Pruebas de APIs REST guiadas por Aprendizaje Activo*

A. Giuliano Mirabella, Alberto Martin-Lopez,
Sergio Segura, Luis Valencia-Cabrera, and Antonio Ruiz-Cortés

SCORE Lab, I3US Institute, Universidad de Sevilla, Seville, Spain
{amirabella,alberto.martin,sergiosegura,lvalencia,aruiz}@us.es

Resumen La generación automática de casos de prueba para APIs REST es un tema de investigación muy activo. La mayoría de técnicas emplean un enfoque de caja negra basado en la generación aleatoria de peticiones a partir de la especificación de la API. Dichas técnicas tienen una limitación importante: ignoran las *dependencias inter-parámetro* (restricciones entre parámetros que se deben cumplir para que la petición a la API sea válida), ya que no están soportadas por los lenguajes de especificación actuales. Como resultado, la mayoría de peticiones generadas automáticamente viola alguna dependencia y es rechazada por la API. En este artículo, proponemos un método para entrenar eficientemente un clasificador que prediga la validez de las peticiones, para así poder descartar las inválidas antes de invocar a la API. Nuestra técnica aprende a medida que genera casos de prueba, de forma que el porcentaje de llamadas válidas aumenta progresivamente hasta un 90 % en APIs comerciales como GitHub y Stripe. Estos resultados prometedores sugieren que nuestra propuesta podría mejorar significativamente la generación automática de casos de prueba para APIs REST.

Palabras clave: APIs RESTful · servicios web · pruebas software.

1. Introducción

Las APIs web RESTful (también denominadas *APIs REST*) [1] son el estándar de facto para la integración en la Web. Estas APIs exponen una interfaz uniforme a través de la cual es posible acceder a datos y servicios, mediante interacciones HTTP. Un fenómeno común en las API REST es que presentan *dependencias inter-parámetro* (o simplemente *dependencias*), es decir, restricciones entre dos o más parámetros de entrada que se deben cumplir para formar llamadas válidas al servicio. Por ejemplo, en la API de Google Maps, al buscar lugares, si

* Trabajo financiado por la Comisión Europea (FEDER), la Junta de Andalucía bajo los proyectos APOLO (US-1264651) y EKIPMENT-PLUS (P18-FR-2895), el Gobierno de España bajo el proyecto HORATIO (RTI2018-101204-B-C21), financiado por FEDER/Ministerio de Ciencia e Innovación – Agencia Estatal de Investigación, y el Programa de ayudas FPU, del Ministerio de Educación y Formación Profesional (FPU17/04077) y la Red de Excelencia SEBASENet 2.0 (RED2018-102472-T).

se usa el parámetro `location`, también debe utilizarse el parámetro `radius`; de lo contrario, se devuelve un error (código de estado 400, "Bad Request"). Un estudio reciente [3] reveló que estas dependencias son muy comunes: aparecen en 4 de cada 5 APIs, en todos los dominios de aplicación y tipos de operaciones. Lamentablemente, los lenguajes de especificación de API actuales, como OpenAPI Specification (OAS) [5], no ofrecen soporte para la descripción formal de este tipo de dependencias.

En un estudio anterior [4] demostramos que, haciendo uso de técnicas de aprendizaje automático, se puede entrenar un algoritmo de clasificación para predecir la validez de una llamada a la API, es decir, si satisface todas las dependencias de la API o no, evitando así peticiones innecesarias a la API. Este enfoque es eficiente (es completamente automático) y efectivo (genera un alto número de peticiones válidas), pero requiere de un conjunto de entrenamiento (dataset) lo suficientemente variado y equilibrado. Un dataset de estas características es costoso de conseguir con las técnicas actuales de pruebas, pues conlleva la generación de muchas peticiones y el consiguiente gasto de recursos de la API (siendo frecuentemente un factor limitante el número de llamadas permitidas a la misma). El objetivo de este trabajo es proponer una técnica para la recolección eficiente de un dataset de peticiones y respuestas, maximizando el aprendizaje del clasificador de peticiones, y por tanto incurriendo en el mínimo número posible de llamadas a la API.

2. Aprendizaje activo

Un sistema de aprendizaje supervisado debe ser entrenado con cientos o miles de observaciones etiquetadas. Hay casos en los que estas observaciones tienen un coste mínimo (por ejemplo, la calificación que los usuarios dan a las películas de un sitio web), pero en otros casos la obtención de las etiquetas puede suponer un coste no despreciable en términos de tiempo, dinero o recursos. Este es el caso que nos ocupa, donde para etiquetar las llamadas de entrenamiento como válidas o inválidas es necesario invocar a las API cientos o miles de veces.

El aprendizaje activo (AL por su nombre en inglés) es una rama del aprendizaje automático cuya idea clave es que el algoritmo de aprendizaje pueda elegir activamente las observaciones de las que aprende [6]. La idea clave del AL es que se puede lograr una mayor precisión con menos etiquetas de entrenamiento si al algoritmo se le permite consultar interactivamente una fuente de información, llamada oráculo, para etiquetar nuevas observaciones con las salidas correctas [6]. Por ejemplo, transcribir un audio a texto puede durar hasta diez veces más que el audio original, y requiere de lingüistas formados. En el paradigma de AL, es el algoritmo el que pide activamente al lingüista que transcriba determinados audios, para posteriormente entrenarse sobre ellos [7].

3. Propuesta

El objetivo de este trabajo es maximizar el porcentaje de llamadas válidas enviadas a la API de forma completamente automática. Para ello, proponemos

una técnica basada en AL que permita recolectar un conjunto de entrenamiento de forma óptima. La técnica que proponemos consiste en dos fases: inicio y aprendizaje (Figura 1).

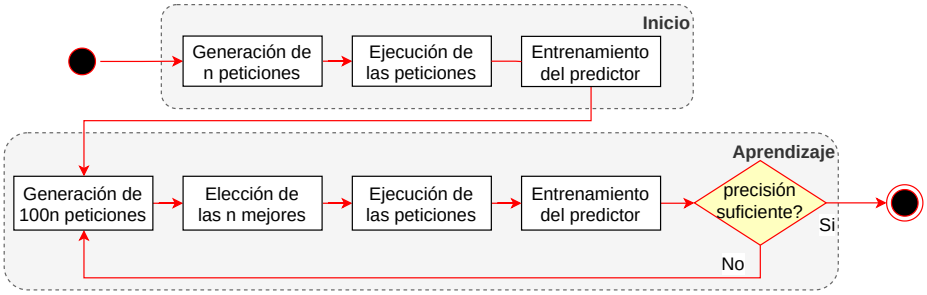

Figura 1: Diagrama de la propuesta.

3.1. Inicio

El proceso comienza con la generación de n peticiones aleatorias, que se ejecutan invocando a la API y de las que se recogen las respuestas. Se etiquetan como válidas aquellas peticiones que hayan recibido un código de estado 2XX, y como inválidas las que hayan recibido un 4XX (esto ocurre sólo cuando se viola alguna dependencia: descartamos la posibilidad de que los errores se deban a valores individuales usando diccionarios de datos predefinidos.). Finalmente, se entrena el predictor sobre estas observaciones.

3.2. Aprendizaje

Mientras la precisión del predictor sea menor que un umbral configurable, se llevará a cabo el aprendizaje activo por iteraciones. En cada iteración, se generan $100n$ peticiones aleatorias. De entre esas, se escogen las n peticiones con mayor *incertidumbre*, siendo ésta tanto mayor cuanto más se parezcan las probabilidades de que esta sea válida o inválida. Por ejemplo, de las peticiones mostradas en la Tabla 1, la que tiene mayor incertidumbre es la número 3, ya que las probabilidades de ser válida (P_v) e inválida (P_i) son muy parecidas (0.49 y 0.51, respectivamente). Finalmente, se invoca la API con las peticiones escogidas, se etiquetan las peticiones y se vuelve a entrenar el predictor, obteniendo otro valor de precisión.

Tabla 1: Ejemplo de tres peticiones sin etiquetar.

ID	**type**	**visibility**	**affiliation**	**sort**	**direction**	P_v	P_i
1	-	'all'	'collaborator'	-	'asc'	0.9	0.1
2	'private'	-	'owner'	'created'	-	0.3	0.7
3	'public'	-	'collaborator'	-	-	0.49	0.51

4. Evaluación preliminar

Hemos evaluado la técnica propuesta en dos operaciones de APIs comerciales: la lectura de repositorios en GitHub y la creación de un producto en Stripe. Para cada operación, generamos 2000 peticiones e intentamos maximizar el número de válidas (aquellas que obtengan un código de estado 2XX). La Tabla 2 muestra el porcentaje de peticiones válidas obtenidas con nuestra técnica (AL) frente a las obtenidas con técnicas aleatorias (Random). El porcentaje de peticiones válidas crece mucho con nuestra propuesta, llegando a un $\sim 98\,\%$. Gracias a esto, somos capaces de detectar hasta el triple de errores (*failures*) de disconformidad con la especificación. La propuesta ha sido implementada en Python con `pandas` y `scikit-learn`, y la técnica de clasificación usada es el Random Forest [2].

Tabla 2: Resultados.

API	Peticiones válidas (%)		Errores	
	Random	AL	Random	AL
GitHub	62.10	98.65	313	1159
Stripe	55.75	99.30	104	284
Media	58.93	98.98	209	722

5. Conclusiones

En este artículo presentamos una nueva técnica para la generación automática de casos de prueba más eficientes en APIs REST. Los resultados preliminares muestran que la técnica basada en aprendizaje activo, consigue hasta un $\sim 98\,\%$ de peticiones válidas, frente al $\sim 59\,\%$ conseguido por técnicas aleatorias, en APIs comerciales de GitHub y Stripe, triplicando el número de errores detectados y demostrando el potencial de la inteligencia artificial para la generación automática de casos de prueba en APIs REST.

Referencias

1. Fielding, R.T.: Architectural Styles and the Design of Network-based Software Architectures. Ph.D. thesis (2000)
2. Ho, T.K.: Random decision forests. In: Proceedings of 3rd international conference on document analysis and recognition. vol. 1, pp. 278–282. IEEE (1995)
3. Martin-Lopez, A., Segura, S., Ruiz-Cortés, A.: A Catalogue of Inter-Parameter Dependencies in RESTful Web APIs. In: International Conference on Service-Oriented Computing. pp. 399–414 (2019)
4. Mirabella, A.G., Martin-Lopez, A., Segura, S., Valencia-Cabrera, L., Ruiz-Cortés, A.: Deep Learning-Based Prediction of Test Input Validity for RESTful APIs. In: International Workshop on Testing for Deep Learning and Deep Learning for Testing (2021)
5. OpenAPI Specification, `https://www.openapis.org`, accessed April 2020
6. Settles, B.: Active learning literature survey. Computer Sciences Technical Report 1648, University of Wisconsin–Madison (2009)
7. Zhu, X., Lafferty, J., Rosenfeld, R.: Semi-Supervised Learning with Graphs. Ph.D. thesis, USA (2005), aAI3179046