Testing en Big Data y aprendizaje automático

Alumno: Nicolás Ezequiel Granata

Tutor Técnico: Dr. Fernando Asteasuain

Profesora Trabajo Final: Dra. Marcela Samela

Trabajo Final presentado para obtener el título de
Licenciatura en Gestión de Tecnología Informática

(Diciembre, 2021)

Resumen

La tecnología de Big Data y aprendizaje automático se ha convertido en uno de los temas de principal interés, no solo en el campo de la investigación sino también en las organizaciones, empresas de desarrollo de software y demanda del mundo moderno en general. Esto se debe a la posibilidad de obtener nuevos conocimientos mediante algoritmos complejos, a partir de grandes y diversos volúmenes de datos, los cuales continúan creciendo día a día.

La implementación de los sistemas de Big Data y aprendizaje automático presentan un problema a la hora de realizar pruebas sobre estos, debido a la cantidad y diversidad de datos que se procesan y los resultados que se obtienen. Asimismo, estos tipos de sistemas son calificados dentro de los *no testeables*, teniendo en cuenta la no existencia de un oráculo específico que permita indicar si los resultados de los casos prueba son correctos.

El presente trabajo tuvo como objetivo investigar las propuestas de pruebas en sistemas de Big Data y aprendizaje automático, mediante el análisis de las estrategias utilizadas por la comunidad en el área, haciendo especial foco en el acercamiento de pruebas metamórficas.

Como aplicación práctica, se realizó un procedimiento de pruebas metamórficas en el cual se verificó el funcionamiento de un sistema de software que analiza, mediante reconocimiento automático, sentimientos de textos escritos en Twitter. Por otro lado, se realizó un segundo procedimiento donde se verificó el funcionamiento del motor de búsqueda de Google.

A partir de la investigación realizada se observó que la generación de relaciones metamórficas, teniendo en cuenta el dominio del sistema, brindaron una solución posible al problema de la ejecución y verificación de pruebas.

Palabras Clave: aplicaciones de big data, aprendizaje automático, pruebas de software

Abstract

Big Data and machine learning technology has become one of the main topics of interest not only in the field of research but also in the organizations, software development companies and demand from the modern world in general. This is due to the possibility of getting new knowledge from large and complex volumes of data, which continues to grow day by day.

The implementation of Big Data and machine learning systems presents a problem when testing them, because of the amount and diversity of data that is processed and the results that are obtained. Likewise, these types of systems are classified within the non-testable, considering the non-existence of a specific oracle to indicate whether the results of the test cases are correct.

This work aimed to investigate the testing proposals in Big Data and machine learning systems, through the analysis of the strategies used by the community in the area, with special focus on the metamorphic testing approach.

As a practical application, a metamorphic testing procedure was performed in which the behavior of a software system that analyzes sentiments through automatic recognition of texts written in Twitter was verified. On the other hand, a second procedure was performed where the operation of the Google search engine was verified.

From the research carried out, it was observed how through the generation of metamorphic relations considering the system domain, provided a possible solution to the problem of test execution and verification.

Keywords: big data applications, machine learning, software testing

Dedicatoria

El presente trabajo está dedicado a mi familia, amigos y amigas por el apoyo que me brindan en todo momento.

Reconocimientos

Al Dr. Fernando Asteasuain del Centro de Altos Estudios en Tecnología Informática, por la ayuda, predisposición y tiempo brindado para la realización del presente trabajo. A la Dra. Marcela Samela por la orientación metodológica brindada durante la cursada de Trabajo Final.

Índice General

Índice de Figuras

Índice de Tablas

Capítulo 1

Introducción

1.1 Problemas y Soluciones

Big Data y aprendizaje automático forman parte de las tecnologías más importantes en materia de investigación y el uso de estos servicios se expanden en sistemas de predicciones, recomendaciones, patrones de reconocimiento, buscadores, redes sociales, entre otros (Jin et al., 2015) (Ding, Zhang, & Hu, 2016) (Tao & Gao, 2016) (Mahesh, 2020). Servicios de consumo masivo utilizados por gran parte de las personas como Netflix[1], Spotify[2], Facebook[3] o YouTube[4], hacen uso de estos sistemas para personalizar contenido, crear campañas de marketing o mejorar la experiencia del usuario a partir de los datos que se generan, los cuales están en constante crecimiento. No solo es posible ver el uso de los datos en estas plataformas de entretenimiento, también se encuentran ligados a otros ámbitos por ejemplo el deporte donde ligas como la NBA hacen uso de Big Data para rastrear y monitorear movimientos de los jugadores (Hayhurst, 2020).

El uso de esta tecnología se encuentra presente en casi todos los campos que generan datos masivos desde la ciencia de la tierra, la oceanografía, pasando por la aeronáutica, la física y la ciencia del genoma, entre otras (Patgiri & Ahmed, 2016).

Parte esencial a la hora de realizar el desarrollo e implementación de un sistema de software es el aseguramiento de la calidad mediante la aplicación de técnicas testing (Cano, 2015). Realizar esta tarea en sistemas de Big Data presenta desafíos propios inherentes a las características de esta tecnología. Validar cuan exacto es un resultado es complejo debido a la gran cantidad de datos que se analizan, por lo que el aseguramiento de la calidad es un tema importante por abordar (Tao & Gao, 2016). Aunque Big Data se ha convertido en un importante campo de investigación en los últimos tiempos, los trabajos sistemáticos sobre el aseguramiento de la calidad de estos sistemas son escasos en la literatura (Ding, Hu, & Gudivada, 2017).

Algunos autores los califican dentro de los softwares no testeables (Otero & Peter, 2015), debido a que no hay un acercamiento específico que permita generar y ejecutar los casos prueba. Este un problema que debe tenerse en cuenta ya que la validez y veracidad de los resultados obtenidos, dependerá de la capacidad de validar los mismos.

---

[1] https://research.netflix.com/research-area/analytics
[2] https://engineering.atspotify.com/tag/big-data/
[3] https://research.fb.com/category/data-science/
[4] https://www.youtube.com/

Al no contar con una aproximación específica para la realización de pruebas, es posible mencionar la necesidad de contar con modelos y criterios de validación apropiados para las características de los sistemas de Big Data. Contando con una de cobertura de pruebas ajustadas a las necesidades de estos, como también el desarrollo de herramientas y soluciones que permitan automatizarlas (Tao & Gao, 2016).

Una forma de abordar el aseguramiento de la calidad, generar casos de pruebas y verificar los resultados en sistemas que no poseen un acercamiento de pruebas específico es mediante una técnica llamada testing metamórfico (Chen, Cheung, & Yiu, 1998). Está estrategia no se enfoca en cada salida individual, sino que a través de relaciones metamórficas, comprueba la relación de las salidas mediante la verificación de las pruebas durante múltiples ejecuciones del programa, las cuales especifican como debería cambiar una salida de acuerdo a los datos que se utilizan como entradas (Segura, Towey, Zhou, & Chen, 2020) (Ding, Zhang, & Hu, A Framework for Ensuring the Quality of a Big Data Service, 2016) (Tao & Gao, 2016).

1.2 Hipótesis

Se establecen las siguientes hipótesis: la adecuada delimitación de las relaciones metamórficas, permiten verificar el funcionamiento en sistemas de aprendizaje automático y Big Data.

El refinamiento iterativo de las relaciones metamórficas es necesario para la correcta construcción de los casos de prueba a ejecutar.

1.3 Nuestra Propuesta

El presente trabajo propone realizar una investigación cualitativa del proceso de pruebas en sistemas de Big Data y aprendizaje automático. Se define como técnicas de recolección de datos a emplear el análisis de contenidos mediante la revisión de documentos y estudio de caso.

Para ello se presenta el estado del arte actual poniendo particular foco en el acercamiento metamórfico, donde es posible generar los casos de prueba mediante el armado de relaciones metamórficas. Así mismo, se persigue la posibilidad de realizar un procedimiento de pruebas que permita ser automatizado mediante el uso de herramientas informáticas.

1.4 Objetivo del Trabajo Final

El presente trabajo tiene como objetivo realizar una indagación y análisis acerca de las técnicas de testing que permitan garantizar el aseguramiento de la calidad de los resultados producidos en sistemas de Big Data y aprendizaje automático. Los objetivos particulares del mismo son:

- Analizar el enfoque de testing metamórfico aplicado en sistemas de Big Data y aprendizaje automático, su estrategia de desarrollo, beneficios, complejidad y aplicación.
- Realizar una propuesta de aplicación de testing metamórfico en el análisis de sentimiento en textos escritos en la red social Twitter y en el motor de búsqueda de Google.
- Analizar de forma introductoria otras propuestas de testing que puedan aplicarse en sistemas de Big Data y aprendizaje automático como alternativa al acercamiento metamórfico.

1.5 Contribuciones Principales

El uso de los sistemas informáticos se ha vuelto esencial en las compañías y en las personas en general. Desde aplicaciones de correo electrónico, pasando por redes sociales hasta automóviles conectados, forman parte de un ecosistema en el cual los datos tienen gran importancia y la generación de los mismos se da en forma exponencial (World Economic Forum, 2019). Estos pueden provenir de diferentes fuentes y en diversas formas, los sistemas de Big Data hacen uso de ellos junto con algoritmos de aprendizaje automático para detectar patrones, realizar predicciones o revelar información que de otra forma no podría conseguirse. Uno de los retos más importantes que esto conlleva, es cómo poder asegurar la exactitud y precisión de los resultados que se obtienen mediante los procesos que ejecutan estos sistemas. En este sentido, el testing se ha vuelto uno de los desafíos que las compañías y la comunidad en el área intentan resolver.

El trabajo de investigación que se presenta a continuación busca plasmar un procedimiento práctico que permita validar los resultados de la ejecución y lleve a la posibilidad de automatizar las pruebas en sistemas que analizan datos de forma masiva.

El procedimiento podrá ser utilizado, no solo para el caso puntual que se analizará durante la investigación, sino que apunta a poder resolver las pruebas sobre diversas implementaciones y sistemas que hagan uso de Big Data y aprendizaje automático.

1.6 Estructura General del Trabajo Final

En el capítulo 2 “Marco Teórico” se presenta el origen del concepto de Big Data y sus características explicadas desde las 5-V, volumen, velocidad, variedad, valor y veracidad. La Internet como fuente principal de generación de datos, la evolución de su uso, crecimiento exponencial de esta y el impacto en la utilización de Big Data. Se expone el concepto de aprendizaje automático, su origen, evolución y sus categorías esenciales de algoritmos y aplicaciones. Por último, se explica el concepto de pruebas en sistemas y aplicaciones, su importancia dentro del proceso de la ingeniería de software y los tipos de pruebas que existen dentro del ciclo de vida del desarrollo de software.

En el capítulo 3 “Testing en Big Data y aprendizaje automático”, se expone un análisis sobre las técnicas de testing en sistemas de Big Data y aprendizaje automático, en particular el enfoque de pruebas metamórficas, su concepto, ventajas, la generación relaciones y la aplicación en los tipos de sistemas mencionados. Así mismo se brinda un análisis sobre cómo se llega a la propuesta metamórfica y los desafíos que lleva una implementación de este tipo de técnica. Por último, se analizan en forma introductoria otras técnicas que están siendo utilizadas por la comunidad, como las pruebas basadas en clasificación, pruebas de MapReduce y pruebas de origen colectivo.

En el capítulo 4 “Aplicación de testing metamórfico”, se propone la realización de un procedimiento de pruebas metamórficas en aplicaciones que consideramos para este trabajo sistemas de Big Data y aprendizaje automático. Para ello se aplican estas pruebas sobre un software de ejemplo, generado específicamente para este trabajo, que utiliza la red social Twitter[5] con el objetivo de realizar análisis de sentimientos de textos escritos por los usuarios. Por otro lado, se ejecuta un procedimiento de pruebas metamórficas sobre el motor de búsqueda de Google[6] para validar su funcionamiento, teniendo en cuenta las funcionalidades básicas para obtener los resultados de las búsquedas sobre una gran cantidad de datos. Por último, se analizan los resultados obtenidos del proceso de pruebas ejecutado, el cual verifica el cumplimiento o no de las relaciones metamórficas definidas para el contexto dado, donde es posible comprobar que tan válida es la delimitación de las relaciones metamórficas planteadas para encontrar defectos en los sistemas mencionados.

[5] https://about.twitter.com/es.html
[6] https://www.google.com/

Finalmente se encuentran reflejadas las conclusiones obtenidas como resultado de la investigación realizada, las líneas futuras de investigación y los anexos.

Capítulo 2

Marco Teórico

2.1 Introducción

En este capítulo se describe el marco teórico de los conceptos, características, utilización, crecimiento de Big Data. El concepto y origen del aprendizaje automático, sus algoritmos, categorías generales y aplicaciones de estos. Por último, se expone el proceso de testing como parte del ciclo de vida de desarrollo de software, sus niveles, estrategias principales y los tipos de prueba. Estos conceptos son los pilares de la investigación del presente trabajo.

2.2 Big Data

La tecnología de Big Data asiste al usuario a utilizar el poder de cómputo para aprovechar las consultas distribuidas sobre grandes volúmenes de datos y ofrecer un resultado deseado. En este sentido, la computación en la nube es un facilitador de esta tecnología, donde adicionalmente Big Data como servicio es uno de los campos de investigación más destacados en la actualidad (Patgiri & Ahmed, 2016).

Un término para describir Big Data es visualización (McNulty, 2014), ya que este es el proceso de mostrar información oculta, para mejorar el rendimiento de los datos y la toma de decisión.

A su vez, es posible explicar Big Data utilizando el concepto de las 3-V (Gartner, 2001), volumen, velocidad y variedad. Estas tres palabras están relacionadas de forma directa con las características de los sistemas de Big Data, ya que refieren de forma general a la cantidad de datos que existe, la velocidad con la que estos son generados y deben consumirse y la forma que tienen no es siempre la misma. Hay una diversidad de datos que pueden ir desde los obtenidos en una base de datos relacional hasta los textos que las personas escriben en sus redes sociales.

Estos datos pueden dividirse a grandes rasgos en dos categorías, los que se obtienen a través de sensores, experimentos científicos y observaciones, datos biológicos, neuronales y astronómicos, los cuales pertenecen al mundo físico, y los datos de la sociedad humana, obtenidos de diversas fuentes como internet, redes sociales, información financiera y de salud e incluso el transporte (Jin, Wah, Cheng, & Wang, 2015).

La característica de volumen está relacionada con la cantidad de datos la cual crece de forma exponencial y que pueden ser analizados y procesados debido a los avances del poder

de cómputo, no sólo referido al avance de los computadores en sí sino también a la computación en la nube, la cual permite una mayor escalabilidad de procesamiento (Patgiri & Ahmed, 2016).

En el año 2019 un artículo del World Economic Forum menciona que Facebook genera 4 Petabytes de datos por día, 500 millones de Tweets son enviados a través de la red social Twitter, 4 Terabytes de datos son generados por autos conectados por día, 3.5 billones de búsqueda son realizadas a través del buscador de Google y 294 billones de correo electrónicos son enviados por día. A su vez, se espera que en el 2020 se alcance una acumulación de datos en torno a los 44 Zettabytes y para el 2025 estiman que se crearan de forma global 463 Exabytes de datos por día (World Economic Forum, 2019). El volumen está íntegramente relacionado con la característica de velocidad, mediante la cual los datos son generados de forma exponencial y consumidos casi en tiempo real. Estos datos provienen de diferentes fuentes generadoras como pueden ser los usuarios que acceden a Internet, la computación en la nube, internet de las cosas (IoT), datos científicos (Patgiri & Ahmed, 2016).

La velocidad en la generación de datos continuará creciendo, según el informe "*Cisco Annual Internet Report* " de Cisco Systems Inc., alrededor de dos tercios de la población global tendrá acceso a Internet para el año 2023 donde habrá 5.3 mil millones de usuarios en total, un 66 por ciento de la población mundial sobre el 51 por ciento del año 2018 con 3.9 mil millones de usuarios conectados e internet. Más del 70 por ciento de la población global tendrá conectividad de internet móvil, donde crecerá de 5.1 billones en el año 2018 a 5.7 billones para el 2023 (Cisco Inc., 2020).

Dentro del IoT, las aplicaciones del tipo hogares conectados tendrán la mayor participación, cercana al 48 por ciento de las conexiones máquina a máquina y los automóviles conectados serán el tipo de aplicación de más rápido crecimiento, con un 30 por ciento de la tasa de crecimiento anual compuesto para el pronóstico del periodo entre 2018 y 2023 (Cisco Inc., 2020).

La variedad hace referencia a la forma que tienen los datos, estas estructuras incluyen desde datos tabulados, imágenes, videos, archivos XML, archivos JSON pasando por los datos publicados por usuarios en redes sociales. Incluye también la variedad de los formatos para un mismo tipo de dato como por ejemplo archivos .docx, .odt, .text, etc. para texto (Patgiri & Ahmed, 2016).

Como mencionan los autores Patgiri y Ahmed en su trabajo "*Big Data: The V's of the Game Changer Paradigm*" si bien existe una confusión y controversia acerca de las V

correspondientes a Big Data, las mencionadas son las más ampliamente aceptadas (Patgiri & Ahmed, 2016) y se incluyen dos más entre ellas que son valor y veracidad.

Los datos por sí mismo no tienen ningún valor, el análisis y procesamiento de estos mediante los sistemas de Big Data posibilitan la extracción de nuevos conocimientos, identificación de patrones y predicciones que permiten generar valor.

La veracidad es la exactitud y sentido de la información de los datos, indica cuanto podemos confiar en estos. "*Los datos inexactos pueden llevar a una dirección o decisión equivocada*" (Patgiri & Ahmed, 2016). Un ejemplo de esto puede ser un sistema de recomendación para usuarios de un servicio de videos, si se tienen muchos datos pero los mismos no reflejan los posibles intereses del usuario, no se obtendrán las visualizaciones que se desean.

Por último, es posible agregar una última característica, llamada validez, en este aspecto existen datos que, aunque el proceso pueda llevarse a cabo, sus resultados no tendrían sentidos ya que los datos utilizados no son válidos, por ejemplo, datos obsoletos sobre comercio electrónico. Por otro lado, está la posibilidad que datos obsoletos puedan ser válidos para determinado proceso, como por ejemplo registro de transacciones bancarias (Patgiri & Ahmed, 2016).

Según Fortune Business, la tecnología de Big Data continúa ganando mercado debido al aumento de las inversiones de actores clave, donde se prevé que el tamaño del mercado mundial alcance los 116.070 millones de dólares para el año 2027 (Fortune Business Insights, 2020). A su vez, Patgiri & Ahmed, indican que Big Data dominará al menos hasta el 2030 el mercado tecnológico, debido a que se introducen nuevas tecnologías que generan datos por cualquier medio día a día y, por lo tanto, los datos seguirán creciendo (Patgiri & Ahmed, 2016).

Es importante destacar un concepto que acompaña a Big Data, la inteligencia accionable o procesable (del inglés, *actionable intelligence*), que significa más que simplemente encontrar o resumir datos, es hacer foco en el descubrimiento de patrones ocultos, usualmente difíciles de encontrar o desconocidos (Otero & Peter, 2015). Estos pueden utilizarse para predecir acontecimientos, tendencias, opiniones, etc. que apoyen a los responsables de la toma de decisiones (Otero & Peter, 2015).

Se proponen tres niveles de inteligencia accionable, Nivel 1 (L1) inteligencia accionable supervisada, Nivel 2 (L2) inteligencia accionable semi-supervisada y Nivel 3 (L3) inteligencia accionable no supervisada (Otero & Peter, 2015).

En el nivel 1, el humano supervisa la inteligencia o resultados producidos por las máquinas antes de tomar alguna acción, es decir que los sistemas funcionan como soporte a la toma decisiones. En el nivel 2, se deposita cierto nivel de confianza en que las máquinas realicen acciones basadas en su propia inteligencia, sin embargo, estas acciones pueden ser revertidas o corregidas por los humanos. Algunos ejemplos de este nivel son los sistemas de detección de spam para predecir que correos electrónicos son enviados a la carpeta de spam. En el nivel 3 se confía plenamente en que las máquinas realicen acciones sin necesidad de intervención o corrección humana. Sistemas de recomendación pueden funcionar en el nivel 3, si bien la mayoría de ellos operan en el nivel 2 (Otero & Peter, 2015).

Otro concepto importante en este sentido es el proceso de producción de inteligencia, donde para los sistemas de Big Data ocurre principalmente en dos formas de procesamiento, de uno o de varios saltos. En el procesamiento de un solo salto, los datos son procesados directamente por un solo componente y, a partir de eso, se produce toda la inteligencia procesable o accionable necesaria y se prepara para presentar a su consumidor, como muestra la Figura 1 (Otero & Peter, 2015).

El procesamiento multisalto, implica interdependencias entre los componentes los cuales son necesarios para extraer inteligencia accionable detallada. En este procesamiento, la salida de un componente es la entrada de otro. La Figura 2 presenta una visión conceptual del software de Big Data que se ejecuta bajo este tipo de procesamiento y es capaz de proporcionar varios niveles de inteligencia procesable. En cada salto, el sistema produce inteligencia que es consumida por otros componentes y, al final, agrega y correlaciona los datos para proporcionar inteligencia que pueda ser utilizada por los consumidores (Otero & Peter, 2015).

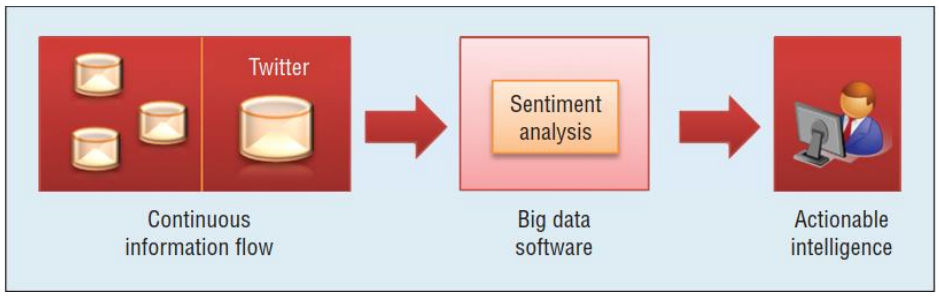

Figura 1 Procesamiento de flujo de información en un solo salto (Otero & Peter, 2015)

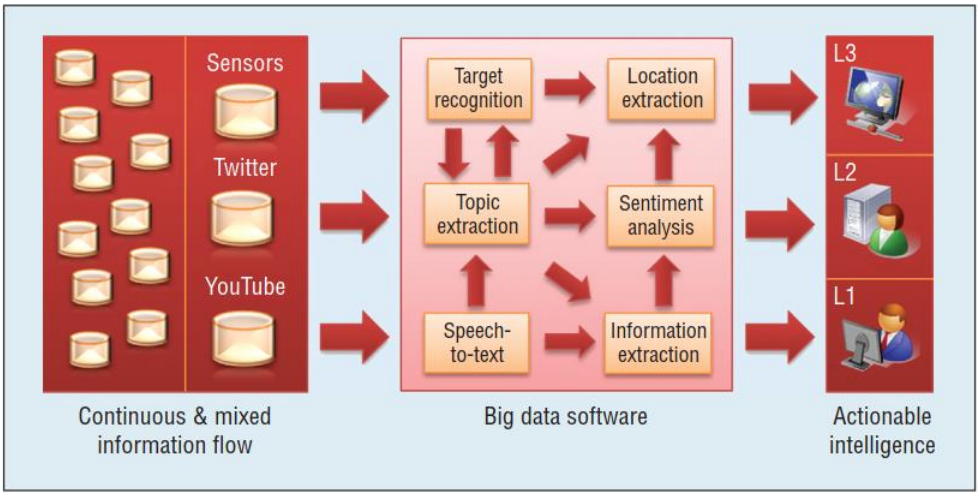

Figura 2 Procesamiento multisalto del flujo de información continua (Otero & Peter, 2015)

2.3 Aprendizaje Automático

Internet y la digitalización dieron paso a un volumen en constante crecimiento de datos estructurados y no estructurados que deben de utilizarse para análisis. En este sentido, el aprendizaje automático (del inglés, *machine learning*) es la tecnología que permite sacar provecho del conocimiento proveniente de los datos disponibles (Alzubi, Nayyar, & Kumar, 2018).

En el año 1959, Arthur Samuel menciona el término aprendizaje automático en un artículo con el título "*Some Studies in Machine Learning Using the Game of Checkers"* publicado en el *IBM Journal of Research*[7], donde se presenta al aprendizaje automático como un subcampo de las ciencias de la computación que dota a las computadoras de la posibilidad de aprender sin ser programado explícitamente (Theobald, 2017). No fue la primera publicación en utilizar el término, pero Arthur Samuel es considerado como la primera persona en precisar al aprendizaje automático de la forma que hoy se conoce (Theobald, 2017).

El aprendizaje como proceso genérico se trata de adquirir nuevos comportamientos, valores, conocimientos y habilidades o de modificar existentes. Para los humanos la forma natural de adquirir conocimientos es aprender de la experiencia, mientras que las máquinas lo realizan a través de los datos (Alzubi et al., 2018). Entonces es posible definir el aprendizaje automático como la capacidad de un sistema de aprender de los datos, en lugar de aprender mediante la programación explícita (IBM, s.f.), como una categoría de la inteligencia artificial que permite a las computadoras aprender por sí mismas (Alzubi et al., 2018). Es posible

[7] https://www.research.ibm.com/journal/

también definirlo como el proceso de utilización de modelos matemáticos con el objetivo que una computadora aprenda sin introducción directa (Microsoft, s.f.).

Antes de poder afrontar un problema, primero se debe poder categorizar y clasificar. Estos pueden ser agrupados en problemas de clasificación, donde se responde a la pregunta ¿es esto A o B?, problemas de detección de anomalía, problemas de regresión, donde se buscar responde cuanto o cuantos, problemas de agrupamiento, donde se busca agrupar basado en la similitud de la estructura de la información. Por último, en problemas de aprendizaje por refuerzo, donde se intenta tomar una decisión o realizar una acción basado en el aprendizaje de experiencias anteriores (Alzubi et al., 2018).

Independientemente del algoritmo utilizado para resolver un problema, el aprendizaje automático puede ser visto como un modelo genérico, el cual consta de seis componentes, recolección y preparación de los datos, selección de las características relevantes, elección del algoritmo, selección de modelos y parámetros, entrenamiento y evaluación de los resultados (Alzubi et al., 2018).

Ahora bien, dependiendo de los algoritmos a aplicar y del problema a resolver, es posible dividir el aprendizaje automático en cuatro categorías generales, aprendizaje supervisado (del inglés, *supervised learning*), aprendizaje no supervisado (del inglés, *unsupervised learning*), aprendizaje de refuerzo (del inglés, *reinforcement learning*) (Mahesh, 2020) (Theobald, 2017) y aprendizaje profundo (del inglés, *deep learning*) (Benuwa, Zhan, Ghansah, Wornyo, & Banaseka, 2016).

En el aprendizaje supervisado el conjunto de datos de ingreso es provisto con las salidas correcta. Basado en un conjunto de datos de entrenamiento, el algoritmo aprende a responder con la mayor exactitud posible mediante la comparación de las entradas con sus salidas. Este procedimiento es conocido también como aprendizaje mediante ejemplos (Alzubi et al., 2018). Este tipo de algoritmos necesita asistencia externa. Los datos de ingreso son divididos en dos conjuntos, datos de prueba y datos de entrenamiento, estos últimos tienen una variable de salida que debe ser predicha o clasificada (Mahesh, 2020). En la Figura 3, se muestra un flujo de trabajo de un algoritmo de aprendizaje supervisado.

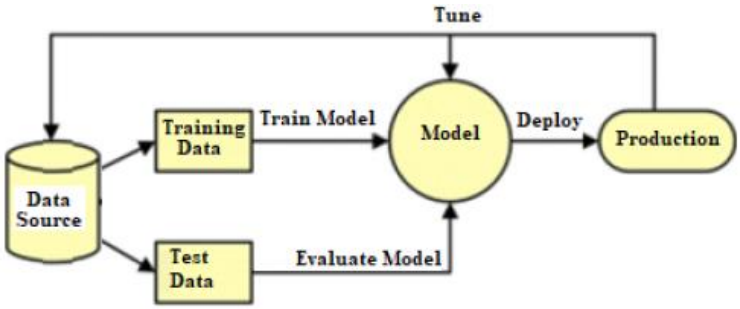

Figura 3 Flujo de trabajo de aprendizaje supervisado (Mahesh, 2020)

El aprendizaje no supervisado se trata del reconocimiento de patrones existentes no identificados en los datos con el objetivo de extraer reglas de ellos. Los datos que se ingresan no están clasificados (Alzubi et al., 2018). No hay respuestas correctas como en el caso del supervisado y cuando nuevos datos son introducidos, el algoritmo utiliza lo aprendido previamente para reconocer la clase de datos (Mahesh, 2020). No hay asistencia de forma externa. La Figura 4 muestra un flujo de trabajo de un algoritmo de aprendizaje no supervisado.

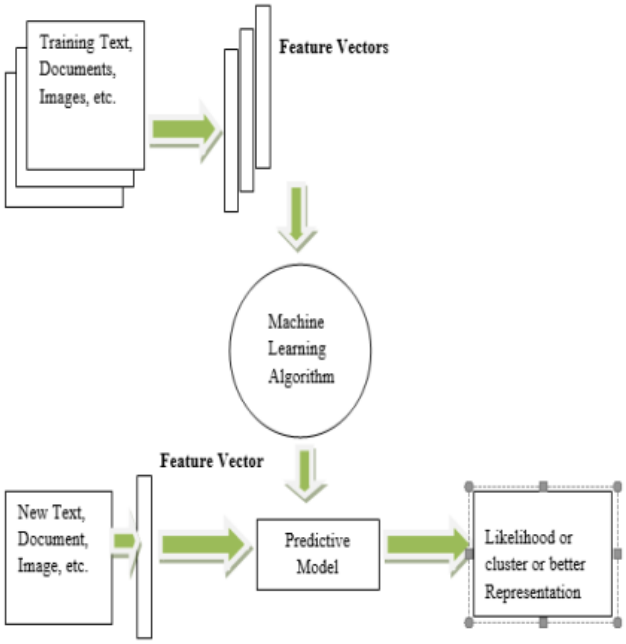

Figura 4 Flujo de trabajo aprendizaje no supervisado (Alzubi et al. 2018)

El aprendizaje de refuerzo, Figura 5, es un área del aprendizaje automático la cual se enfoca en la forma que los agentes de software deben tomar medidas en un ambiente para maximizar alguna noción de la recompensa acumulativa (Mahesh, 2020). El algoritmo provee solamente una respuesta la cual dice si la salida es correcta o no, por lo que tiene que explorar y descartar varias posibilidades para obtener la salida correcta, basándose en la mejora continua del modelo mediante la obtención de retroalimentación de las iteraciones previas (Alzubi et al., 2018) (Theobald, 2017).

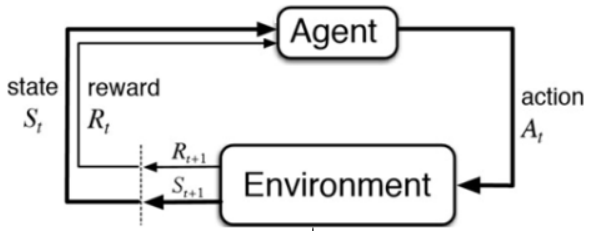

Figura 5 Aprendizaje de refuerzo (Mahesh, 2020)

El aprendizaje profundo es una rama del aprendizaje automático, la cual utiliza múltiples capas de procesamiento con estructuras completas con el objetivo de modelar abstracciones de alto nivel en los datos utilizando una serie de algoritmos (Hinton, Osindero, & Teh, 2006) (Bengio Y. , 2009). Estos algoritmos están basados en una representación distribuida, donde los datos observados son generados por la interacción de factores organizados en capas, donde cada capa utiliza la salida generada por una capa previa como datos de ingreso. Los algoritmos utilizados pueden ser supervisados o no supervisados (Benuwa et al., 2016).

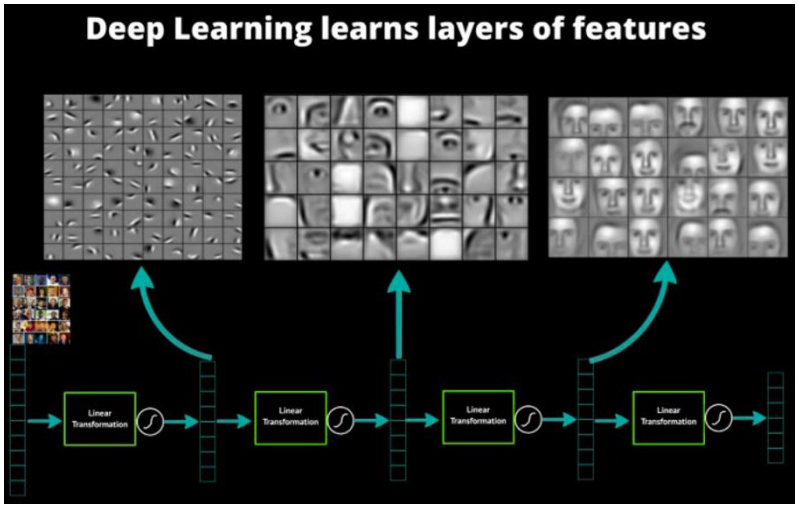

Figura 6 Arquitectura aprendizaje profundo (Bengio, Deng, Larochelle, Lee, & Salakhutdinov, 2013)

Algunas de las aplicaciones de aprendizaje automático utilizadas en el día a día son, reconocimiento de voz y facial, vehículos autónomos, filtros antispam, robótica e inteligencia artificial, redes sociales, medicina, detección de fraudes en tarjetas de crédito, entre otras (Alzubi et al., 2018).

2.4 Testing

La evaluación, validación y verificación de cumplimiento de los requerimientos definidos originalmente por el usuario en un sistema, se define como el proceso de testing o de

pruebas (Jamil, Arif, Ahmad, & Awang Abubakar, 2016). El proceso de pruebas evalúa y verifica que un producto de software haga lo que se supone debe realizar (IBM, 2020) y es un componente importante para el aseguramiento de la calidad (Jamil, Arif, Ahmad, & Awang Abubakar, 2016). Se estima que, en el año 2016, las fallas de software le costaron a la economía de los Estados Unidos $1.1 billones de dólares en activos (Siroky, 2017).

El proceso de pruebas forma parte del ciclo de vida de desarrollo de software (SDLC, del inglés *Software Development Lifecycle*) y como se muestra en la Figura 7 es posible dividirlo en cuatros niveles, pruebas unitarias (del inglés, *unit tests*), pruebas de integración (del inglés, *integration tests*), pruebas de sistema (del inglés, *system tests*) y pruebas de aceptación (del inglés, *acceptance tests*) (Honest, 2019).

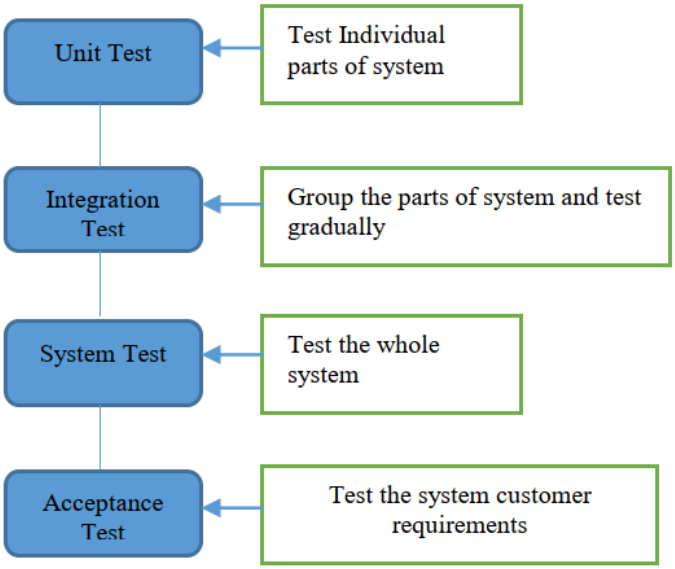

Figura 7 Niveles de pruebas (Honest, 2019)

El nivel de pruebas unitarias separa los componentes individuales para realizar pruebas de forma aislada, con el objetivo de corroborar que cada módulo funcione de la forma esperada sin ninguna dependencia (Honest, 2019) (Jamil, Arif, Ahmad, & Awang Abubakar, 2016).

En el nivel de pruebas de integración los componentes individuales son probados como grupo, para verificar la existencia de fallas durante la interacción entre estos (Honest, 2019). Este nivel permite probar y validar la funcionalidad deseada cuando los módulos son construidos para interactuar entre sí (Lawanna, 2012).

El nivel de pruebas de sistema es el proceso por el cual se realiza el testeo de un sistema de software completo e integrado (Honest, 2019), donde no sólo se prueban los requerimientos

funcionales del sistema, sino también los no funcionales como la seguridad y confiabilidad (Sneha & Malle, 2017).

En el nivel de pruebas de aceptación el sistema se evalúa para comprobar que cumple con los requerimientos de negocio y es aceptable para la entrega (Honest, 2019). Las mismas son realizadas cuando el software pasa a manos de los usuarios (Sneha & Malle, 2017).

Las pruebas pueden comenzar desde la fase de reunión de requerimientos del desarrollo de software hasta luego del despliegue de este. No existe un momento determinado en el cual parar las pruebas, es un proceso que no tiene fin que muestra la presencia de errores y no la ausencia de estos. Igualmente, las pruebas pueden detenerse cuando se alcance algún plazo determinado, cuando se completen ciertos casos de prueba o ante alguna decisión de negocio o gerencial (Kale, Bandal, & Chaudhari, 2019) (Anwar & Kar, 2019).

Igualmente como muestra la Figura 8 puede definirse un proceso de ciclo de vida de pruebas, pudiendo constar de las siguientes etapas, análisis de requerimientos, planeamiento de pruebas, armado de casos de prueba, ejecución de casos de prueba, evaluación de resultados e informes, re-ejecución de pruebas sobre defecto, ejecución de pruebas de regresión y actividad de cierre de pruebas (Kale, Bandal, & Chaudhari, 2019) (Jamil, Arif, Ahmad, & Awang Abubakar, 2016) (De Nicola et al., 2005).

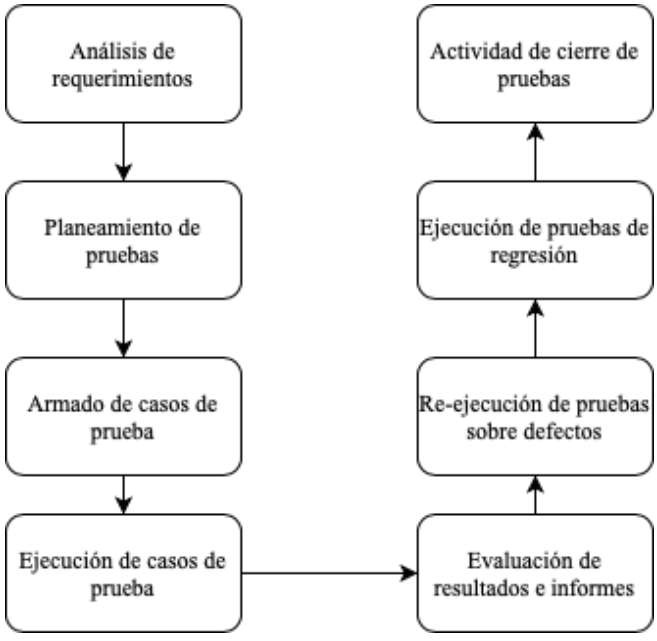

Figura 8 Proceso de ciclo de vida de pruebas

Para la ejecución de las pruebas, existen dos tipos de estrategias principales, pruebas de caja blanca (del inglés, *white-box testing*) y pruebas de caja negra (del inglés, *black-box testing*). Las primeras se encargan de la verificación de la lógica interna y estructura del código.

Son utilizadas principalmente para la detección de errores lógicos dentro del código del programa. Mayormente es utilizado a nivel de unidad, pero puede ser implementado en el nivel de integración o de sistema. Existen diferentes acercamientos de pruebas de caja blanca, estos son pruebas de flujo de datos (del inglés, *data flow testing*), pruebas de bucle (del inglés, *loop testing*), pruebas de ramificación o de ramas (del inglés, *branch testing*) pruebas de control de flujo (del inglés, *control flow testing*) y pruebas de ruta base (del inglés, *basis path testing*) (Anwar & Kar, 2019) (Gupta, 2014) (Lawanna, 2012). Si bien expone errores que están en el código puede que no identifique requerimientos que no fueron implementados o que faltan de acuerdo con las especificaciones de usuario (Gupta, 2014).

Las pruebas de caja negra validan que cada entrada sea debidamente aceptada y produzca la salida esperada. Se encargan de los aspectos fundamentales de un sistema, sus especificaciones y funcionalidad sin tener en cuenta la implementación interna. Principalmente es utilizado para determinar si el programa cumple o no con los requisitos especificados por el usuario. Existen diferentes acercamientos para realizar este tipo de pruebas entre los que se encuentran, particionamiento de equivalencia (del inglés, *equivalence partitioning*), análisis de valor límite (del inglés, *boundary value analysis*), grafo de causa y efecto (del inglés, *cause-effect graph*), pruebas *fuzz* (del inglés, *fuzz testing*), pruebas de regresión (del inglés, *regression testing*), pruebas de patrones (del inglés, *pattern testing*), pruebas de matrices ortogonales (del inglés, *orthogonal array testing*), pruebas por parejas (del inglés, *all pair testing*) y pruebas de transición de estado (del inglés, *state transition testing*) (Anwar & Kar, 2019) (Jamil et al., 2016) (Lawanna, 2012).

Por otro lado, existe un tercer acercamiento llamado pruebas de caja gris (del inglés, *grey-box testing*), el cual es una combinación de las pruebas de caja blanca y de caja negra. Tiene como objetivo probar una aplicación o sistema teniendo solamente conocimiento parcial de la estructura interna y un conocimiento de las especificaciones o requerimientos del sistema (De Nicola et al., 2005) (Anwar & Kar, 2019).

Los diferentes tipos de pruebas que se llevan a cabo durante el ciclo de vida del software pueden ejecutarse de forma manual, donde la persona encargada de la prueba compara los resultados obtenidos con los esperados.

Por otro lado, también pueden ser ejecutados de forma automatizada cuando se requiere repetir de forma sistemática la ejecución de casos (De Nicola et al., 2005). Para esto se hace uso de un software particular para llevar a cabo el proceso de pruebas, así como hace la comparación de los resultados reales con los resultados esperados, siendo este acercamiento

efectivo en términos de tiempo (Jamil et al., 2016) ya que el proceso de pruebas de software es costoso.

La ejecución de pruebas es una actividad que requiere hasta el 50% del costo de desarrollo e incluso puede incrementarse en aplicaciones de misión crítica. Es por eso que la automatización de las pruebas puede conducir a una reducción del costo, ayudando a minimizar los errores humanos, reducción de errores por omisión, eliminar parte de la variación en la calidad de las pruebas causada por las diferencias en las habilidades o experiencia de las personas que las ejecutan, permite también que las pruebas de regresión puedan llevarse a cabo de una manera más fácil (Ammann & Offutt, 2008).

Capítulo 3

Testing en Big Data y aprendizaje automático

3.1 Introducción

Las técnicas habituales de verificación y validación de software pueden ser utilizadas a nivel sistema en Big Data, sin embargo, estos enfoques no son lo suficientemente buenos para garantizar la calidad de los componentes esenciales de este tipo de aplicaciones: Big Data, algoritmos de aprendizaje automático y componentes de software no testeables (Ding, Hu, & Gudivada, 2017).

El proceso de pruebas en las aplicaciones de Big Data es fundamental debido a la naturaleza dinámica de estas. Un proceso deficiente puede llegar a tener un impacto negativo en las organizaciones a la hora de obtener datos precisos para la toma de decisiones (Punn, Agarwak, Syafrullah, & Adiyarta, 2019). Para garantizar la fiabilidad y la alta disponibilidad, las aplicaciones y la infraestructura de Big Data tienen que ser validadas y verificadas, y para ello existen diversas herramientas como Apache Hadoop[8] o bases de datos NoSQL[9] como MongoDb[10]. Sin embargo las características propias de Big Data generan nuevos desafíos, por ejemplo, la selección y la validación de los datos son fundamentales para la eficacia y el rendimiento del análisis de estos, pero el gran volumen y la variedad crean un gran desafío para ello (Ding, Hu, & Gudivada, 2017).

Este ese aspecto, los resultados de un procesamiento masivo de datos no son prácticos de verificar para los humanos y puede que sea imposible. La complejidad de las herramientas de análisis de datos y los algoritmos de aprendizaje automático utilizados para procesamiento son difíciles de validar. Muchos defectos pueden pasar desapercibidos durante el proceso de pruebas y las aplicaciones que utilizan Big Data pueden volverse problemáticas a la hora de testear, ya que pertenecen a la clase de software no testeable, es decir no existe un oráculo o procedimiento que permita la verificación y validación de los resultados (Otero & Peter, 2015) (Ding, Hu, & Gudivada, 2017).

Como menciona Segura en su ejemplo sobre la ejecución de pruebas en el sitio de reservas de alojamiento Booking.com[11], ¿cómo podría decirse que un resultado es correcto cuando se realiza una búsqueda que retorna más de 7000 resultados? ¿Puede haber un dato

[8] https://hadoop.apache.org/
[9] https://www.mongodb.com/es/nosql-explained
[10] https://www.mongodb.com/
[11] https://www.booking.com/

faltante en esa salida o alguno que no cumpla con la condición de búsqueda deseada? (Segura et al., 2020).

Lo mismo ocurre en el caso que platean, Otero & Peter, para el análisis de sentimiento de los textos escritos en Twitter, donde como entrada se tiene un flujo de datos de la red social y como salida una acción basada en la predicción de sentimiento que puede ser positivo, neutral o negativo (Otero & Peter, 2015). Las predicciones podrían ser muy precisas, pero no es posible garantizar la fiabilidad de estas y en algunos casos sería incluso difícil para los humanos detectar el sentimiento en un texto.

Los casos mencionados son ejemplos que enfrentan al problema del oráculo y al problema del conjunto de pruebas fiable. El primero refiere a situaciones en las que es extremadamente difícil, o imposible, verificar el resultado de un caso de prueba determinado. El segundo problema, dado que normalmente no es posible ejecutar exhaustivamente todos los casos de prueba posibles, es un reto seleccionar eficazmente un subconjunto de casos de prueba con capacidad para determinar cuan correcto es el programa bajo test (Chen et al., 2018).

En este sentido el acercamiento de testing metamórfico (Chen, Cheung, & Yiu, 1998) puede ser utilizado para la generación de casos de pruebas como para la verificación de los resultados de estas, abordando así los dos problemas fundamentales al momento de la ejecución de pruebas.

La siguiente sección presentará y analizará la propuesta metamórfica, las relaciones metamórficas como elemento central y los desafíos que lleva la implementación de esta técnica y su aplicación en sistemas de Big Data y aprendizaje automático.

3.2 Testing Metamórfico

Como se menciona en la sección previa, el testing metamórfico (MT) puede ser utilizado tanto para la generación de casos de pruebas como para la verificación de los resultados.

Este tipo de pruebas se componen de un conjunto de relaciones metamórficas (Chen et al., 2018), de ahora en más MRs (del inglés, metamorphic *relations*). Estas relaciones son propiedades necesarias de la función, el algoritmo o la funcionalidad del programa previsto en relación con múltiples entradas y sus salidas esperadas. Una propiedad necesaria de un algoritmo es una condición que se puede deducir lógicamente del algoritmo (Chen et al., 2018) (Segura et al., 2020). Cuando se aplica MT, algunas de las entradas son generadas como primeros casos de prueba o también denominados casos de prueba de origen (Chen et al., 2018),

a partir de los cuales se generan nuevos casos, también llamados casos de seguimiento (Chen et al., 2018), utilizando las relaciones metamórficas. En este sentido una relación metamórfica transforma los casos de prueba en nuevos casos de seguimiento.

Cuando se aplica está técnica de testing se verifican los casos de prueba de origen y de seguimiento, así como los resultados, en relación con la MR correspondiente. Si el comportamiento del programa a través de estos casos de prueba de origen y de seguimiento rompe la relación metamórfica, el programa debe ser defectuoso (Chen et al., 2018) (Segura et al., 2020). Este acercamiento difiere de las formas tradicionales de ejecución de pruebas donde se verifican los resultados de cada caso de prueba individual.

Supóngase el siguiente ejemplo dado por Chen et al., donde un algoritmo *f* calcula el camino más corto para un grafo no dirigido G, donde un programa P implementa dicho algoritmo. Para cualquier punto *a* y *b* en un G grande, puede ser muy difícil verificar si el resultado calculado de P dadas las entradas G, *a*, y *b* es realmente el camino más corto entre *a* y *b*. Una posible forma de verificar el resultado es generar todos los posibles caminos de *a* a *b*, y luego comprobar con ellos si P (G, *a*, *b*) es realmente el camino más corto (Chen et al., 2018).

Puede que no sea factible en la práctica generar todos los caminos posibles de *a* a *b*, sobre todo si el número de caminos posible crece en forma exponencial con el número de vértices. Pero por otro lado es posible hacer uso de algunas propiedades para verificar parcialmente el resultado que genera el programa. Por ejemplo, se puede generar una relación metamórfica de la siguiente propiedad, si se intercambian los vértices a y b, la longitud del camino más corto no cambia, $| f(G, b, a) | = | f(G, a, b) |$. Esta relación necesita dos ejecuciones de prueba, una con el caso de prueba de origen (G, a, b) y otra con el caso de prueba de seguimiento (G, b, a). Se verifican los resultados de las ejecuciones con respecto a la MR: donde se comprueba si la relación | P(G, *b*, *a*) | = | P(G, *a*, *b*) | se satisface o se viola. Si se detecta una violación, es posible decir que el programa P es defectuoso (Chen et al., 2018).

Para avanzar con otro ejemplo que pueda aplicarse en un sitio web para usuarios finales, supongamos que se desea realizar pruebas en el sitio de comercio electrónico Mercado Libre[12], en este sentido se desea comprobar que los resultados de búsqueda de automóviles sean correctos. Para ello se ejecuta una búsqueda de autos usados el cual retorna 92559 resultados como se muestra en la Figura 9, tal como se pregunta Segura et al., 2020 en el ejemplo acerca de Booking, ¿estos resultados son correctos? ¿todos los autos son usados?

[12] https://www.mercadolibre.com.ar/

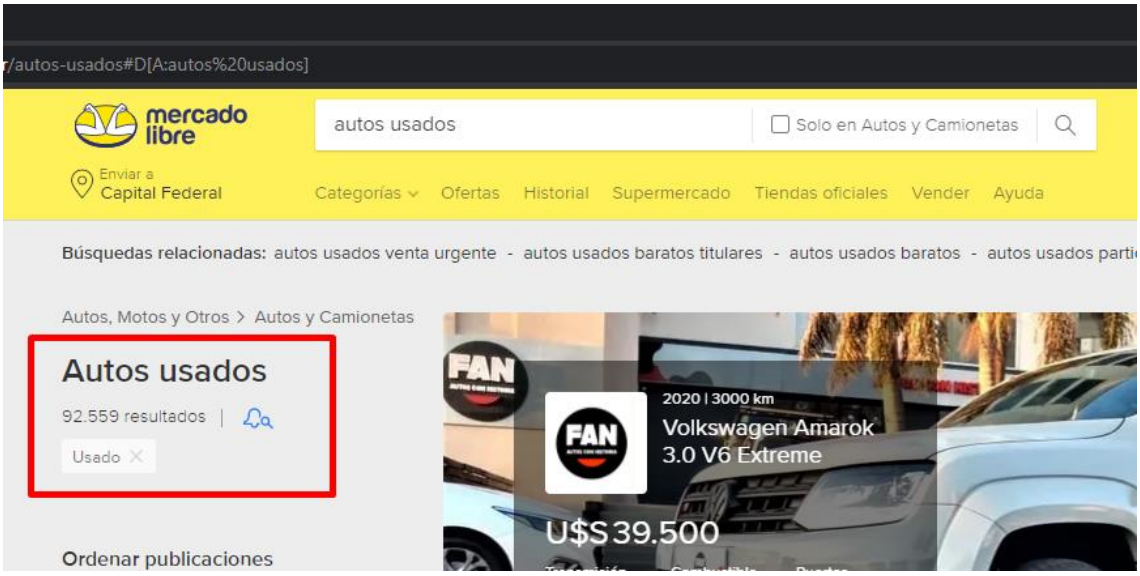

Figura 9 Resultado de búsqueda Mercado Libre

Cómo se menciona en el ejemplo previo, puede no ser posible o factible analizar cada uno de los resultados que se obtienen, sobre todo si continúan creciendo en número a medida que nuevas publicaciones de autos usados vayan siendo agregadas, pero es posible analizar algunas relaciones que permitan verificar el resultado. Supóngase entonces, que se genera la primera relación metamórfica donde al resultado de búsqueda obtenido anteriormente, conjunto A, se le aplica un filtro de año, en este caso modelos a partir del año 2018. El resultado de esta búsqueda (*x)*, ahora es un caso de seguimiento o nuevo caso de prueba y debería ser un subconjunto del resultado anterior.

Se aplica un ordenamiento por mayor precio, subconjunto B, y luego se vuelve a ordenar por menor precio subconjunto C. De aquí podemos sacar una nueva propiedad donde los subconjuntos B y C deberían tener la misma cantidad de resultados.

Verificando las ejecuciones con las MR definidas, es posible decir que si $x \varepsilon A$, $B = C$ y $C = B$, las relaciones se satisfacen. Si se detecta que alguna relación es quebrantada, en este caso las búsquedas en el sitio son defectuosas con respecto al dominio del problema.

Respecto a los ejemplos mencionados, es posible utilizar las relaciones entre las salidas esperadas y las múltiples entradas para determinar si un programa es defectuoso, incluso ante la no posibilidad de determinar la exactitud de la salida de una entrada individual.

No hay un ámbito específico para la aplicación de las pruebas metamórficas, las mismas se han utilizado en sistemas bioinformáticos, sistemas embebidos, solucionadores de ecuaciones diferenciales parciales, servicios web y web APIs, compiladores, bases de datos, funciones de búsqueda en línea y motores de búsqueda (Segura et al., 2020) (Chen et al., 2018) (Ding, Hu, & Gudivada, 2017) (Zhou, Xiang, & Chen, 2016).

3.2.1 Definiciones formales

Durante las secciones previas de este capítulo, se ha provisto una definición no formal de las pruebas y relaciones metamórficas. Es atinado realizar a continuación las definiciones formales respecto a esta metodología de pruebas.

> Definición 1 – Relación metamórfica. Sea *f* una función objetivo o un algoritmo. Una relación metamórfica (MR) es una propiedad necesaria de *f* sobre una secuencia de dos o más entradas $\langle x_1, x_2, \ldots, x_n \rangle$, donde $n \geqslant 2$, y sus correspondientes salidas $\langle f(x_1), f(x_2), \ldots, f(x_n) \rangle$. Puede expresarse como una relación $R \subseteq X^n \times Y^n$, donde $\subseteq$ denota la relación de subconjunto, y X^n e Y^n son los productos cartesianos de n espacios de entrada y n de salida, respectivamente. Siguiendo la práctica informal habitual, podemos escribir simplemente $R(x_1, x_2, \ldots, x_n, f(x_1), f(x_1), \ldots, f(x_n))$ para indicar que $\langle x_1, x_2, \ldots, x_n, f(x_1), f(x_2), \ldots, f(x_n) \rangle \in R$. (Chen et al., 2018, pág. 4)

> Definición 2 - Entrada de origen y entrada de seguimiento. Consideremos una MR $R(x_1, x_2, ..., x_n, f(x_1), f(x_2), ..., f(x_n))$. Supóngase que cada x_j $(j = k + 1, k + 2, ..., n)$ es construido a partir de $\langle x_1, x_2, \ldots, x_k, f(x_1), f(x_2), \ldots, f(x_k) \rangle$ según R. Para cualquier $i = 1, 2, ..., k$, nos referimos a x_i como una entrada de origen. Para cualquier $j = k + 1, k + 2, ..., n$, nos referimos a x_j como una entrada de seguimiento. En otras palabras, para una R dada, si se especifican todas las entradas de origen x_i $(i = 1,2, ..., k)$, las entradas de seguimiento x_j $(j = k + 1, k + 2, ..., n)$ pueden construirse a partir de las entradas de origen y si es necesario, de sus correspondientes salidas. (Chen et al., 2018, pág. 5)

> Definición 3 - Grupo Metamórfico de Entradas (MG). Considere un MR $R(x_1, x_2, ..., x_n, f(x_1), f(x_2), ..., f(x_n))$. La secuencia de entradas $\langle x_1, x_2, ..., x_n \rangle$ se define como un grupo metamórfico (MG) de entradas para la MR. Más concretamente, el MG es la secuencia de entradas de origen $\langle x_1, x_2, ..., x_k \rangle$ y las entradas de seguimiento $\langle x_k+1, x_k+2, ..., x_n \rangle$ relacionadas con R. (Chen et al., 2018, pág. 5)

> Definición 4 - Pruebas Metamórficas (MT). Sea *P* una implementación de un algoritmo objetivo *f*. Para una MR *R* supóngase que tenemos $R(x_1, x_2, ..., x_n, f(x_1), f(x_2), ..., f$

(x_n)). La prueba metamórfica (MT) basada en esta MR para *P* implica los siguientes pasos: (1) Definir *R*' sustituyendo *f* por *P* en *R*, (2) Dada una secuencia de casos de prueba fuente ⟨x_1, x_2, ..., x_k⟩, ejecutarlos para obtener sus respectivas salidas ⟨*P* (x_1), *P* (x_2), ..., *P* (x_k)⟩. Construir y ejecutar una secuencia de casos de seguimiento ⟨x_k+1, x_k+2, ..., x_n⟩ según *R*' y obtener sus respectivas salidas ⟨*P* (x_k+1), *P* (x_k+2), ..., *P* (x_n)⟩. (3) Examinar los resultados con referencia a *R′*. Si *R′* no se satisface, entonces este MR ha revelado que *P* es defectuoso. (Chen et al., 2018, pág. 5)

3.2.2 Ventajas del acercamiento metamórfico

Algunas de las ventajas que pueden enumerarse del acercamiento metamórfico son las siguientes:

- Simplicidad en el concepto (Chen et al., 2018): Testers pueden aprender a utilizar el acercamiento metamórfico e identificar relaciones incluso sin tener demasiada experiencia. Esto puede observarse en un estudio previo, (Liu, Kuo, Towey, & Chen, 2014), donde se reclutó estudiantes de grado y posgrado para identificar relaciones metamorficas. Aunque algunos de ellos que participaron en el estudio ya habían aprendido algunos conceptos básicos de pruebas de software, no tenían experiencia practica en testing ni conocimiento previo de las pruebas metamórficas.
- Aplicación directa (Chen et al., 2018): Independientemente de la dificultad de automatización del proceso de implementación e identificación de relaciones metamórficas, de acuerdo a la definición 4 provista en la sección 3.2.1, a modo general la prueba metamórfica implica realización de tres pasos.
- Facilidad de automatización dada la disponibilidad de las MRs (Chen et al., 2018): Sin tener en cuenta el proceso de identificación de MR, el cual puede ser la única parte que podría no ser totalmente automatizada, no debería ser difícil automatizar los principales pasos en las pruebas metamorficas, incluyendo la generación, ejecución y verificación de casos de prueba. Como pueden verse algunos estudios (Segura et al., 2020) (Jarman, Zhou, & Chen, 2017), las automatización de los casos de prueba puede ser realizada con técnicas existentes utilizando librerías que pueden no ser específicas para esta metodología.
- Bajo costo (Chen et al., 2018): Los procesos manuales son necesarios tanto en técnicas tradicionales de pruebas como en el acercamiento metamórfico, por ejemplo, es posible enumerar actividades como el análisis de requisitos, construcción de modelos formales

o generación de casos. En este sentido, los casos de prueba de seguimiento para las pruebas metamórficas se generan de forma sencilla mediante transformaciones según las relaciones establecidas y aunque la verificación de los resultados de las pruebas implica cotejar las salidas con las MRs, la sobrecarga asociada es relativamente baja en comparación con el costo de la verificación de los resultados cuando existe el problema del oráculo.

Es conveniente resaltar, que la técnica metamórfica puede aplicarse con un conjunto de pruebas de cualquier tamaño, independientemente de la escala y la complejidad del programa sometido a prueba. El tamaño del conjunto de pruebas no afecta a la aplicación de las pruebas metamorficas (Chen et al., 2018). Por otro lado, se destaca que este acercamiento es agnóstico al lenguaje de programación utilizado.

3.2.3 Enfoque para la selección de relaciones metamórficas

La calidad de las relaciones metamórficas identificadas tiene una fuerte relación en la eficacia de las pruebas metamórficas. Las mismas son clave en este tipo de pruebas y la identificación de estas es un paso crítico (Segura et al., 2020) (Chen et al., 2018) (Ding, Hu, & Gudivada, 2017). Desde el punto de vista técnico, las MRs forman parte de las especificaciones y, por tanto, intuitivamente, deberían poder identificarse a partir de ellas. Sin embargo, a pesar de su enfoque sistemático, sigue dependiendo en cierta medida de los conocimientos y la experiencia de las personas que ejecutan las pruebas para identificar las relaciones (Chen et al., 2018).

Existen dos enfoques que pueden ser descriptos para la identificación de las relaciones, el basado en las entradas (del inglés, *input-drive approach*) y el basado en las salidas (del inglés, *output-drive approach*) (Segura et al., 2020). En el acercamiento basado en las entradas, se modifican las entradas del programa de tal forma, que deberían producir salidas con cambios esperados. En el acercamiento basado en las salidas, se proponen a partir de las posibles relaciones entre los resultados que se encuentran típicamente en el dominio objetivo. Luego se piensan en los tipos de cambio en las entradas del programa que llevarían a la satisfacción de la relación esperada entre los resultados (Segura et al., 2020).

Algunos autores mencionan a su vez la importancia de encontrar la forma de perfeccionar o refinar las relaciones metamórficas y las pruebas a partir de los resultados y de la evaluación de estas, proponiendo un enfoque iterativo de pruebas metamórficas (Ding, Hu, & Gudivada, 2017) (Ding, Zhang, & Hu, 2016).

3.2.4 Conceptos frecuentemente malinterpretados en pruebas metamórficas

Chen et al., señalan en su investigación (Chen et al., 2018), que detectaron conceptos que frecuentemente son malinterpretados por los lectores, revisores y profesionales del software. El primer concepto mencionado es que no todas las propiedades necesarias son relaciones metamórficas de un algoritmo dado, por ejemplo, aunque $-1 \leq \text{sen}(x) \leq 1$ es una propiedad necesaria de la función seno, sólo implica una única instancia de la entrada y, por tanto, no puede considerarse una MR. Otro concepto es que no todas las relaciones metamórficas se separan en sub-relaciones de sólo entrada y de sólo salida. Tomando el ejemplo utilizado en la definición de pruebas metamórficas (véase, Chen et al. 2018), para el cálculo del camino más corto en un grafo no dirigido, es posible definir una segunda MR donde |P (*G*, *a*, *c*)| + |P (*G*, *c*, *b*)| = |P (*G* ,*a*, *b*)| siendo *c* el nodo que se presenta en el camino más corto de *a* a *b.* En dicha relación, los casos de prueba de seguimiento (*G*, *a*, *c*) y (*G*, *c*, *b*) dependen de la salida del caso de prueba de origen (*G*, *a*, *b*). Igualmente cumple la definición 1 (Chen et al., 2018, pág. 4) aunque es diferente a la primera que tiene la forma R_{entrada} y R_{salida}.

El tercer concepto indica que no todas las MRs son relaciones de igualdad, es decir pueden incluir, entre otras, a las relaciones de igualdad. Esto se observa en el ejemplo brindado utilizando el buscador de la plataforma Mercado Libre o en el ejemplo de Booking.com (Segura, Towey, Zhou, & Chen, 2020). Por último, existe una confusión entre los investigadores donde consideran que las pruebas metamórficas sólo se utilizan cuando el software o aplicación no cuenta con un oráculo específico de pruebas, pero también puede aplicarse cuando se dispone de un oráculo utilizable.

3.2.5 Pruebas metamórficas en sistemas de Big Data y aprendizaje automático

Como se menciona en la sección 3.1, la ejecución de pruebas y verificación de resultados en las aplicaciones de Big Data y aprendizaje automático, son complejos de realizar debido a la naturaleza propia de estas tecnologías, las cuales utilizan un procesamiento masivo de datos. Normalmente, la cantidad de datos son tan grandes y complejos que las técnicas tradicionales de pruebas pueden no ser suficientes. El tamaño de los datos, sus diversos tipos y formatos, hacen que el problema del oráculo sea frecuente, lo que convierte las pruebas en un gran reto (Chen et al., 2018).

Los ejemplos mencionados en la sección 3.2, utilizan las relaciones metamórficas para la verificación en las búsquedas realizadas en sitios de comercio electrónico, el mismo

acercamiento puede ser utilizado para la verificación y validación en sistemas de Big Data y aprendizaje automático. Una propuesta inicial de aplicación de MT en este tipo de sistemas es provista por Otero y Peter en su trabajo "*Research Directions for Engineering Big Data Analytics"* (Otero & Peter, 2015), en el cual es posible hacer uso de las relaciones metamórficas para ejecutar un proceso de pruebas, en un procesamiento de fujo de información para el análisis de sentimientos de datos obtenidos de Twitter como muestra la Figura 10.

Basándose en el conocimiento del dominio del problema, se define un vocabulario *v* para las palabras asociadas al sentimiento positivo, donde cada v_i representa una única palabra positiva en un Tweet $v = [v_1, v_2, v_3, \ldots, v_n]$ (Otero & Peter, 2015).

De *v* pueden derivarse las siguientes relaciones metamórficas para verificar la salida del sistema, vocabulario de sinónimos, de antónimos y negaciones respecto a *v*. Con estos pueden crearse cuatro diferentes casos de prueba o seguimiento utilizando Tweets, de tal manera que la clasificación del sentimiento para los mensajes que utilizan cada palabra v_i, debería dar como resultado un sentimiento positivo. Reutilizando los mismos mensajes, pero intercambiando cada palabra v_i con cada sinónimo definido también debería dar como resultado una clasificación positiva del sentimiento. Mientras que los mensajes con palabras de los vocabularios de antónimos y negaciones respecto a *v* deberían dar como resultado una clasificación negativa.

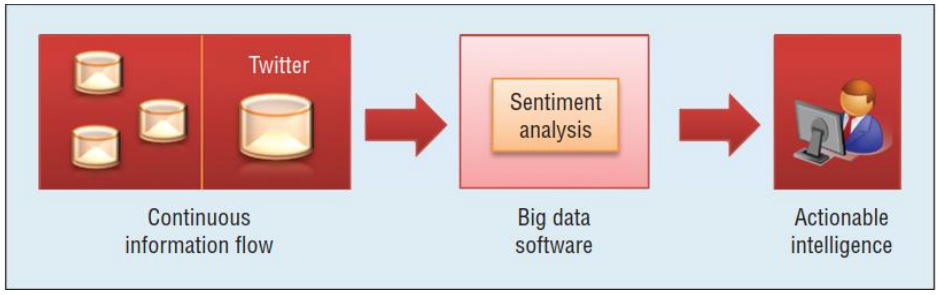

Figura 10 Procesamiento de flujo de información en un solo salto (Otero & Peter, 2015)

Si se detecta una violación de alguna de las relaciones, es posible decir que el programa para el análisis de sentimiento es defectuoso y se descubrirían errores para ese dominio concreto.

Estas técnicas de verificación deben ser examinadas, si tienen éxito, podrían ser empleadas por los especialistas en pruebas sin que necesiten tener experiencia en el campo de Big Data o aprendizaje automático (Otero & Peter, 2015).

En este sentido, si bien el ejemplo está realizado sobre un solo salto, es posible extenderlo sobre un procesamiento de flujo de información continua de diferentes fuentes de

datos con un esquema multisalto y diversas tomas de acción como se representa en la Figura 2.

Por otro lado, puede encontrarse en la investigación realizada por Ding et al. (Ding, Hu, & Gudivada, 2017), una implementación de pruebas metamórficas en sistemas de Big Data y aprendizaje automático en su desarrollado llamado CMA (del inglés, *Cell Morphology Assay).* Este sistema incluye un grupo de herramientas de software científico, algoritmos de aprendizaje automático y un repositorio de imágenes celulares a gran escala. El mismo es utilizado para investigar la clasificación de las células biológicas a partir de la morfología celular que se capta en las imágenes de difracción.

El marco de trabajo para la validación y verificación propuesto por ellos incluye tareas en tres capas. La capa base es la técnica para la selección y validación automatizada de Big Data, la capa intermedia es un enfoque para la verificación y validación de los algoritmos de aprendizaje automático y la capa superior es un enfoque para probar los sistemas de modelado de dominios, las herramientas de análisis de datos y las aplicaciones (Ding, Hu, & Gudivada, 2017). Esto puede observarse en la Figura 11. La validación de los datos, la evaluación de la eficacia de los algoritmos de aprendizaje automático y la comprobación del software científico es soportado por el marco de trabajo de CMA.

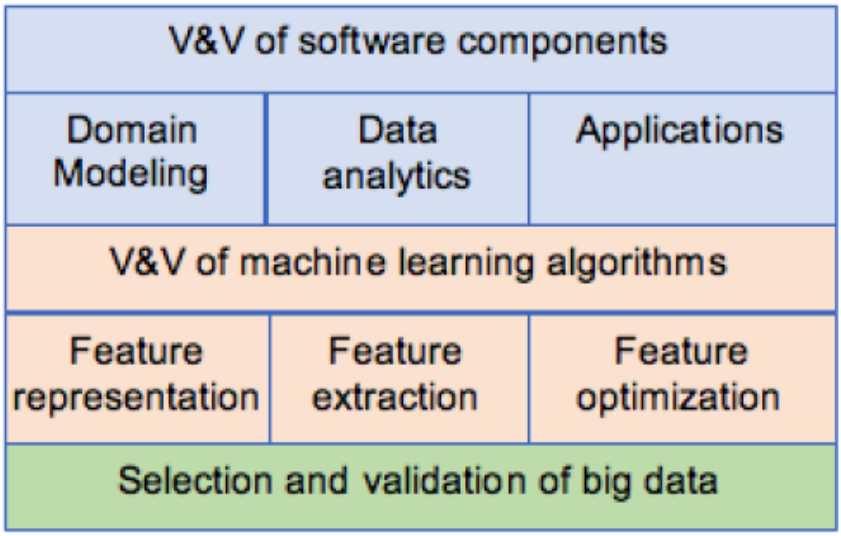

Figura 11 Esquema de validación y verificación de sistemas de Big Data (Ding, Hu, & Gudivada, 2017)

Con el objetivo de analizar el acercamiento utilizado, se hará foco en las pruebas sobre una de las partes del CMA, en este caso en el componente de reconstrucción de estructuras en 3D, el cual se utiliza para procesar secciones de imágenes de una célula y construir su estructura 3D (Ding, Hu, & Gudivada, 2017).

Según Ding, Hu, & Gudivada (2017) la parte más difícil del proyecto es la correcta construcción de las estructuras 3D de las mitocondrias en las células, ya que cada sección de imagen puede incluir muchas mitocondrias las cuales pueden estar muy cercas entre sí. Por lo tanto existe la posibilidad que dos mitocondrias en dos secciones adyacentes podrían estar conectadas incorrectamente, lo cual daría lugar a una estructura 3D errónea.

Realizar la verificación para comprobar la estructura 3D reconstruida comparándola con la célula original es impracticable, ya que la célula está muerta o su estructura 3D ha cambiado mucho durante la reconstrucción. Para realizar la verificación del software, propusieron realizar pruebas metamórficas mediante un acercamiento iterativo. Los tres pasos para la ejecución de las pruebas consisten en el desarrollo inicial de las relaciones metamórficas y las pruebas, evaluación de las MRs y como tercer paso el refinamiento de las relaciones.

La Figura 12 muestra un ejemplo de entrada al programa, la cual consiste en una pila de secciones de imágenes de una célula, y su correspondiente salida reconstruida en 3D. En el primer paso, definieron 5 relaciones metamórficas basándose en el conocimiento del dominio, las cuales se indican a continuación.

> MR1: Inclusivo/Exclusivo. Si se añade/elimina una mitocondria artificial a/de una pila de secciones de imágenes confocales[13][14] originales de una célula, la nueva añadida debe ser reconocida e incluida/excluida en/de la estructura reconstruida, el número total de las mitocondrias debe aumentar/disminuir en uno, y el volumen calculado de las mitocondrias en la célula debe aumentar/disminuir en consecuencia (Ding, Zhang, & Hu, 2016, pág. 84).
>
> MR2: Multiplicativo. Si el tamaño de una mitocondria en las secciones de la imagen original se aumenta con un pequeño porcentaje. El número total de las mitocondrias debería mantenerse igual, y se espera que el volumen de las mitocondrias aumente (Ding, Zhang, & Hu, 2016, pág. 84).
>
> MR3: Longitudes. Hay un espacio entre las secciones de la imagen. Por lo tanto, una mitocondria pequeña puede aparecer sólo en una sección y una grande puede aparecer en múltiples secciones. Una mitocondria artificial puede añadirse a una sola sección o a múltiples secciones. La mitocondria se construirá a partir de las secciones de imagen modificadas. Se espera que la nueva mitocondria añadida aparezca en la salida de la

[13] https://www.rosario-conicet.gov.ar/equipamiento/centrodecomputosdealtorendimiento-2
[14] https://www.news-medical.net/life-sciences/Applications-of-Confocal-Imaging.aspx

estructura 3D junto con la original, y el volumen de las mitocondrias también aumentará (Ding, Zhang, & Hu, 2016, pág. 84).

MR4: Formas. Las mitocondrias tienen formas diferentes y estas determinan la estructura 3D de las mitocondrias reconstruidas. Se añaden mitocondrias artificiales con diferentes formas a las imágenes confocales para comprobar si las estructuras 3D de estas pueden construirse como se espera, y la estructura 3D de la original no debe modificarse (Ding, Zhang, & Hu, 2016, pág. 84).

MR5: Ubicaciones. El programa procesa las mitocondrias que están cerca del núcleo de forma diferente a las que están cerca del límite celular. Las mitocondrias artificiales se añaden a diferentes ubicaciones en la imagen, como la ubicación donde está cerca del núcleo o donde está cerca del límite de la célula. Las nuevas mitocondrias añadidas deberían ser reconocidas, y la estructura 3D de las mitocondrias añadidas debería reconstruirse como se espera y las estructuras 3D de otras mitocondrias originales no deberían verse afectada (Ding, Zhang, & Hu, 2016, pág. 84).

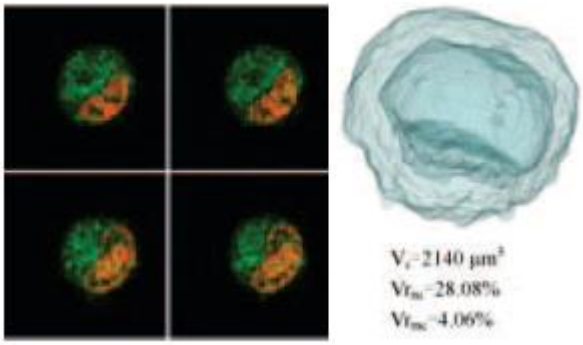

Figura 12 Entrada y salida resultante (Ding, Zhang, & Hu, 2016).

Como parte de la generación de las pruebas, se realiza la transformación de las pruebas de origen de acuerdo con las MRs. Las pruebas de origen y las transformadas se ejecutan una por una y sus resultados se comparan para decidir si la prueba pasa la relación metamórfica. En función de los resultados, pueden ser necesarias pruebas adicionales (Ding, Zhang, & Hu, 2016).

Para la evaluación de las MRs y las pruebas, agregaron código manualmente dentro del código fuente del programa, con el objetivo de comprobar la cobertura de pruebas. Una vez ejecutadas las mismas se comprobó que cumplieron las relaciones metamórficas definidas (Ding, Zhang, & Hu, 2016). Según Ding et al., (2016), los criterios de cobertura del programa fueron satisfechos al 100% por las pruebas.

El último paso consiste en el refinamiento o perfeccionamiento de las relaciones metamórficas, en este sentido, de acuerdo con el trabajo (Ding, Zhang, & Hu, 2016) las MR1 y la MR2 pueden refinarse aún más para saber cuál es el cambio exacto del volumen de las mitocondrias cuando se añade o se elimina una de ellas.

Hasta acá se ha podido revisar dos propuestas de aplicación de pruebas metamórficas para sistemas de Big Data y aprendizaje automático, los cuales no comparten dominio del conocimiento, pero si la característica que ambos no cuentan con un oráculo de pruebas. En la siguiente sección se analizarán otras técnicas de testing que pueden aplicarse a estos tipos de sistemas.

3.3 Otros Tipos de Testing

Hadoop ofrece una plataforma para pruebas de Big Data, las cuales pueden dividirse en tres fases, validación de datos o pre-etapa Hadoop, validación MapReduce y validación de la salida.

La validación de datos es el inicio del proceso el cual ayuda a garantizar que los datos que se ingresen al sistema de archivos distribuidos Hadoop sean correctos. Estas pruebas son realizadas para las diferentes fuentes desde las cuales se toman los datos, por ejemplo, redes sociales, bases de datos relacionales, etc. Algunas herramientas utilizadas para esta etapa son Talend[15] y Datameer[16] (Punn et al., 2019).

Durante la etapa de validación MapReduce, se verifica que el procedimiento funcione sin fallos y los conjuntos de clave-valores se generen correctamente de acuerdo con las reglas de negocio especificadas (Punn et al., 2019).

En la etapa de validación de la salida, se comprueban que las reglas de cambio estén aplicadas correctamente, y la efectiva carga de los datos en la organización resultante (Punn et al., 2019).

Otra técnica es la prueba basada en clasificación (Tao & Gao, 2016), la cual involucra dos pasos, el entrenamiento de un clasificador para diferenciar casos fallidos de los exitosos sobre un subconjunto de resultados seleccionados para luego aplicar dicho clasificador al conjunto principal de datos.

Otro enfoque es el de pruebas de origen colectivo (del inglés, *crowd-sourced testing*) (Tao & Gao, 2016), la cual utiliza testers autónomos y/o ingenieros contratados en una

[15] https://www.talend.com/es/
[16] https://www.datameer.com/

comunidad de origen colectivo, es decir personas que se contratan para ejecutar las pruebas. Estas pueden ser realizadas ya sea forma manual o generando automatizaciones para la validación de un software. Es un método rentable para validar sistemas de aplicaciones basados en el aprendizaje automático, como el reconocimiento facial humano. Un ejemplo de estas comunidades es uTest[17].

[17] https://www.utest.com/

Capítulo 4

Aplicación de testing metamórfico

4.1 Introducción

Teniendo en cuenta lo visto en el capítulo 3 sobre aplicación de pruebas metamórficas, en este capítulo se busca implementar este tipo de pruebas en dos sistemas de software, los cuales hacen uso de Big Data como fuente de datos y algún tipo de algoritmo de aprendizaje automático dentro de su código.

En la sección 4.2, tomando el ejemplo mencionado en el trabajo "*Research Directions for Engineering Big Data Analytics Software*" (Otero & Peter, 2015, pág. 14), se lo lleva a la práctica de forma parcial para el procesamiento de un solo salto, utilizando como fuente de datos textos de redes sociales, específicamente la red de microblogging Twitter. Se analiza el sentimiento de los textos, donde se entrena un modelo basándose en conjunto de datos etiquetados previamente curados.

En la sección 4.3, teniendo en cuenta lo realizado en los trabajos "*Metamorphic Testing for Software Quality Assessment: A Study of Search Engines*" (Zhou, Xiang, & Chen, 2016) y "*Metamorphic Testing: Testing the Untestable*" (Segura, Towey, Zhou, & Chen, 2020), se diseñan relaciones metamórficas y se aplican en el motor de búsqueda de Google, ya que se considera para este trabajo, que es un sistema el cual realiza búsquedas sobre grandes volúmenes de datos utilizando técnicas de aprendizaje automático como procesamiento de lenguaje natural (Nayak, 2019).

Los resultados obtenidos de las pruebas en ambas aplicaciones se reflejan en la sección 4.4.

4.2 Testing en análisis de sentimiento en Tweets

Como menciona el artículo del World Economic Forum, en el año 2019, 500 millones de Tweets fueron enviados a través de la red social Twitter (World Economic Forum, 2019). En el año 2020, esta red ha alcanzado los 186 millones de usuarios activos diarios (Iqbal, 2021). Considerando estos datos no sería posible ejecutar pruebas para analizar sentimientos sobre todas las posibilidades de textos escritos, sinónimos o combinación de palabras en una red social como Twitter.

Para ello, teniendo en cuenta el trabajo "*Research Directions for Engineering Big Data Analytics Software*" (Otero & Peter, 2015, pág. 14), se aplican pruebas metamórficas a un sistema realizado a modo de ejemplo para análisis de sentimiento de texto mediante aprendizaje

automático (ver Anexo I). En este caso se lo trata como un sistema de caja blanca ya que se conoce internamente el funcionamiento del código y el algoritmo de aprendizaje automático supervisado utilizado (Mahesh, 2020) (Theobald, 2017), clasificación binaria[18]. Este algoritmo dará como resultado sentimiento positivo o negativo mediante el entrenamiento provisto por la librería utilizada [19]. Respecto al conjunto de datos para entrenar al modelo de aprendizaje automático, se utilizaron los del trabajo "*From Group to Individual Labels using Deep Features*" (Kotzias, Denil, Smyth, & Freitas, 2015), el cual se encuentra ya etiquetado y alojado en el repositorio UCI Machine Learning Repository[20].

Para la ejecución de las pruebas se identifican las siguientes relaciones en el contexto dado, mediante el enfoque basado en las salidas (Segura et al., 2020).

MR #1: Dado un Tweet T con un vocabulario V en su texto, que arroja determinado resultado de ejecución de análisis de sentimiento R1 (caso de prueba de origen), volver a ejecutar el análisis cambiando el texto con sinónimos de V donde se espera obtener el mismo resultado R1 (caso de seguimiento).

MR#2: Dado un Tweet T con un resultado de análisis de sentimiento R1 (caso de prueba de origen), invertir las palabras del texto de T sin modificar el sentido de este y ejecutar el análisis donde debe dar como resultado R1 (caso de seguimiento).

MR #3: Dado un Tweet T con un vocabulario V en su texto dando como resultado de análisis de sentimiento R1 (caso de prueba de origen), cambiando el texto con antónimos de V y volviendo a ejecutar la aplicación se espera el resultado contrario a R1 (caso de seguimiento).

MR #4: Dado un Tweet T con un vocabulario V en su texto con un resultado de análisis de sentimiento R1 (caso de prueba de origen), volver a ejecutar el análisis cambiando el texto con una negación de V donde se espera el resultado contrario a R1(caso de seguimiento).

Para cada relación metamórfica se ejecutaron en forma automatizada dos iteraciones, ingresando en forma manual Tweets obtenidos mediante la web de la red social. La información detallada sobre el procedimiento efectuado puede verse en el Anexo I.

A continuación, a modo de ejemplo se demuestra una iteración para la MR #1. Para ello se utiliza como entrada el Tweet *"@MovistarArg son un desastre, nunca tengan un problema con estos tipos porque no te lo solucionan"* como se muestra en la Figura 13 y luego se vuelve a llevar a cabo la ejecución del análisis de sentimiento cambiando las palabras "un *desastre"*

[18] https://docs.microsoft.com/en-us/dotnet/machine-learning/how-to-choose-an-ml-net-algorithm#linear-algorithms
[19] https://docs.microsoft.com/en-us/dotnet/machine-learning/tutorials/sentiment-analysis
[20] https://archive.ics.uci.edu/ml/datasets/Sentiment+Labelled+Sentences

por "*una calamidad*". El resultado del análisis de sentimiento en el caso de seguimiento es el mismo que el caso de origen, por lo que es posible decir que la modificación de un adjetivo por un sinónimo de este no cambia el resultado entonces no se rompe la relación definida. Esto puede verse en la Figura 14, en la ejecución realizada de forma automatizada mediante el marco de trabajo de código abierto NET Core, el lenguaje de programación C#, la librería de pruebas unitarias xUnit.

Figura 13 Tweet 1 Ejecución 1.1 - MR #1

```
[Fact]
0 references
public void MR1Iteration1Test()
{
    // Arrange
    var input1 = @"@MovistarArg son un desastre, nunca tengan un problema con
                   estos tipos porque no te lo solucionan";
    var input2 = @"@MovistarArg son una calamidad, nunca tengan un problema con
                   estos tipos porque no te lo solucionan";

    // Act Source test case
    var resultIteration1 = _fixture.SentimentAnalyzerService.UseModelWithSingleItem(_fixture.MlContext, _fixture.Model, input1);
    // Act Follow-up test case
    var resultIteration2 = _fixture.SentimentAnalyzerService.UseModelWithSingleItem(_fixture.MlContext, _fixture.Model, input2);

    // Metamorphic relation #1 assertion
    Assert.Equal(Result.Negative, resultIteration1.Prediction);
    Assert.Equal(Result.Negative, resultIteration2.Prediction);
}
```

Figura 14 Ejecución 1.1 MR #1

Los resultados obtenidos para todas las relaciones definidas se encuentran en la sección 4.4.

4.3 Testing en motor de búsqueda de Google

Teniendo en cuenta que el motor de búsqueda de Google utiliza grandes volúmenes de datos, más de 100.000.000 de gigabytes de tamaño en su índice (Google, s.f.) y que los mismos provienen de diferentes fuentes como páginas web, contenido enviado por usuarios, libros escaneados, base de datos públicas en internet, etc. (Google, 2021), una búsqueda realizada por un usuario puede retornar una cantidad de resultados potencialmente imposible de verificar en términos de tiempo para una persona. En este sentido, estamos frente al problema del oráculo de prueba mencionado en la sección 3.1.

Para aplicar pruebas metamórficas en este contexto, se identificaron y establecieron las siguientes relaciones utilizando el enfoque basado en las salidas (Segura et al., 2020). Para este

caso se lo trata como un sistema de caja negra, ya que no se conoce internamente el funcionamiento del código, pero si las posibilidades de búsqueda según la documentación de soporte del fabricante[21].

MR #1: Realizar una búsqueda utilizando como entrada una frase (caso de prueba de origen). Volver a realizar una búsqueda, pero ahora la misma frase entre comillas (caso de seguimiento). Se espera que la cantidad de resultados obtenidos en el caso de seguimiento sea menor que la prueba de origen, ya que al utilizar las comillas se está buscando por la frase exacta.

MR #2: Realizar una búsqueda utilizando como entrada una frase. Tomando el segundo resultado en caso de ser mayor a 2 o el primero si solo retornó 1, transformar el caso de prueba de origen aplicando el filtro de sitio o dominio específico. Se espera que el primer resultado obtenido en el caso de seguimiento sea igual al enlace tomado del caso de origen.

MR #3: Realizar una búsqueda utilizando como entrada 3 palabras sueltas que guarden poca o nula relación entre sí y utilizar el operador lógico "AND" entre ellas (caso de prueba de origen). Volver a realizar una búsqueda, pero ahora intercambiando las palabras de lugar (caso de seguimiento). Se espera que la cantidad resultados obtenidos en ambos casos sea la misma o con una intersección similar.

MR #4: Realizar una búsqueda utilizando como entrada una frase ambigua, ejemplo "***velocidad del jaguar"*** donde puede referirse tanto al animal como al vehículo. Ejecutar la búsqueda, pero aplicando el filtro de exclusión para limitar los resultados, para el ejemplo mencionado, sería ***-auto***. Se espera que la cantidad de resultados obtenidos en el caso de seguimiento sea menor a la cantidad de resultados de origen.

MR #5: Dada una búsqueda exacta para una frase de al menos dos palabras que retorna resultados, MR #1, al aplicar filtro de exclusión sobre la misma no debe haber resultados posibles.

Para cada relación metamórfica se ejecutaron en forma manual dos iteraciones, utilizando el navegador Microsoft Edge[22], información detallada sobre el procedimiento efectuado puede verse en el Anexo II. A continuación, se demuestra una iteración para la MR #1 a modo ilustrativo, para ello se utiliza como consulta de entrada la frase ***big data aplicado al marketing digital*** como se muestra en la Figura 15 y luego se vuelve a llevar a cabo la búsqueda con comillas, Figura 16. Los resultados obtenidos en el caso de seguimiento son

[21] https://support.google.com/websearch/answer/2466433
[22] https://www.microsoft.com/es-es/edge?r=1

menores al del caso de origen, por lo que es posible decir que es un subconjunto del caso de prueba de origen, ya que al utilizar las comillas se está buscando por la frase exacta y la cantidad de coincidencias debería ser menor, como se muestra en las Figuras 15 y 16.

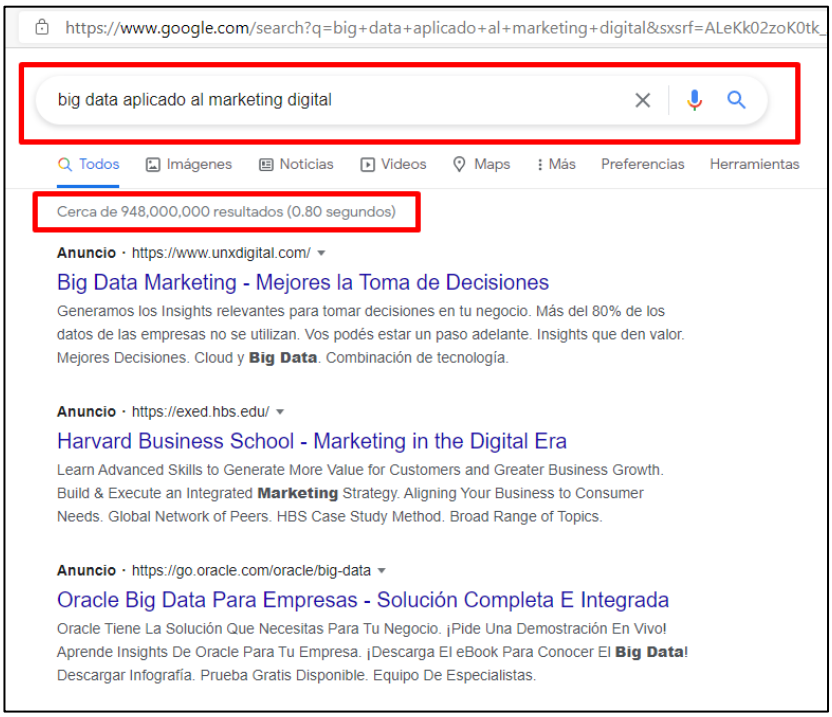

Figura 15 Ejecución 1.1 - MR #1

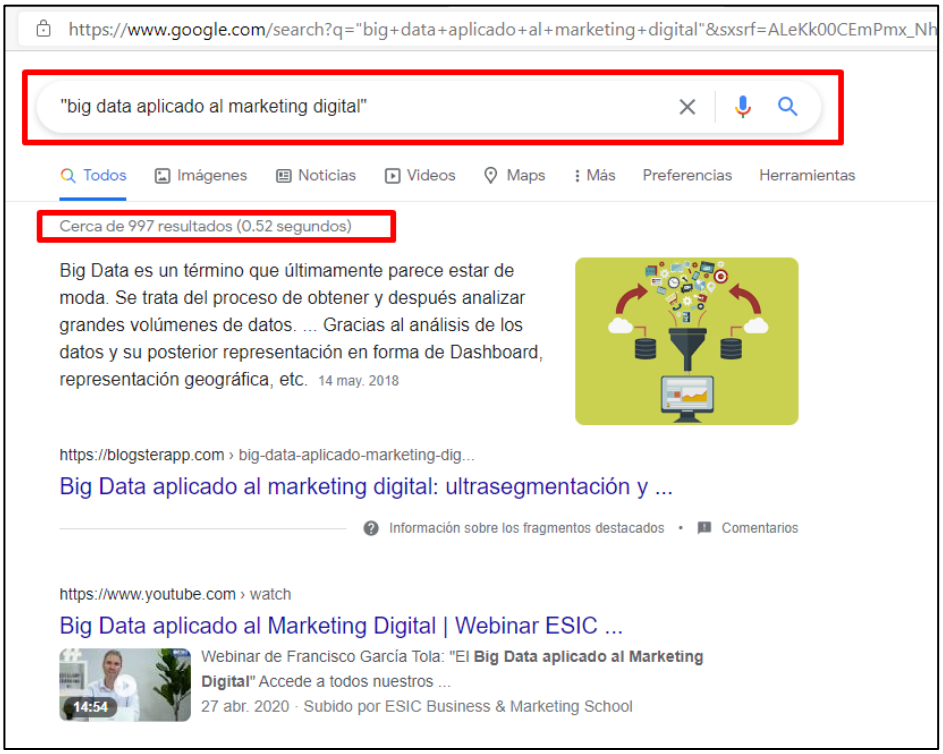

Figura 16 Ejecución 1.2 - MR #1

Para el caso de la MR #1, se automatiza la iteración 1 de la prueba mediante el marco de trabajo de código abierto NET Core, el lenguaje de programación C#, la librería de pruebas unitarias xUnit y la librería para automatización de pruebas de navegador web Selenium[23]. En la Figura 17 se puede ver parte del código realizado para probar dicha MR. Para detalles de la implementación ver el Anexo II.

```
using TestingMetamorfico.Services;
using Xunit;

namespace TestingMetamorfico.Tests
{
    0 references
    public class SearchServiceTest
    {
        [Fact]
        0 references
        public void MR1Iteration1Test()
        {
            // Arrange
            var searchService = new SearchService();
            var input1 = "big data aplicado al marketing digital";
            var input2 = ""big data aplicado al marketing digital"";

            // Act Source test case
            var resultIteration1 = searchService.CountSearchResults(input1);
            // Act Follow-up test case
            var resultIteration2 = searchService.CountSearchResults(input2);

            // Metamorphic relation #1 assertion
            Assert.True(resultIteration1 > resultIteration2);
        }
    }
}
```

Figura 17 Ejecución Iteración 1 MR #1 Automatizada

Los resultados obtenidos para todas las relaciones definidas se encuentran en la sección 4.4.

4.4 Resultados Obtenidos

La Tabla 1 muestra los resultados obtenidos para las pruebas realizadas del sistema de análisis de sentimiento en la red social Twitter. Información detallada sobre los resultados para cada una de las relaciones metamórficas y sus ejecuciones pueden encontrarse en el Anexo I.

Relación metamórfica	Cumple la MR
MR #1	Si para las dos iteraciones.
MR #2	Cumple solo la iteración 1.
MR #3	Si para las dos iteraciones.
MR #4	Cumple solo la iteración 1.

Tabla 1 Resultados pruebas en análisis de sentimiento en redes sociales

[23] https://github.com/SeleniumHQ/selenium

Si bien se tiene en cuenta que el algoritmo y la cantidad de datos utilizados es limitada a fines de poder producir un sistema lo menos complejos posible y a solo efectos de aplicar pruebas metamórficas, para la MR #2 se observa que la iteración 2 no cumple con el resultado esperado por lo cual no se verifica la relación metamórfica, lo mismo sucede con la iteración 2 de la MR #4. En este sentido, es posible decir que el sistema es defectuoso, si bien no sabemos exactamente lo que está sucediendo, con la ejecución de las MR es posible saber que no está libre de errores. Probablemente este defecto se deba al conjunto de datos utilizados de prueba para la alimentación del modelo ya que es acotado a fines prácticos debido a las limitaciones de obtener conjuntos de datos más grandes que se encuentren correctamente depurados y etiquetados para este trabajo.

La Tabla 2 muestra los resultados de las pruebas en el motor de búsqueda de Google. Información detallada sobre los resultados para cada una de las relaciones metamórficas y sus ejecuciones pueden encontrarse en el Anexo II.

Relación metamórfica	Cumple la MR
MR #1	Si para las dos iteraciones.
MR #2	Si para las dos iteraciones.
MR #3	Si para las dos iteraciones.
MR #4	Si para las dos iteraciones.
MR #5	Si para las dos iteraciones.

Tabla 2 Resultados pruebas en motor de búsqueda de Google

Para la MR #1 se observa que la automatización de la prueba no cambia el resultado de la MR, por lo cual se puede decir que es posible automatizar las pruebas metamórficas, ya que las mismas son agnósticas a la tecnología utilizada. Todas las pruebas cumplieron las MRs, pero en particular para la #5, si bien las entradas de las iteraciones definidas en la sección 4.3 cumplieron en un principio con la relación, al realizar una entrada alternativa al caso de seguimiento, se observa que la misma ya no se cumple por lo que debe refinarse con las iteraciones.

Con esto no es posible decir que el sistema bajo pruebas está libre de errores. Si bien la cantidad de pruebas e iteraciones ejecutadas son limitadas y no son suficientes ni significativas para sistemas de Big Data que manejan grandes volúmenes de datos, es factible decir que no se encontraron defectos de acuerdo con el dominio del problema. En la MR #5, el caso de seguimiento alternativo no se cumple por la limitación en la cantidad de iteraciones, las cuales

son necesarias para refinar las relaciones metamórficas a ejecutarse en una nueva ronda de pruebas.

Conclusiones

Mediante la indagación realizada acerca de las de pruebas en sistemas de Big Data y aprendizaje automático específicamente para el procedimiento de pruebas metamórficas, es posible decir que la adecuada delimitación de las relaciones permite generar casos de pruebas que asisten en la comprobación de la validez y veracidad de los resultados que se producen. Esto no quiere decir que el sistema bajo pruebas está libre de errores, pero permite verificar que el sistema no es defectuoso de acuerdo con el dominio del problema y las relaciones definidas.

Si bien los procedimientos realizados en el capítulo 4, presentan la limitación de tener un número bajo de iteraciones (2 por cada MR, debido al tiempo que conlleva generar una suficiente cantidad de datos de ingreso) y relaciones definidas, los resultados obtenidos indican que el refinamiento iterativo de las relaciones metamórficas es necesario para la correcta generación de los casos de prueba.

Por otro lado, es válido mencionar que se debe tener conocimiento del dominio del problema del software bajo prueba. Es decir que por más que seamos especialistas de testing, no podremos generar las relaciones que permitan validar un software sin el correcto conocimiento del problema en cuestión, si analizamos el ejemplo de la sección 4.3, no podríamos definir relaciones sin conocer previamente cómo funciona el motor de búsqueda de Google.

Una restricción que podría desprenderse del procedimiento expuesto es la necesidad de identificar las relaciones metamórficas, lo cual típicamente requiere un proceso manual que demanda esfuerzo y tiempo. Si bien existen aproximaciones para automatizarlas, se dan mayormente en programas numéricos.

Por último, se observa que existen otras técnicas de prueba que pueden ser aplicadas a sistemas de Big data, pero dependen de la tecnología en la cual dichos sistemas están soportados, como puede ser en el caso de la técnica de MapReduce cuando se hace uso de Hadoop o depende de la contratación de personal externo como puede ser el caso de las pruebas de origen colectivo. En este sentido, la aplicación de pruebas metamórficas es aplicable a cualquier tipo de sistema, como se ve en el Capítulo 3, se aplica a sistemas científicos, compiladores, buscadores en sitios webs, sin necesidad de contar con una tecnología específica para ello. Esto también puede verse en los ejemplos del Capítulo 4, donde se utilizó tecnología que no es específica para este tipo de pruebas, de hecho, se utilizan mayormente para realizar pruebas unitarias, pero con unos pequeños cambios es totalmente posible utilizarlas para

automatización o generar los casos que luego se ejecutan para comprobar la validación en sistemas de Big Data y aprendizaje automático.

Líneas Futuras de Investigación

En base al trabajo de indagación realizado y el procedimiento generado para la aplicación de pruebas metamórficas, se desprenden las siguientes líneas de investigación:

- Incorporar ejecución de pruebas metamórficas en un flujo de integración y despliegue continuo para aplicaciones que utilicen Big Data y aprendizaje automático.
- Investigar y analizar la ejecución de pruebas de performance en Big Data y aprendizaje automático, para evaluar tiempos de respuesta, escalabilidad y disponibilidad.
- Investigar y analizar marcos de trabajo para la generación de mutantes que puedan ser eliminados mediante relaciones metamórficas para la verificación y validación de un software de Big Data y aprendizaje automático.
- Evaluar otras técnicas de pruebas en Big Data que no fueron abarcadas en detalle en este trabajo de investigación como pruebas basadas en clasificación, pruebas de MapReduce y pruebas de origen colectivo.

Acrónimos

IoT Internet of Things.

JSON JavaScript Object Notation. Formato de archivo utilizado para el intercambio de datos.

XML Extensible Markup Language. Formato de archivo utilizado para el intercambio de datos.

NBA National Basketball Association

L1 Nivel 1, del inglés Level 1

L2 Nivel 2, del inglés, Level 2

L3 Nivel 3, del inglés, Level 3

SDLC Software Development Lifecycle

APIs Application Programming Interfaces

MT Metamorphic Testing

MR Metamorphic Relation

MRs Metamorphic Relations

MG Metamorphic Group of Inputs

CMA Cell Morphology Assay

URL Uniform Resource Locator

ML.NET Machine Learning .NET

Referencias

Alzubi, J., Nayyar, A., & Kumar, A. (Noviembre de 2018). Machine Learning from Theory to Algorithms: An Overview. *Journal of Physics: Conference Series, 1142*, 012012. doi:10.1088/1742-6596/1142/1/012012

Ammann, P., & Offutt, J. (2008). *Introduction to Software Testing.* Nueva York, Estados Unidos de Norte America: Cambridge University Press.

Anwar, N., & Kar, S. (2019). Review Paper on Various Software Testing Techniques & Strategies. *Global Journal of Computer Science and Technology, 19*(2), 43-49.

Bengio, S., Deng, L., Larochelle, H., Lee, H., & Salakhutdinov, R. (18 de Junio de 2013). Guest Editors' Introduction: Special Section on Learning Deep Architectures. *IEEE Transactions on Pattern Analysis and Machine Intelligence, 35*(8), 1795-1797.

Bengio, Y. (2009). Learning Deep Architectures for AI. *Foundations and Trends® in Machine Learning, 2*(1), 1-127.

Benuwa, B. B., Zhan, Y. Z., Ghansah, B., Wornyo, D. K., & Banaseka, F. K. (2016). A Review of Deep Machine Learning. *International Journal of Engineering Research in Africa, 24*, 124-136.

Cano, D. (2015). *Tests de regresión automáticos creados por el usuario final sobre ambientes persistentes orientados a objetos.* Universidad Nacional de La Plata, Facultad de Informática. La Plata, Buenos Aires: Universidad Nacional de La Plata.

Chen, T. Y., Kuo, F.-C., Liu, H., Poon, P.-L., Towey, D., Tse, T., & Zhou, Z. Q. (Enero de 2018). Metamorphic Testing: A Review of Challenges and Opportunities. *ACM Computing Surveys, 51*(1), 4:1-4:27. doi:https://doi.org/10.1145/3143561

Chen, T., Cheung, S., & Yiu, S. (1998). *Metamorphic testing: a new approach for generating next test cases.* Hong Kong University of Science and Technology, Department of Computer Science, Hong Kong.

Cisco Inc. (9 de Marzo de 2020). *Cisco Annual Internet Report (2018–2023) White Paper.* Cisco Inc. https://www.cisco.com/c/en/us/solutions/collateral/executive-perspectives/annual-internet-report/white-paper-c11-741490.html

De Nicola, G., Di Tommaso, P., Rosaria, E., Francesco, F., Pietro, M., & Orazzo, A. (2005). A Grey-Box Approach to the Functional Testing of Complex Automatic Train Protection Systems. *European Dependable Computing Conference* (págs. 305-317). Berlin, Heidelberg: Springer Berlin Heidelberg.

Ding, J., Hu, X.-H., & Gudivada, V. (09 de Marzo de 2017). A Machine Learning Based Framework for Verification and Validation of Massive Scale Image Data. *IEEE Transactions on Big Data*. doi:10.1109/TBDATA.2017.2680460

Ding, J., Zhang, D., & Hu, X.-H. (2016). A Framework for Ensuring the Quality of a Big Data Service. *2016 IEEE International Conference on Services Computing (SCC)* (págs. 82-89). San Francisco, CA: IEEE. doi:10.1109/SCC.2016.18

Fortune Business Insights. (Octubre de 2020). *Fortune Business Insights - Technology & Media, Big Data Technology Market*. Fortune Business Insights. https://www.fortunebusinessinsights.com/industry-reports/big-data-technology-market-100144

Gartner. (2001). Information Technology Gartner Glossary. https://www.gartner.com/en/information-technology/glossary/big-data#:~:text=Big%20data%20is%20high%2Dvolume,decision%20making%2C%20and%20process%20automation.

Google. (22 de Abril de 2021). *¿Cómo funciona la Búsqueda de Google?* Google Developers. https://developers.google.com/search/docs/beginner/how-search-works

Google. (s.f.). *How Search organizes information*. Google Search. https://www.google.com/search/howsearchworks/crawling-indexing/

Gupta, N. (2014). Different Approaches to White Box Testing to Find Bug. *International Journal of Advanced Research in Computer Science & Technology*, *2*(3), 46-49.

Hayhurst, C. (26 de Agosto de 2020). *How NBA Data Analytics is Changing the Sport*. Dell Technologies. https://www.delltechnologies.com/en-us/perspectives/tech-takes-the-court-nba-teams-turn-to-data-and-analytics-for-a-competitive-edge/

Hinton, G. E., Osindero, S., & Teh, Y.-W. (2006). A Fast Learning Algorithm for Deep Belief Nets. *Neural Computation, 18*(7), 1527-1554.

Honest, N. (31 de Mayo de 2019). Role of Testing in Software Development Life Cycle. *International Journal of Computer Sciences and Engineering, 7*(5), 886-889. doi:https://doi.org/10.26438/ijcse/v7i5.886889

IBM. (s.f.). *¿Qué es Machine Learning?* IBM. https://www.ibm.com/ar-es/analytics/machine-learning

IBM. (2020). *Software testing*. IBM Web Site. https://www.ibm.com/topics/software-testing

IBM. (s.f.). *Extracting business value from the 4 V's of big data*. IBM Big Data & Analyctics Hub. https://www.ibmbigdatahub.com/infographic/extracting-business-value-4-vs-big-data

Iqbal, M. (14 de Mayo de 2021). *Twitter Revenue and Usage Statistics (2021)*. Business of Apps. https://www.businessofapps.com/data/twitter-statistics/

Jamil, A., Arif, M., Ahmad, A., & Awang Abubakar, N. (2016). Software Testing Techniques: A Literature Review. *2016 6th International Conference on Information and Communication Technology for The Muslim World (ICT4M)* (págs. 177-182). Jakarta: IEEE. doi:10.1109/ICT4M.2016.045

Jarman, D. C., Zhou, Z. Q., & Chen, T. Y. (2017). Metamorphic Testing for Adobe Data Analytics Software. *2017 IEEE/ACM 2nd International Workshop on Metamorphic Testing (MET)* (págs. 21-27). Buenos Aires: IEEE. doi:10.1109/MET.2017.1

Jin, X., Wah, B. W., Cheng, X., & Wang, Y. (2015). Significance and Challenges of Big Data Research. *Big Data Research, 2*(2), 59-64.

Kale, A. M., Bandal, V. V., & Chaudhari, K. (Enero de 2019). A Review Paper on Software Testing. *International Research Journal of Engineering and Technology, 6*(1), 1268-1273.

Kotzias, D., Denil, M., Smyth, P., & Freitas, N. d. (Agosto de 2015). From Group to Individual Labels Using Deep Features. *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 597–606. doi:10.1145/2783258.2783380

Lawanna, A. (Junio de 2012). The Theory of Software Testing. *Intelligent Transportation Systems Journal, 16*, 35-40. https://www.researchgate.net/publication/236031163_The_Theory_of_Software_Testing

Liu, H., Kuo, F.-C., Towey, D., & Chen, T. Y. (Enero de 2014). How Effectively does Metamorphic Testing Alleviate the Oracle Problem? *IEEE Transactions on Software Engineering, 40*(1), 4-22. doi:10.1109/TSE.2013.46

Mahesh, B. (Enero de 2020). Machine Learning Algorithms - A Review. *International Journal of Science and Research (IJSR), 9*(1), 381-386. doi:10.21275/ART20203995

McNulty, E. (Mayo de 2014). *Understanding Big Data: The Seven V's*. Dataconomy. https://dataconomy.com/2014/05/seven-vs-big-data/

Microsoft. (s.f.). *What is machine learning?* Microsoft Azure. https://azure.microsoft.com/en-us/overview/what-is-machine-learning-platform/

Nayak, P. (25 de Octubre de 2019). *Understanding searches better than ever before*. Google Blog. https://blog.google/products/search/search-language-understanding-bert/

Oracle. (s.f.). *¿Qué es big data?* Oracle. https://www.oracle.com/ar/big-data/what-is-big-data.html#link3

Otero, C. E., & Peter, A. (Enero-Febrero de 2015). Research Directions for Engineering Big Data Analytics Software. *IEEE Intelligent Systems, 30*(1), 18.

Patgiri, R., & Ahmed, A. (2016). Big Data: The V's of the Game Changer Paradigm. *18th IEEE High Performance Computing and Communications* (págs. 18-21). Sydney: IEEE. doi:10.1109/HPCC-SmartCity-DSS.2016.0014

Punn, N. S., Agarwak, S., Syafrullah, M., & Adiyarta, K. (2019). Testing Big Data Application. *2019 6th International Conference on Electrical Engineering, Computer Science and Informatics (EECSI)* (págs. 159-162). Bandung, Indonesia, Indonesia: IEEE. doi:https://doi.org/10.23919/EECSI48112.2019.8976972

Samuel, A. (Julio de 1959). Some Studies in Machine Learning Using the Game of Checkers. *IBM Journal of Research and Development, 3*(3), 210-229. doi:10.1147/rd.33.0210

Segura, S., Towey, D., Zhou, Z., & Chen, T. (Mayo-Junio de 2020). Metamorphic Testing: Testing the Untestable. *IEEE Software, 37*(3), 46-53.

Siroky, D. (30 de Octubre de 2017). *The glitch economy: Counting the cost of software failures*. CloudTech. https://cloudcomputing-news.net/news/2017/oct/30/glitch-economy-counting-cost-software-failures/

Sneha, K., & Malle, G. M. (2017). Research on software testing techniques and software automation testing tools. *2017 International Conference on Energy, Communication, Data Analytics and Soft Computing (ICECDS)* (págs. 77-81). Chennai: IEEE. doi:10.1109/ICECDS.2017.8389562

Tao, C., & Gao, J. (2016). Quality Assurance for Big Data Applications– Issues, Challenges, and Needs. *The Twenty-Eighth International Conference on Software Engineering and Knowledge Engineering.* San Francisco CA.

Theobald, O. (2017). *Machile Learning For Absolute Beginners* (Segunda ed.). Independently Published.

World Economic Forum. (17 de Abril de 2019). *How much data is generated each day?*. World Economic Forum Web Site. https://www.weforum.org/agenda/2019/04/how-much-data-is-generated-each-day-cf4bddf29f/

Zhou, Z. Q., Xiang, S., & Chen, T. Y. (1 de Marzo de 2016). Metamorphic Testing for Software Quality Assessment: A Study of Search Engines. *IEEE Transactions on Software Engineering*, *42*(3), 264-284. doi:10.1109/TSE.2015.2478001

Anexo I

Procedimiento en análisis de sentimiento en Tweets

Para realizar el programa de análisis de sentimiento de los textos de Twitter, se tomó como ejemplo una aplicación de análisis de sentimientos de los comentarios de un sitio web utilizando el algoritmo supervisado de aprendizaje automático clasificación binaria para la librería ML.NET[24].

Como se muestra en la Figura 18, se creó una clase llamada "*SentimentAnalyzerService*" la cual cuenta con el método "*UseModelWithSingleItem*" que recibe como parámetro el texto a analizar y el modelo de aprendizaje automático entrenado, devolviendo la predicción que cuenta con el resultado, negativo o positivo, y una probabilidad que va de 0 a 1 respecto a la predicción. Los métodos "*LoadData*", "*BuildAndTrainModel*" y "*Evaluate*", son los encargados de cargar y entrenar el modelo basado en el conjunto de datos etiquetados utilizados para este software. Para el desarrollo del código fuente de la prueba se utilizó Net Core 3.1 (SDK 3.1.409) como marco de trabajo, C# 8.0 como lenguaje, ML.NET versión 1.5.5 como librerías aprendizaje automático y xUnit 2.4.0 como librería para ejecución de pruebas unitarias. El código fuente desarrollado tanto para el programa de análisis de sentimiento como para las pruebas metamórficas se encuentra alojado en un repositorio en Github[25].

```
namespace TestingMetamorfico.Services
{
    2 references
    public class SentimentAnalyzerService
    {
        1 reference
        public TrainTestData LoadData(MLContext mlContext, string dataPath)...

        1 reference
        public ITransformer BuildAndTrainModel(MLContext mlContext, IDataView splitTrainSet)...

        1 reference
        public void Evaluate(MLContext mlContext, ITransformer model, IDataView splitTestSet)...

        16 references | 6/8 passing
        public PredictionResult UseModelWithSingleItem(MLContext mlContext, ITransformer model, string text)
        {
            PredictionEngine<SentimentData, SentimentPrediction> predictionFunction = mlContext.Model.CreatePredictionEngine<SentimentData

            SentimentData sampleStatement = new SentimentData
            {
                SentimentText = text
            };

            var resultPrediction = predictionFunction.Predict(sampleStatement);

            return new PredictionResult()
            {
                Prediction = Convert.ToBoolean(resultPrediction.Prediction) ? Result.Positive : Result.Negative,
                Probability = resultPrediction.Probability
            };
        }
    }
}
```

Figura 18 Clase SentimentAnalyzerService

[24] https://docs.microsoft.com/en-us/dotnet/machine-learning/tutorials/sentiment-analysis
[25] https://github.com/nicolasgranata/testing-metamorfico

El conjunto de datos fue traducido de ingles a español con el objetivo de entrenar un modelo que pueda distinguir sentimientos positivos o negativos en textos escritos en dicho idioma, el mismo se encuentra alojado en el repositorio del código fuente bajo el nombre "*yelp_labelled.txt*".

Para la automatización de la comprobación de las relaciones metamórficas, se utilizó la librería xUnit que permite la generación de pruebas unitarias, las cuales invocan al método "*UseModelWithSingleItem"* utilizando como entradas textos obtenidos de la red social Twitter y se espera obtener los resultados que comprueban la relación. A modo de ejemplo La Figura 19, muestra la prueba llamada "MR1Iteration1Test", la cual ejecuta la iteración 1 de relación metamórfica #1.

```
[Fact]
0 references
public void MR1Iteration1Test()
{
    // Arrange
    var input1 = @"@MovistarArg son un desastre, nunca tengan un problema con
                   estos tipos porque no te lo solucionan";
    var input2 = @"@MovistarArg son una calamidad, nunca tengan un problema con
                   estos tipos porque no te lo solucionan";

    // Act Source test case
    var resultIteration1 = _fixture.SentimentAnalyzerService.UseModelWithSingleItem(_fixture.MlContext, _fixture.Model, input1);
    // Act Follow-up test case
    var resultIteration2 = _fixture.SentimentAnalyzerService.UseModelWithSingleItem(_fixture.MlContext, _fixture.Model, input2);

    // Metamorphic relation #1 assertion
    Assert.Equal(Result.Negative, resultIteration1.Prediction);
    Assert.Equal(Result.Negative, resultIteration2.Prediction);
}
```

Figura 19 Método MR1Iteration1Test

El procedimiento de pruebas metamórficas automatizadas fue llevado a cabo con la configuración de ambiente que se muestra en la Tabla 3.

Sistema Operativo	Memoria RAM	CPU
Windows 10 Enterprise Version 1909 (OS Build 18363.1556)	16 GB	Intel Core i7-8665U

Tabla 3 Ambiente de pruebas

Relación metamórfica #1

La Tabla 4 lista los datos y los resultados de la ejecución de la prueba para la MR #1. En las Figuras 20 y 21 pueden verse las entradas utilizadas y en la Figura 22, pueden verse los resultados obtenidos para cada iteración.

MR	Iteración	Caso	Entrada	Resultado	Cumple la MR	Fecha de ejecución
# 1	1	Origen	Ver Tabla 5	Negativo	Si	23/06/2021
		Seguimiento	Ver Tabla 5	Negativo		
# 1	2	Origen	Ver Tabla 5	Positivo	Si	23/06/2021
		Seguimiento	Ver Tabla 5	Positivo		

Tabla 4 Ejecución MR #1

MR	Iteración	Caso	Entrada
# 1	1	Origen	“@MovistarArg son un desastre, nunca tengan un problema con estos tipos porque no te lo solucionan”
# 1	1	Seguimiento	“@MovistarArg son una calamidad, nunca tengan un problema con estos tipos porque no te lo solucionan”
# 1	2	Origen	Nooo no podes seguir privándote de esas delicias. Los churros son todos ricos, dulces o salados. La sensación que te da cuando los mordes, es algo inexplicable. ¡Tenes que probarlos!
# 1	2	Seguimiento	Nooo no podes seguir privándote de esas delicias. Los churros son todos sabrosos, dulces o salados. La sensación que te da cuando los mordes, es algo indescriptible¡Tenes que probarlos!

Tabla 5 Entradas MR #1

Figura 20 Entrada MR #1 Iteración 1 Caso de origen

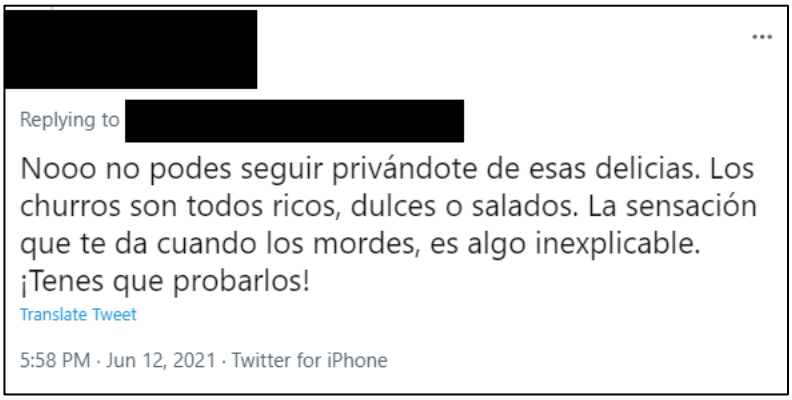

Figura 21 Entrada MR #1 Iteración 2 Caso de origen

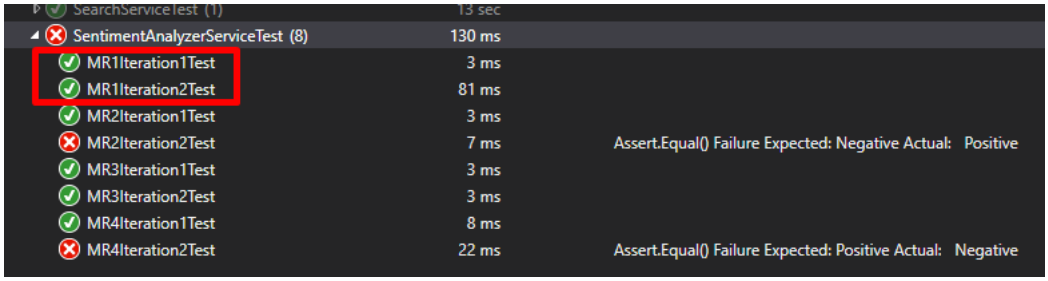

Figura 22 Resultados pruebas automatizadas MR #1

Relación metamórfica #2

La Tabla 6 lista los datos y los resultados de la ejecución de la prueba para la MR #2. En las Figuras 23 y 24 pueden verse las entradas utilizadas y en la Figura 25, pueden verse los resultados obtenidos para cada iteración.

MR	Iteración	Caso	Entrada	Resultado	Cumple la MR	Fecha de ejecución
# 2	1	Origen	Ver Tabla 7	Positivo	Si	23/06/2021
		Seguimiento	Ver Tabla 7	Positivo		
# 2	2	Origen	Ver Tabla 7	Negativo	No	23/06/2021
		Seguimiento	Ver Tabla 7	Positivo		

Tabla 6 Ejecución MR #2

MR	Iteración	Caso	Entrada
# 2	1	Origen	"El “chocomensaje” es buenísimo!!"
# 2	1	Seguimiento	"Es buenísimo el “chocomensaje”!!"
# 2	2	Origen	"@ClaroArgentina Claro, es una verguenza el servicio que tienen. 0 barras de señal. Y encima me subis a 1300 pesos el abono. Una solución?"
# 2	2	Seguimiento	“@ClaroArgentina Claro, 0 barras de señal. Una solución? Y encima me subis a 1300 pesos el abono, es una verguenza el servicio que tienen."

Tabla 7 Entradas MR #2

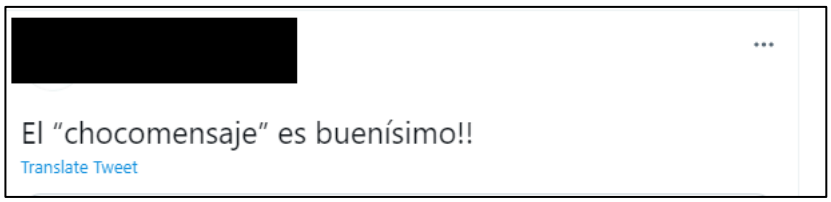

Figura 23 Entrada MR #2 y MR #4 Iteración 1 Caso de origen

Figura 24 Entrada MR #2 Iteración 2 Caso de origen

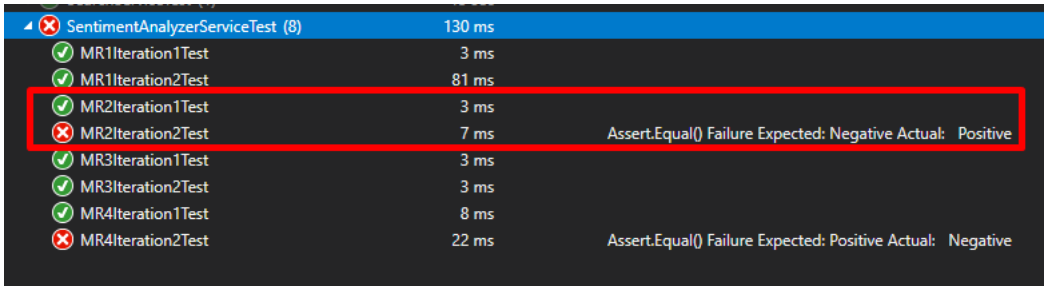

Figura 25 Resultados pruebas automatizadas MR #2

Relación metamórfica #3

La Tabla 8 lista los datos y los resultados de la ejecución de la prueba para la MR #3. En las Figuras 26 y 27 pueden verse las entradas utilizadas y en la Figura 28, pueden verse los resultados obtenidos para cada iteración.

MR	Iteración	Caso	Entrada	Resultado	Cumple la MR	Fecha de ejecución
# 3	1	Origen	Ver Tabla 9	Positivo	Si	23/06/2021
		Seguimiento	Ver Tabla 9	Negativo		
# 3	2	Origen	Ver Tabla 9	Negativo	Si	23/06/2021
		Seguimiento	Ver Tabla 9	Positivo		

Tabla 8 Ejecución MR #3

MR	Iteración	Caso	Entrada
# 3	1	Origen	“me pedí unas pastitas por rappi, excelente servicio, vinieron con pancito casero y queso rallado viva l'italia”
# 3	1	Seguimiento	"me pedí unas pastitas por rappi, mal servicio, vinieron con pancito casero y queso rallado viva l'italia"
# 3	2	Origen	“@pedidosya horrible atención al cliente. Nunca mas.”
# 3	2	Seguimiento	"@pedidosya buena atención al cliente. Siempre mas."

Tabla 9 Entradas MR #3

Figura 26 Entrada MR #3 Iteración 1 Caso de origen

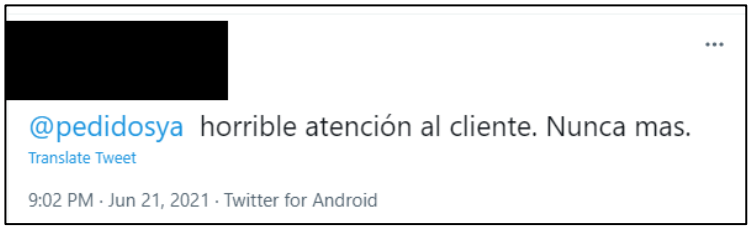

Figura 27 Entrada MR #3 Iteración 2 Caso de origen

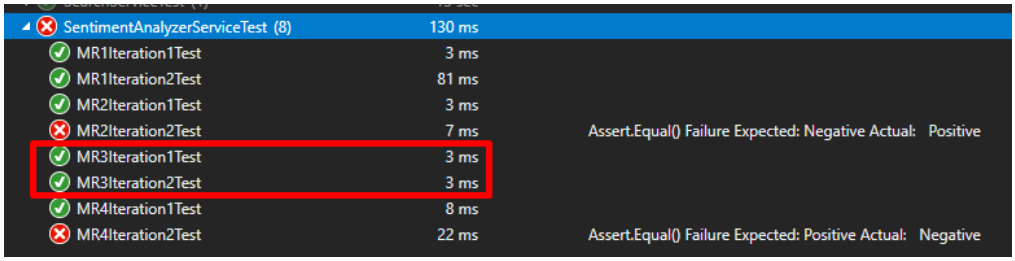

Figura 28 Resultados pruebas automatizadas MR #3

Relación metamórfica #4

La Tabla 10 lista los datos y los resultados de la ejecución de la prueba para la MR #4. En las Figuras 23 y 29 pueden verse las entradas utilizadas y en la Figura 30, pueden verse los resultados obtenidos para cada iteración.

MR	Iteración	Caso	Entrada	Resultado	Cumple la MR	Fecha de ejecución
# 4	1	Origen	Ver Tabla 11	Positivo	Si	23/06/2021
		Seguimiento	Ver Tabla 11	Negativo		
# 4	2	Origen	Ver Tabla 11	Negativo	No	23/06/2021
		Seguimiento	Ver Tabla 11	Negativo		

Tabla 10 Ejecución MR #4

MR	Iteración	Caso	Entrada
# 4	1	Origen	"El “chocomensaje” es buenísimo!!"
# 4	1	Seguimiento	"El “chocomensaje” no es buenísimo!!"
# 4	2	Origen	“@CorreoOficialSA va a ser un mes que envié un paquete a Catamarca y aún no llega , no me responden por redes , ni por mail ni por teléfono , estan dando un servicio malísimo , y sigo esperando una respuesta !”
# 4	2	Seguimiento	“@CorreoOficialSA va a ser un mes que envié un paquete a Catamarca y ya llegó , me responden por redes , por mail y por teléfono , estan dando un

			servicio que no es malísimo , y no sigo esperando una respuesta !”

Tabla 11 Entradas MR #4

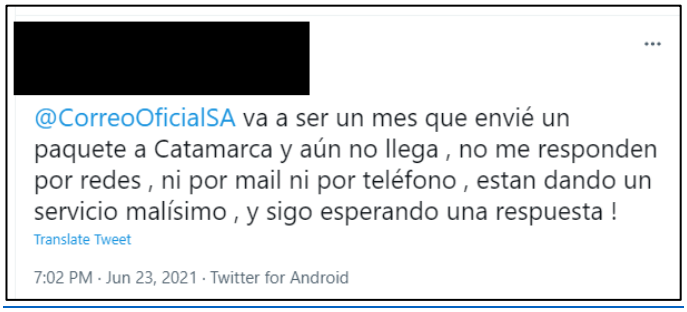

Figura 29 Entrada MR #4 Iteración 2 Caso de origen

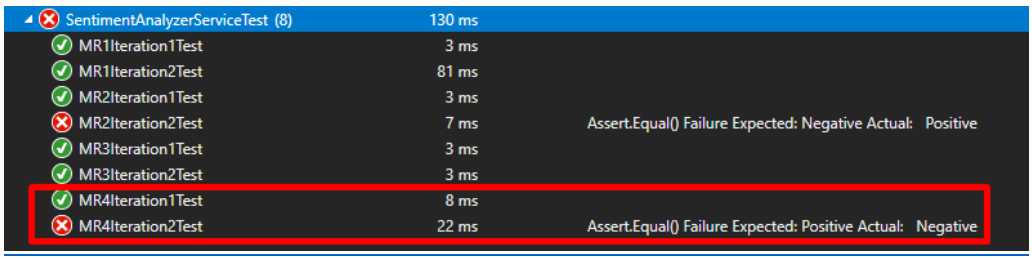

Figura 30 Resultados pruebas automatizadas MR #4

Anexo II

Procedimiento en motor de búsqueda de Google

El procedimiento de pruebas manuales fue llevado a cabo con la configuración de ambiente que se muestra en la Tabla 12, para el cual se ejecutó el navegador web utilizando la URL http://www.google.com, la cual lleva al buscador.

Sistema Operativo	Memoria RAM	CPU	Navegador
Windows 10 Enterprise Version 1909 (OS Build 18363.1556)	16 GB	Intel Core i7-8665U	Microsoft Edge Version 90.0.818.66 (Official build) (64-bit)

Tabla 12 Ambiente de pruebas manuales

El procedimiento automatizado para la relación metamórfica #1 fue realizado utilizando la configuración de ambiente que se muestra en la Tabla 13. Para el desarrollo del código fuente de la prueba se utilizó Net Core 3.1 (SDK 3.1.409) como marco de trabajo, C# 8.0 como lenguaje, Selenium Web Driver versión 3.141.0 y Selenium Chrome Web Driver versión 91.0.4472.19 como librerías de pruebas automatizadas de navegador web y xUnit 2.4.0 como librería para ejecución de pruebas unitarias. El código fuente desarrollado se encuentra alojado en un repositorio en Github[26].

Sistema Operativo	Memoria RAM	CPU	Navegador
Windows 10 Enterprise Version 1909 (OS Build 18363.1556)	16 GB	Intel Core i7-8665U	ChromeDriver 91.0.4472.19

Tabla 13 Ambiente de pruebas automatizadas

Relación metamórfica #1

La Tabla 14 lista los datos de la ejecución de la prueba para la MR #1. En las Figuras 31, 32, 33 y 34, pueden verse los resultados obtenidos del uso del buscador para cada iteración.

MR	Iteración	Caso	Entrada	Resultado	Cumple la MR	Fecha de ejecución
# 1	1	Origen	big data aplicado al marketing digital	Cerca de 948.000.000 resultados	Si	24/05/2021
		Seguimiento	“big data aplicado al	Cerca de 997 resultados		

[26] https://github.com/nicolasgranata/testing-metamorfico

			marketing digital”			
# 1	2	Origen	hisopado covid	Cerca de 5.390.000 resultados	Si	24/05/2021
		Seguimiento	“hisopado covid”	Cerca de 9.460 resultados		

Tabla 14 Ejecución MR #1

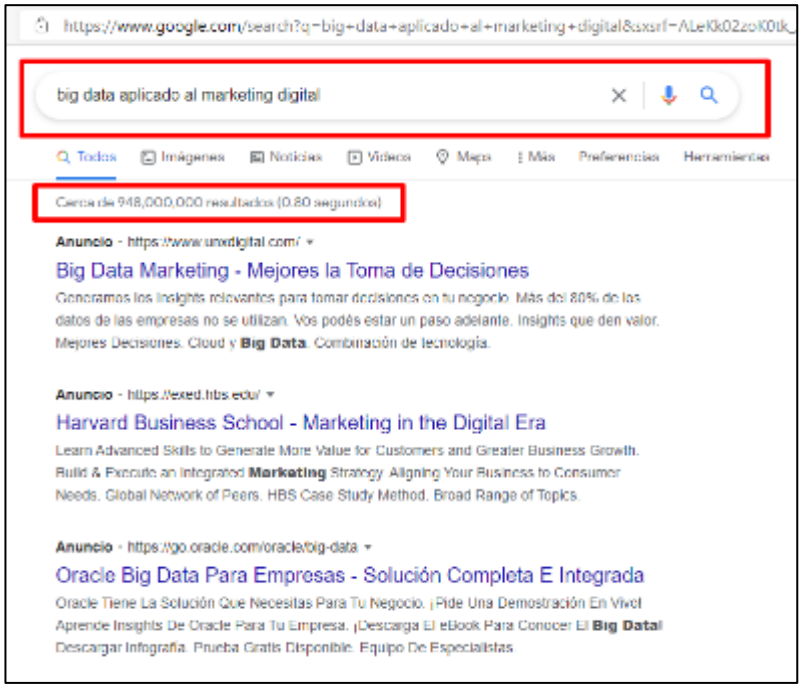

Figura 31 MR #1 Iteración 1 Caso de origen

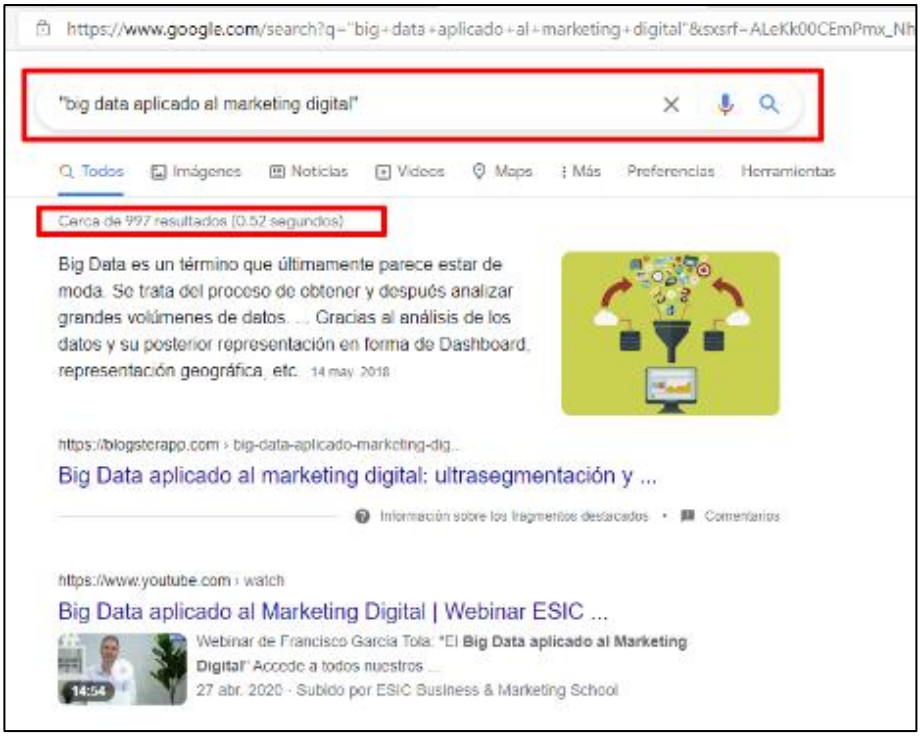

Figura 32 MR #1 Iteración 1 Caso de seguimiento

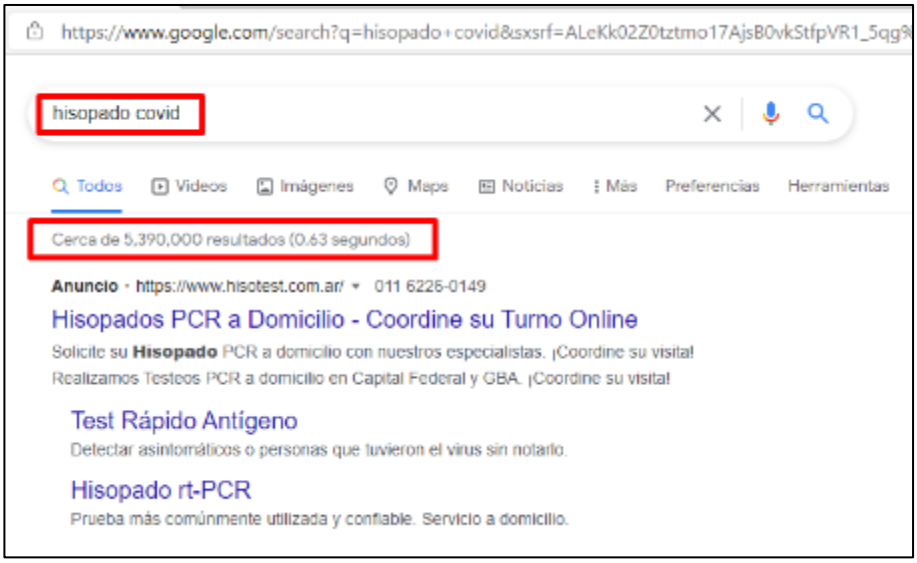

Figura 33 MR #1 Iteración 2 Caso de origen

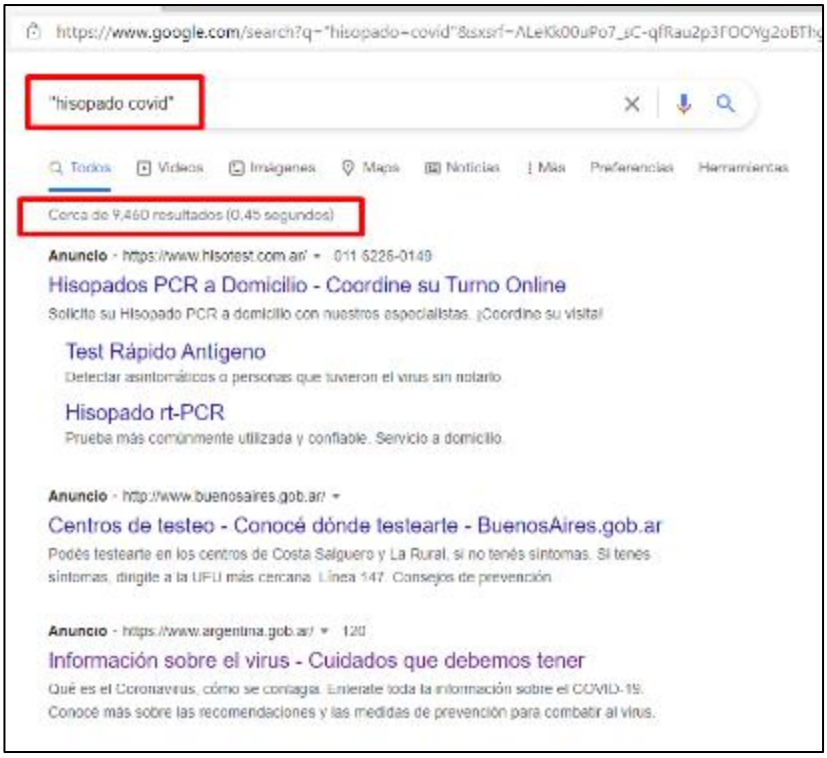

Figura 34 MR #1 Iteración 2 Caso de seguimiento

Para automatizar la prueba de la MR #1, como se muestra en la Figura 35, se creó una clase llamada "*SearchService*" la cual cuenta con el método "*CountSearchResults*" el cual recibe el texto de la consulta a realizar en el buscador y devuelve la cantidad de resultados encontrados. La Figura 36, muestra la prueba llamada "MR1Iteration1Test", que ejecuta la relación metamórfica #1. En la Figura 37 se puede ver como dicha prueba cumple con la MR.

El código fuente puede descargarse de https://github.com/nicolasgranata/testing-metamorfico.

```
7   namespace TestingMetamorfico.Services
8   {
        1 reference
9       public class SearchService
10      {
            2 references
11          public long CountSearchResults(string searchQuery)
12          {
13              IWebDriver driver = new ChromeDriver(AppDomain.CurrentDomain.BaseDirectory);
14              driver.Navigate().GoToUrl("http://www.google.com/");
15              driver.Manage().Window.Maximize();
16
17              var searchBox = driver.FindElement(By.Name("q"));
18              searchBox.SendKeys(searchQuery);
19              searchBox.SendKeys(Keys.Enter);
20
21              var resultStats = driver.FindElement(By.Id("result-stats"));
22              var textResult = Regex.Matches(resultStats.Text, @"\d+").SkipLast(2).Select(x => x.Value);
23              var result = long.Parse(textResult.Aggregate((s1, s2) => s1 + s2));
24
25              return result;
26          }
27      }
28  }
29
```

Figura 35 Clase SearchService

```
4   namespace TestingMetamorfico.Tests
5   {
        0 references
6       public class SearchServiceTest
7       {
8           [Fact]
            0 references
9           public void MR1Iteration1Test()
10          {
11              // Arrange
12              var searchService = new SearchService();
13              var input1 = "big data aplicado al marketing digital";
14              var input2 = ""big data aplicado al marketing digital"";
15
16              // Act Source test case
17              var resultIteration1 = searchService.CountSearchResults(input1);
18              // Act Follow-up test case
19              var resultIteration2 = searchService.CountSearchResults(input2);
20
21              // Metamorphic relation #1 assertion
22              Assert.True(resultIteration1 > resultIteration2);
23          }
24      }
25  }
```

Figura 36 Código de la prueba MR #1

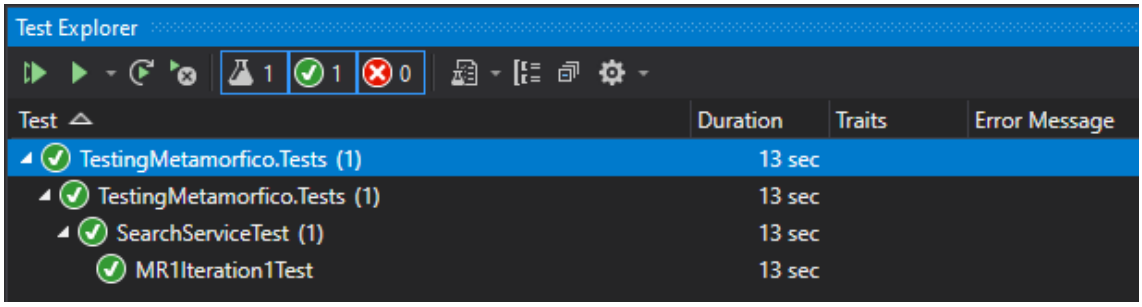

Figura 37 Ejecución de la prueba MR #1

Relación metamórfica #2

La Tabla 15 lista los datos de la ejecución de la prueba para la MR #2. En las Figuras 38, 39, 40 y 41, pueden verse los resultados obtenidos del uso del buscador para cada iteración.

MR	Iteración	Caso	Entrada	Resultado	Cumple la MR	Fecha de ejecución
# 2	1	Origen	big data aplicado al marketing digital	Cerca de 948.000.000 resultados	Si	24/05/2021
		Seguimiento	big data aplicado al marketing digital site:blogsterapp.com	Cerca de 27 resultados		

# 2	2	Origen	“hisopado covid”	Cerca de 9.460 resultados	Si	24/05/2021
		Seguimiento	“hisopado covid” site:labmolecular.com.ar	Cerca de 51 resultados		

Tabla 15 Ejecución MR #2

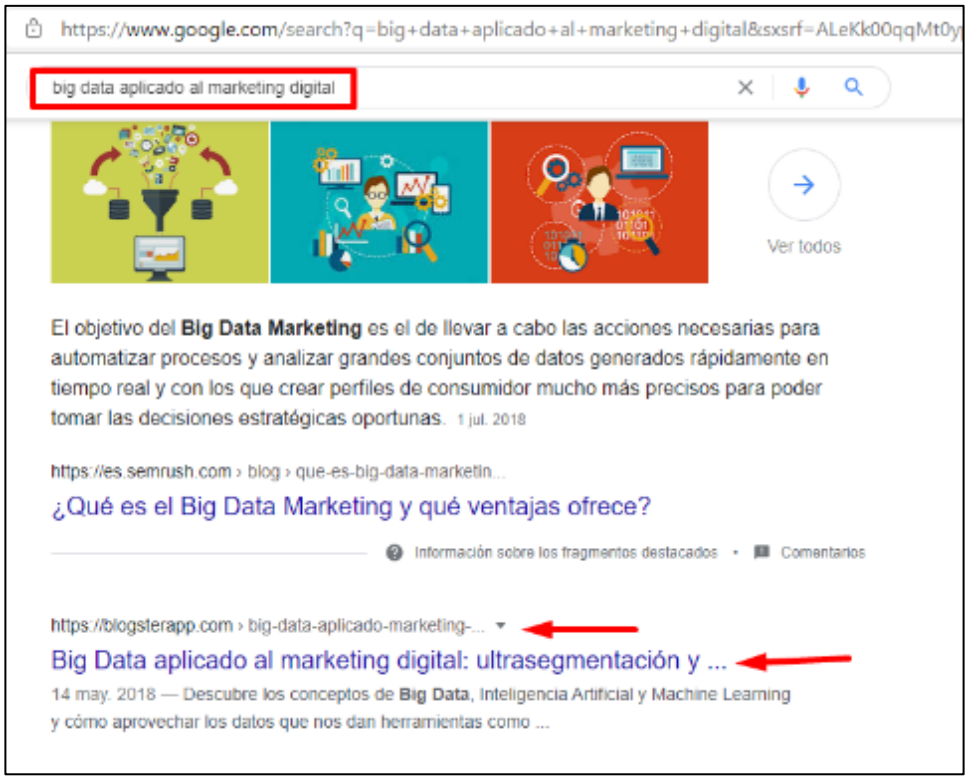

Figura 38 MR #2 Iteración 1 Caso de origen

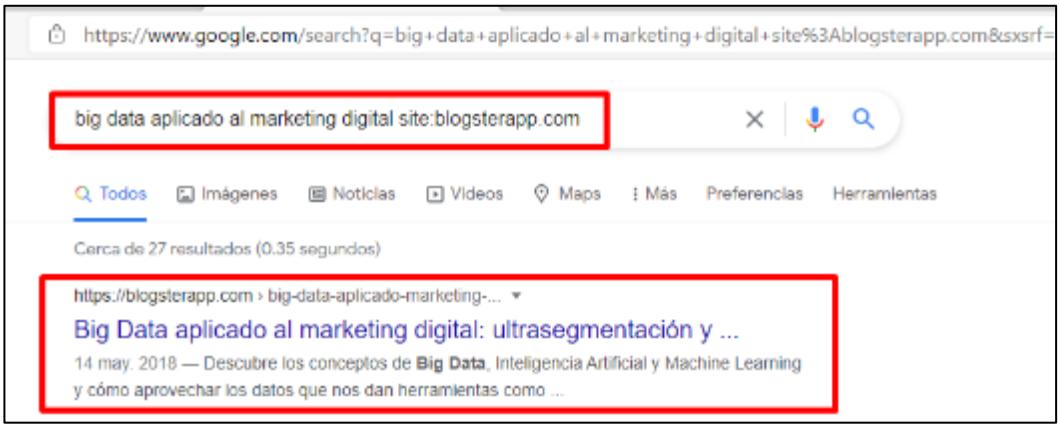

Figura 39 MR #2 Iteración 1 Caso de seguimiento

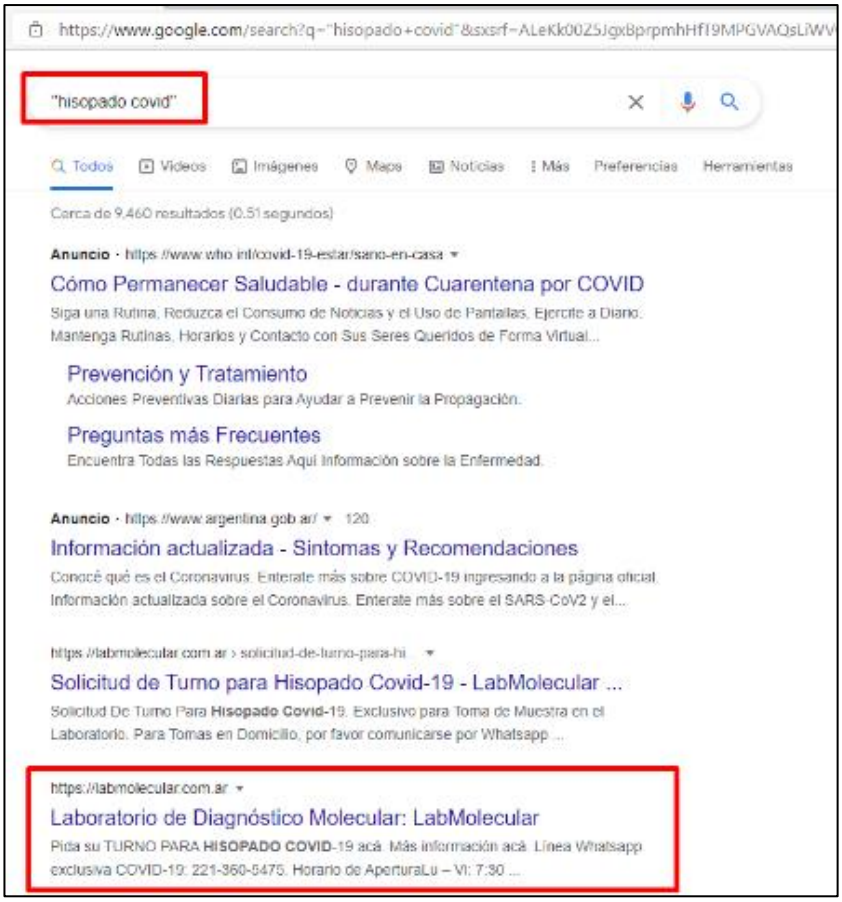

Figura 40 MR #2 Iteración 2 Caso de origen

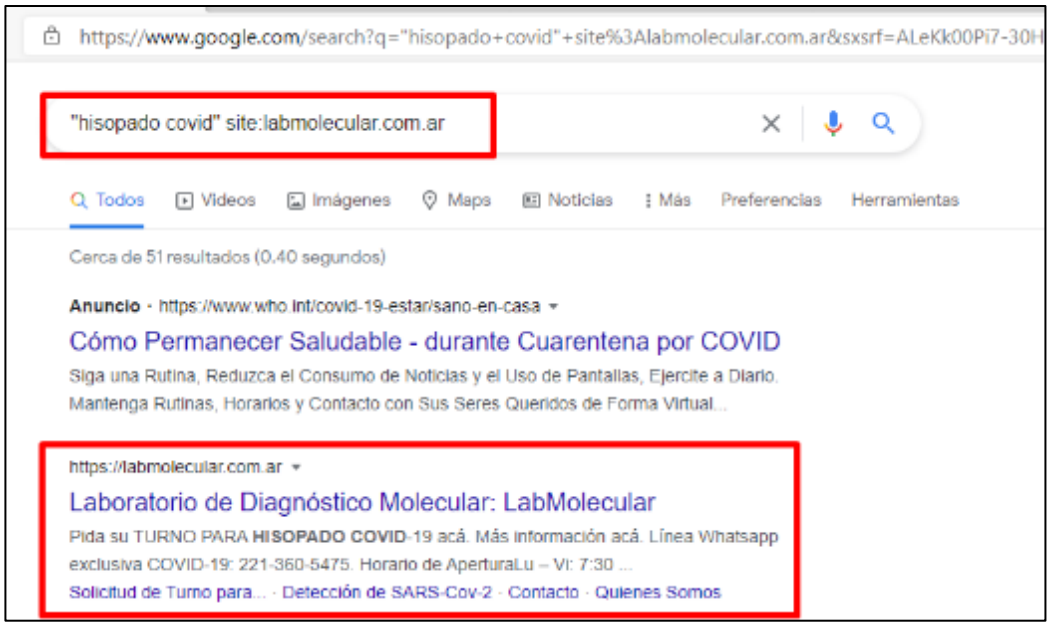

Figura 41 MR #2 Iteración 2 Caso de seguimiento

Relación metamórfica #3

La Tabla 16 lista los datos de la ejecución de la prueba para la MR #3. En las Figuras 42, 43, 44 y 45, pueden verse los resultados obtenidos del uso del buscador para cada iteración.

MR	Iteración	Caso	Entrada	Resultado	Cumple la MR	Fecha de ejecución
# 3	1	Origen	“motos” AND “heladera” AND “elefante”	Cerca de 45.400 resultados	Si	24/05/2021
		Seguimiento	“heladera” AND “elefante” AND “motos”	Cerca de 43.900 resultados		
# 3	2	Origen	"cafe" AND "martillo" AND "piloto"	Cerca de 324.000 resultados	Si	24/05/2021
		Seguimiento	"martillo" AND "piloto" AND "cafe"	Cerca de 325.000 resultados		

Tabla 16 Ejecución MR #3

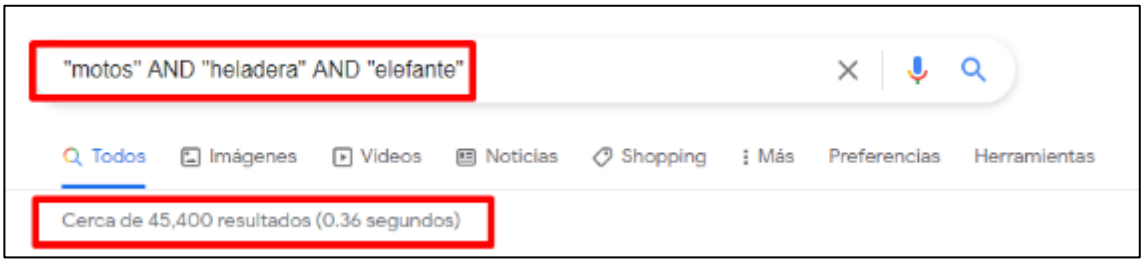

Figura 42 MR #3 Iteración 1 Caso de origen

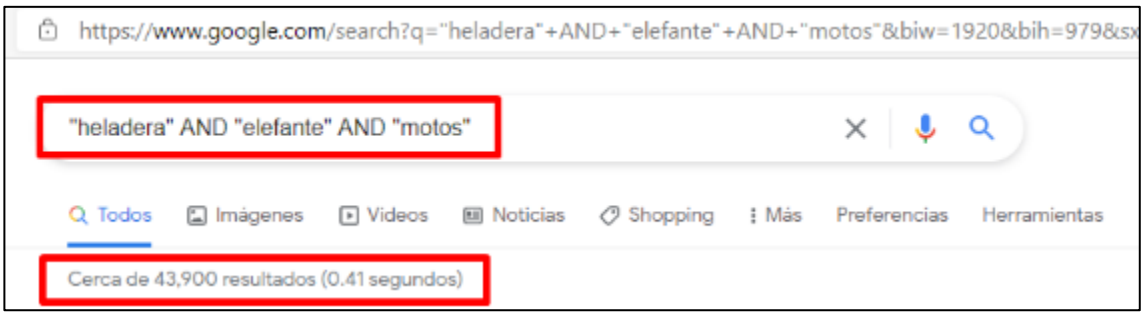

Figura 43 MR #3 Iteración 1 Caso de seguimiento

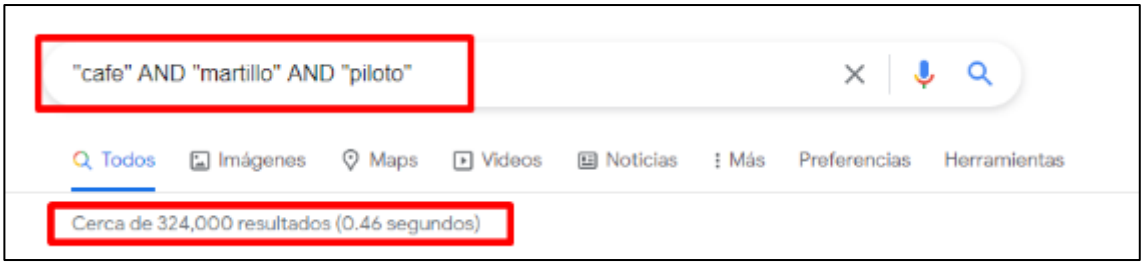

Figura 44 MR #3 Iteración 2 Caso de origen

Figura 45 MR #3 Iteración 2 Caso de seguimiento

Relación metamórfica #4

La Tabla 17 lista los datos de la ejecución de la prueba para la MR #4. En las Figuras 46, 47, 48 y 49, pueden verse los resultados obtenidos del uso del buscador para cada iteración.

MR	Iteración	Caso	Entrada	Resultado	Cumple la MR	Fecha de ejecución
# 4	1	Origen	velocidad del jaguar	Cerca de 14.600.000 resultados	Si	24/05/2021
		Seguimiento	velocidad del jaguar -auto	Cerca de 4.080.000 resultados		
# 4	2	Origen	autos en venta	Cerca de 411.000.000 resultados	Si	24/05/2021
		Seguimiento	autos en venta - nuevo	Cerca de 68.900.000 resultados		

Tabla 17 Ejecución MR #4

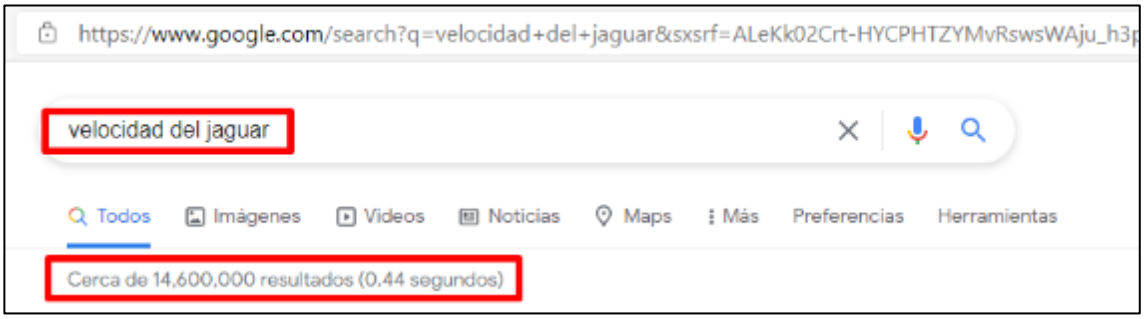

Figura 46 MR #4 Iteración 1 Caso de origen

Figura 47 MR #4 Iteración 1 Caso de seguimiento

Figura 48 MR #4 Iteración 2 Caso de origen

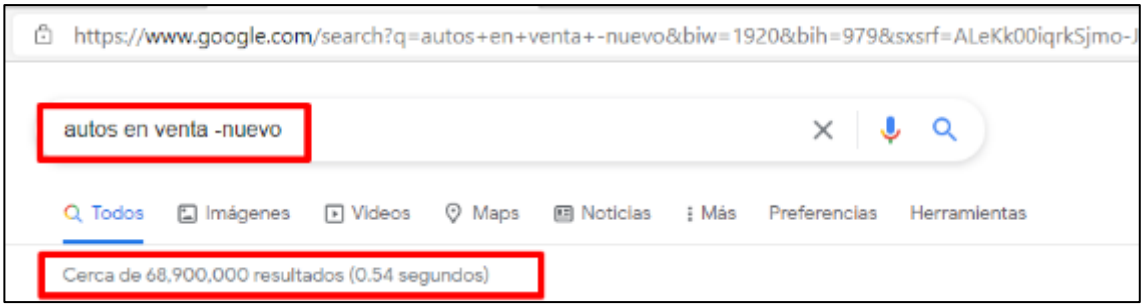

Figura 49 MR #4 Iteración 2 Caso de seguimiento

Relación metamórfica #5

La Tabla 18 lista los datos de la ejecución de la prueba para la MR #5. En las Figuras 50, 51, 52 y 53, pueden verse los resultados obtenidos del uso del buscador para cada iteración. En la Figura 54 puede verse que el caso de seguimiento alternativo no cumple la MR, por lo cual debe refinarse la relación.

MR	Iteración	Caso	Entrada	Resultado	Cumple la MR	Fecha de ejecución

# 5	1	Origen	“café de especialidad buenos aires”	Cerca de 470 resultados	Si	24/05/2021
		Seguimiento	“café de especialidad buenos aires” - especialidad	No se encontraron resultados		
# 5	2	Origen	"cadena de distribución autos"	Cerca de 393 resultados	Si	24/05/2021
		Seguimiento	"cadena de distribución autos" - cadena	No se encontraron resultados		

Tabla 18 Ejecución MR #5

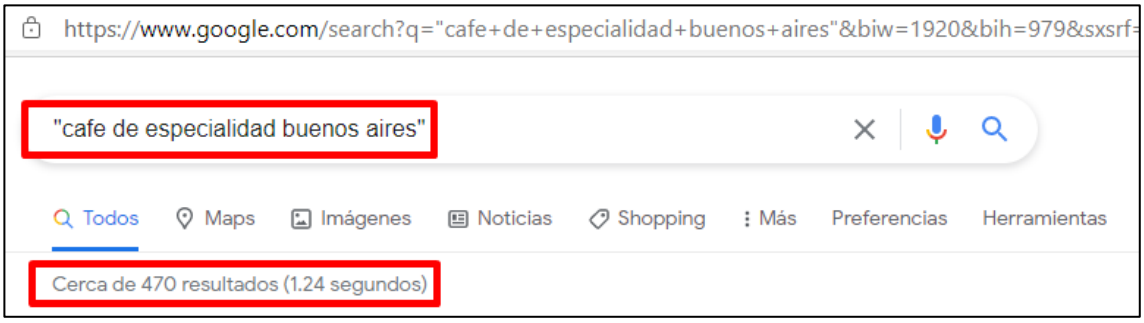

Figura 50 MR #5 Iteración 1 Caso de origen

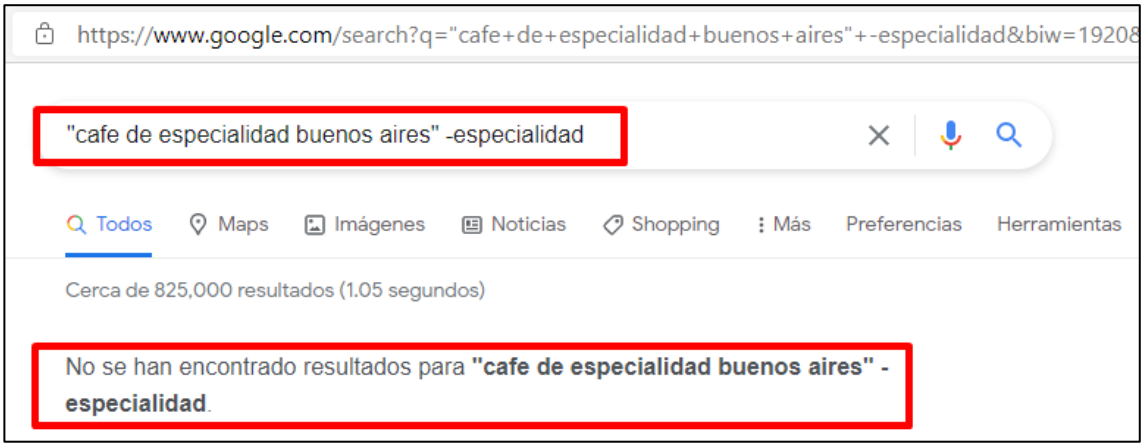

Figura 51 MR #5 Iteración 1 Caso de seguimiento

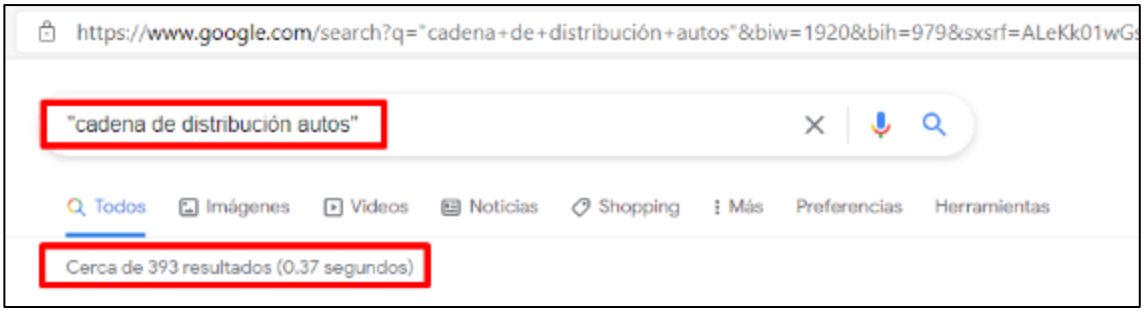

Figura 52 MR #5 Iteración 2 Caso de origen

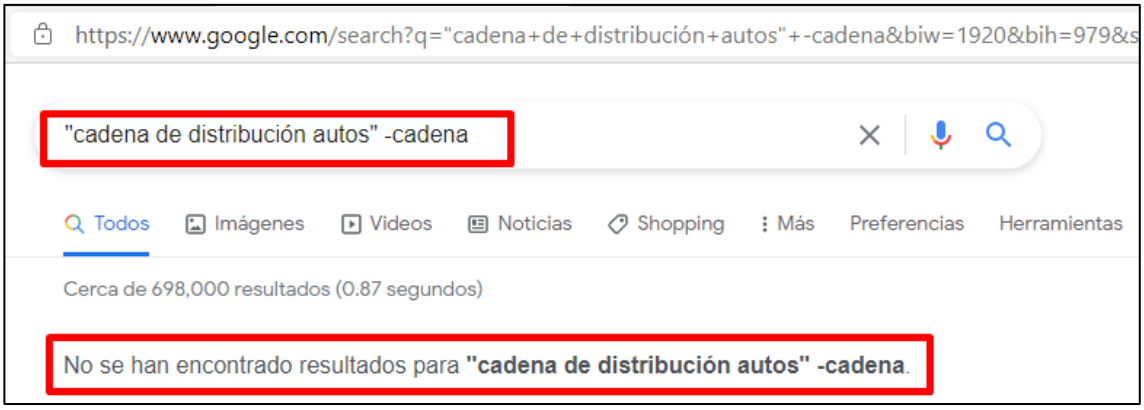

Figura 53 MR #5 Iteración 2 Caso de seguimiento

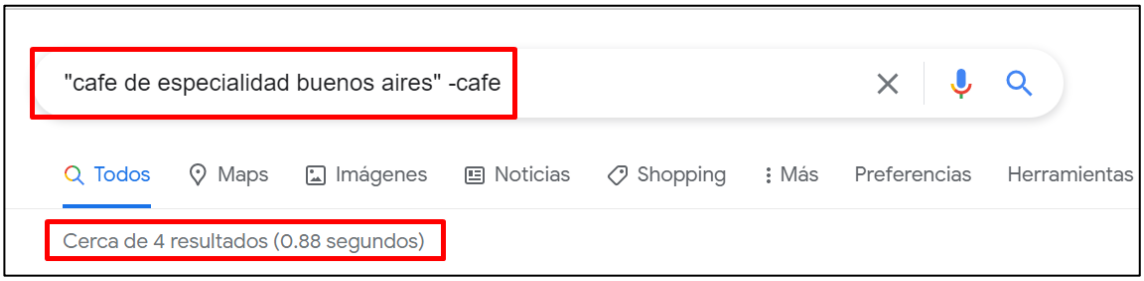

Figura 54 MR #5 Iteración 1 Caso de seguimiento alternativo