificación. En términos de la evaluación, este hecho resulta positivo ya que el paquete de incidencias 5 fue diseñado precisamente para parecerse más a un entorno real.

Por un lado, este experimento sugiere que los modelos de clasificación de incidencias tienen su ámbito de aplicación en algunos servidores de alta agregación. Por otro lado, esta prueba compone un *test* para determinar si el modelo es efectivo sobre un servidor dado. Basta con tomar los datos de red históricos, inducir incidencias artificiales y comprobar si alguno de los dos modelos consigue una clasificación satisfactoria. En caso de conseguirla, el gestor de red puede utilizar los resultados de la prueba para elegir el umbral del servidor de acuerdo a sus exigencias de sensibilidad y precisión.

Partiendo de los resultados del experimento, uno podría preguntarse si el método auto regresivo es siempre superior al modelo basado en LSTMs. Examinando los resultados de la clasificación para cada servidor por separado, obtenemos que, si bien el método auto regresivo suele tener mejores métricas, hay algunos servidores para los cuales las LSTM tienen mejores resultados. En definitiva, el modelo idóneo a utilizar depende del servidor concreto.

Por último, recordamos que este experimento utiliza incidencias **artificiales** por lo que cabe todavía estudiar los resultados de los modelos en circunstancias completamente reales. En la siguiente sección abordamos esta cuestión.

4.2. Análisis de incidencias en datos reales

Hemos comprobado que los modelos son capaces de detectar con bastante precisión incidencias inducidas artificialmente. En esta sección estudiaremos cómo se desenvuelven los modelos utilizando datos reales, sin incidencias inducidas. Los datos reales no están clasificados, por lo que el principal objetivo del experimento es obtener el número de incidencias que detecta el sistema por servidor por día, para comprobar si el sistema sería utilizable en la práctica. Luego, inspeccionaremos manualmente algunas de las alarmas disparadas por el sistema, para evaluar si correspondían, o no, con una incidencia real.

4.2.1. Sobre los datos del experimento

Al igual que para el experimento anterior, de los registros de cada servidor se toma el 70 % inicial para el entrenamiento de los modelos y el 30 % final para realizar las pruebas. Ambos conjuntos de datos se utilizan totalmente inalterados, como se recopilarían en un contexto real.

4.2.2. Sobre el experimento

Esencialmente, el experimento simula la siguiente circunstancia:
El día 2 de agosto del 2019 se despliega el sistema en el centro de datos estudiado para detectar incidencias en 10 servidores. El sistema habría sido entrenado con los datos de los pasados meses junio y julio. Para los modelos, seleccionamos por cada servidor el umbral que resultó en el mejor clasificador cuando se indujeron incidencias artificiales. Luego, durante los siguientes 15 días, se monitorizan las alarmas que el sistema notifica.

Los resultados de la monitorización se dividen en dos partes:

Parte 1

La primera, estudia el número de incidencias que notifica el sistema. Se calcula y discute, para ambos modelos, el número medio, mínimo y máximo de incidencias obtenidas por servidor en los 15 días.

Parte 2

Aunque la inspección manual de todas las alarmas que notifica el modelo es inviable, podemos analizar casos concretos. Para este apartado, elegimos dos servidores y estudiamos las incidencias que detectan ambos modelos.

Recordamos que el modelo nos permite ordenar las características de red que fueron más relevantes durante la clasificación de la incidencia (véase la sección 3.2.3). Utilizando esta información ordenaremos las series temporales que fueron más importantes para cada alarma y serán estas las que inspeccionemos en busca de comportamiento anómalo, replicando el procedimiento que seguiría un gestor utilizando el modelo.

De este resultado tendremos una idea superficial de la precisión del modelo cuando es implementado en un entorno real.

4.2.3. Resultados

Parte 1

En el caso de las LSTM, se obtuvo de media 10.4 incidencias por servidor durante los 15 días. El servidor con menos incidencias obtuvo solo 3 incidencias, y el servidor con más incidencias notificó 29 anomalías durante esas dos semanas. Para el modelo basado VAR, se obtuvo de media 11.3 incidencias por servidor, con un mínimo de 5 incidencias y un máximo de 19 alarmas durante esos mismo 15 días.

Por otra parte, de los 10 servidores estudiados, 6 tuvieron más alarmas con el modelo basado en

VAR y 4 tuvieron más con el modelo basado en LSTM. Teniendo en cuenta que la precisión parece ser cercana al 100 % (lo veremos en la siguiente parte de los resultados), el modelo que detecta más incidencias es el más efectivo. Con esto concluimos que la eficacia del modelo (VAR frente a LSTM) depende fuertemente del servidor en cuestión. Esto no debería de suponer un problema ya que, realizando un estudio previo a su despliegue, se puede decidir de manera automatizada qué modelo se ajusta mejor a cada servidor concreto [1].

Parte 2

Para la inspección manual de incidencias se eligieron dos servidores: uno que tuviese más incidencias detectadas con el modelo auto regresivo y el otro que tuviese más incidencias detectadas con el modelo basado en LSTM. A continuación exponemos cada uno:

Parte 2 - Servidor 1

Para el primer servidor se detectaron 9 incidencias con el modelo basado en VAR y 5 con el basado en LSTM. En este caso, el modelo auto regresivo detectó las 5 anomalías del otro modelo más otras 4 distintas.

Inspeccionando manualmente, comprobamos que todas las alarmas detectadas coinciden con comportamientos anómalos, de modo que, la *precisión aparente* para esta prueba es de un 100 %. Ilustramos en la figura 4.1 algunas de las incidencias detectadas en los datos reales. En círculos verdes marcamos las incidencias detectadas por el modelo, que claramente componen circunstancias dignas de atención por parte de un gestor. Con recuadros rojos señalamos ejemplos de comportamientos extraños que no fueron detectados por el clasificador. La presencia de recuadros rojos demuestra que la sensibilidad del modelo es inferior al 100 %.

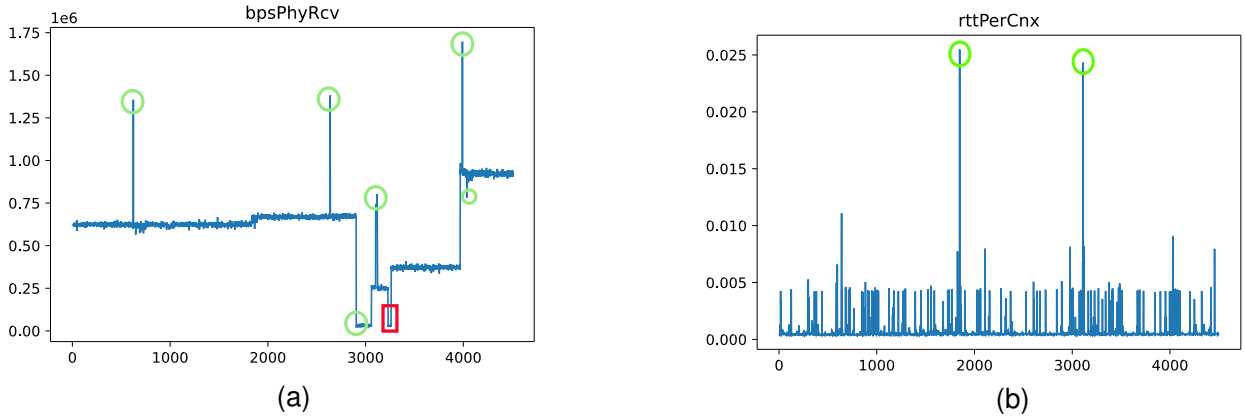


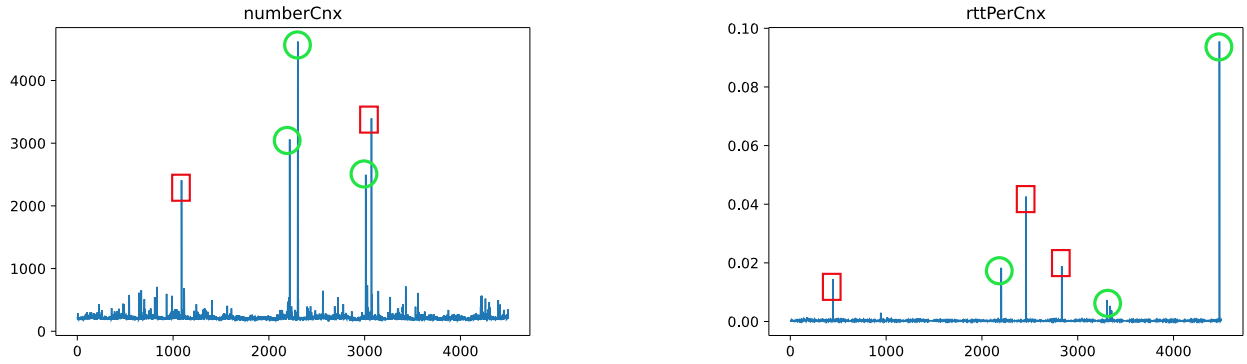
Figura 4.1: Series de características de red del primer servidor.

Parte 2 - Servidor 2

[1] Para decidirlo puede utilizarse un test basado en el experimento de la sección 4.1

El segundo servidor estudiado tuvo un total de 9 incidencias para el modelo basado en LSTM y 5 para el modelo auto regresivo. En este caso, las incidencias detectadas por el modelo LSTM contienen todas las incidencias detectadas por el modelo basado en VAR.

De las 9 incidencias detectadas por el modelo basado en LSTM, 8 de ellas correspondían con comportamientos anómalos (de las cuales las 5 del modelo auto regresivo eran correctas). De nuevo, ilustramos en la figura 4.2 algunas de las incidencias detectadas por los modelos. En este segundo servidor, la sensibilidad aparente parece ser considerablemente más baja, ya que encontramos varios ejemplos de comportamientos inusuales que no detecta el sistema.



(a) (b)

Figura 4.2: Series de características de red del segundo servidor servidor.

Sobre los resultados

En entornos reales podríamos esperar que un mismo técnico de redes gestionase en el orden de miles de servidores. En estas circunstancias, y de acuerdo con los resultados obtenidos, el número de incidencias que dispara el modelo desbordaría la capacidad de trabajo de cualquier gestor. Sin embargo, recordamos aquí que el modelo permite calificar las incidencias por su gravedad (véase la sección 3.2.3). Esto permite mostrar al gestor las incidencias ordenadas y así poder concentrar su esfuerzo en subsanar las incidencias más críticas.

4.3. Tiempo de cómputo

En el contexto de gestión de red, los centros de datos estudiados no superan las decenas de miles de servidores. Por tanto, este experimento busca estudiar cuáles son las prestaciones necesarias para desplegar el modelo en centros con este orden de magnitud.

Utilizando los datos que se disponen, calculamos el tiempo que cada modelo tarda en clasificar 72400 registros de red:

1.– El modelo basado en LSTM tardó 14 segundos en clasificarlos.

2.– El modelo basado en VAR tardó 6 segundos.

Dado que los registros se recogen cada 5 minutos y que el orden de magnitud de los centros de datos es de diez mil servidores, concluimos que el tiempo de ejecución de los clasificadores es despreciable frente al intervalo en el que se agregan las características de red. El modelo podría integrarse en una herramienta de gestión de red sin necesidad de equipo especializado ni optimizaciones.

5

Conclusiones

En este proyecto se desarrolló un sistema para la detección automatizada de incidencias en grandes centros de datos. El objetivo era obtener un modelo que, automatizando parte de la gestión de red, facilitara a un potencial gestor el manejo de un gran volumen de servidores.

Tras la definición e implementación del modelo, se procedió a evaluarlo para verificar su eficacia y viabilidad:

En el primer experimento, se simularon incidencias en los datos y comprobamos cómo los modelos propuestos eran capaces de detectar las incidencias inducidas de manera efectiva. Además, el propio experimento componía un *test* para descartar aquellos servidores cuyo comportamiento errático era difícil de predecir y, por tanto, de automatizar.

Como segundo experimento, utilizamos datos reales para simular el modelo en un entorno real. De la simulación, una inspección manual de las incidencias detectadas indicó una precisión aparente cercana al 100 %. Sin embargo, comprobamos también que el número de alarmas disparadas por el sistema desbordaría a un gestor humano. Esto indica que los comportamientos anómalos detectados por el sistema son probablemente de importancia secundaria. Para ello se propone una alternativa con la cuál el gestor puede ordenar la incidencias por relevancia, utilizando una métrica extraída del propio modelo.

Finalmente, una breve prueba demostró que el tiempo de cómputo del modelo es despreciable frente al tiempo de agregación de los datos. Por lo que no se requieren consideraciones especiales para implementar el sistema en un entorno real.

De los experimentos podemos concluir que el modelo propuesto compone una alternativa viable para la detección automatizada de incidencias. Ahora bien, cabe destacar que hay espacio para la investigación futura y el perfeccionamiento del modelo. Un interesante frente de desarrollo podría ser adaptar el modelo para considerar la retroalimentación de un gestor en tiempo real. Esto permitiría al modelo aprender a adaptar la sensibilidad y precisión de sus alarmas, a la vez que generar un conjunto de datos etiquetados que permitan el desarrollo de futuros modelos supervisados.

Bibliografía

[1] Pavel Filonov, Andrey Lavrentyev, and Artem Vorontsov. Multivariate industrial time series with cyber-attack simulation: Fault detection using an lstm-based predictive data model. 12 2016.

[2] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9:1735–80, 12 1997. doi:10.1162/neco.1997.9.8.1735.

[3] Fazle Karim, Somshubra Majumdar, Houshang Darabi, and Shun Chen. Lstm fully convolutional networks for time series classification. *IEEE Access*, PP, 09 2017. doi:10.1109/ACCESS.2017.2779939.

[4] Rashid Mijumbi, Abhaya Asthana, Markku Koivunen, Fu Haiyong, and Qinjun Zhu. Design, implementation, and evaluation of learning algorithms for dynamic real-time network monitoring: Learning algorithms for dynamic real-time network monitoring. *International Journal of Network Management*, page e2108, 03 2020. doi:10.1002/nem.2108.

Apéndices

Apéndice A

El experimento de la sección 4.1 estudia la efectividad de los modelos identificando incidencias artificiales. En este capítulo detallamos en qué consisten cada uno de los paquetes de incidencias utilizados en ese experimento.

A.1. Sobre los paquetes de incidencias

Cada paquete de incidencias representa un conjunto de modificaciones que se plasmará sobre los datos. Estos paquetes buscan reproducir las alteraciones en los datos que surgirían con una incidencia en un entorno real. Por ejemplo, podríamos replicar un ataque de denegación de servicio incrementando repentinamente en el número de flujos de un servidor (véase el paquete 4). Cada paquete definido en este apéndice atiende a un tipo distinto de incidencia que podría encontrarse en el ámbito de la gestión de red.

A.2. Detallando los paquetes de incidencias

Los 4 primeros paquetes modificarán únicamente una características de red. Podremos así contrastar si la clasificación es consistente independientemente de la característica modificada. Por su parte, el último paquete inducirá modificaciones en varias características a la vez, esto pondrá a prueba los modelos en un contexto más realista.

Cada paquete de incidencias consta de varias modificaciones en distintos intervalos e intensidades. Para más detalles sobre cómo se inducen las incidencias consulte la sección 3.3.

Paquete de incidencias 1

El paquete 1 dispara únicamente los bits por segundos recibidos. La especificación se encuentra en la tabla A.1.

Desde	Hasta	Rasgo	Clientes afectados	Factor multiplicativo
00:00:00 20-09-2019	03:00:00 21-09-2019	Bps recibidos	80.0 %	3
04:00:00 09-09-2019	23:00:00 10-09-2019	Bps recibidos	90.0 %	2

Tabla A.1: Especificación del paquete de incidencias 1

Paquete de incidencias 2

El paquete 2 dispara los RTT medios. La especificación se encuentra en la tabla A.2.

Desde	Hasta	Rasgo	Clientes afectados	Factor multiplicativo
00:00:00 03-09-2019	03:00:00 05-09-2019	RTT	90.0 %	3
12:00:00 10-09-2019	16:00:00 15-09-2019	RTT	60.0 %	4

Tabla A.2: Especificación del paquete de incidencias 2

Paquete de incidencias 3

El paquete 3 dispara el número de retransmisiones. La especificación se encuentra en la tabla A.3.

Desde	Hasta	Rasgo	Clientes afectados	Factor multiplicativo
04:00:00 12-09-2019	12:00:00 12-09-2019	Retransmisiones	100 %	1.5
08:00:00 04-09-2019	12:00:00 04-09-2019	Retransmisiones	90.0 %	2
03:00:00 21-09-2019	23:00:00 21-09-2019	Retransmisiones	70.0 %	3

Tabla A.3: Especificación del paquete de incidencias 3

Paquete de incidencias 4

El paquete 4 dispara el número de conexiones. La especificación se encuentra en la tabla A.4.

Desde	Hasta	Rasgo	Clientes afectados	Factor multiplicativo
00:00:00 21-09-2019	03:00:00 21-09-2019	Conexiones	100 %	3
12:00:00 10-09-2019	23:00:00 10-09-2019	Conexiones	100 %	10

Tabla A.4: Especificación del paquete de incidencias 4

Paquete de incidencias 5

El paquete 5 se compone de modificaciones a varias series temporales. Contiene modificaciones de cada uno de los 4 paquetes de incidencias anteriores. La especificación se encuentra en la tabla A.5.

Desde	Hasta	Rasgo	Clientes afectados	Factor multiplicativo
00:00:00 21-09-2019	03:00:00 21-09-2019	Conexiones	100 %	4
23:00:00 04-09-2019	03:00:00 05-09-2019	RTT	90.0 %	4
03:00:00 20-09-2019	06:00:00 20-09-2019	Retransmisiones	70.0 %	6
00:00:00 18-09-2019	03:00:00 19-09-2019	Bps recibidos	80.0 %	3

Tabla A.5: Especificación del paquete de incidencias 5