RIC

Implementación de un sistema intérprete de la experiencia del cliente en tiempo real bajo plataformas escalables de redes neuronales y aprendizaje profundo

Implementation of an interpreter system of the client experience in real time under scalable neural networks and deep learning platforms

Jaime Villar Ortega[1] & Euclides Samaniego González[2*]

[1]Licenciatura en Ingeniería de Sistemas y Computación – Facultad de Ingeniería de Sistemas Computacionales – Universidad Tecnológica de Panamá

[2]Grupo de Investigación de Inteligencia Computacional – GIICOM – Facultad de Ingeniería de Sistemas Computacionales – Universidad Tecnológica de Panamá

Resumen En este trabajo se implementa un sistema con la capacidad de interpretar en cierta medida los estados de satisfacción y el ánimo del cliente, tomando datos generados por una interfaz de visión por computadora integrada en una aplicación móvil y registrada en una base de datos para luego ser cargados y desplegados en tiempo real por una plataforma web. Se tiene como objetivo implementar este sistema intérprete empleando plataformas escalables y accesibles de redes neuronales y aprendizaje profundo. Así como también de analizar los sistemas actuales para medir la satisfacción de clientes, sus ventajas y desventajas. De esta manera demostrar como este proyecto podría funcionar como complemento a las metodologías actuales usadas para determinar la satisfacción de los clientes.

Palabras claves Redes neuronales, visión por computadora, intérprete, tiempo real.

Abstract This project implements a system with the ability to interpret the satisfaction and mood of the client, taking data generated by a computer vision interface integrated in a mobile application and registered in a database to be tare loaded and deployed in real time by a web application. This project aims to implement thus interpreter system using scalable and accessible platforms of neural networks and deep learning. As well analyse the current systems and methodologies used to measure the customer satisfaction, its advantage and disadvantages. Demonstrate how this project could work as a complement to the current methodologies used to determine customer satisfaction.

Keywords Neural networks, computer vision, interpreter, real-time.

*** Corresponding author:** euclides.samaniego@utp.ac.pa

1. Introducción

En la implementación de un sistema intérprete de la experiencia del cliente en tiempo real se busca dotar al sistema con la capacidad de interpretar los estados de satisfacción y el ánimo del cliente, tomando datos generados por una interfaz de visión por computadora integrada en una aplicación móvil y registrados en una base de datos para luego ser cargados y desplegados en tiempo real por una plataforma web.

El principal propósito del sistema es el de desplegar datos en tiempo real provenientes de las características faciales hechas por un cliente, captadas por la aplicación móvil y un módulo de detección de rostros.

Existen distintas herramientas empleadas en la actualidad que buscan medir la satisfacción de clientes. Dentro de este proyecto se estudian y aplican herramientas de *software* como servicio para este propósito. Se presentan las principales estructuras de comunicación, se describen los principales algoritmos usados en el aprendizaje automático y sus distintos enfoques, así como también las principales plataformas escalables que emplean tecnologías de aprendizaje profundo.

La detección de las emociones por medio de inteligencia artificial es el marco principal de este trabajo, enfocado específicamente a la atención de clientes por medios digitales.

Gracias a que los medios digitales y en la nube han ido en incremento, los sistemas son capaces de manejar un lenguaje más natural y aprenden de nosotros a cada instante. Sin embargo, para que un sistema pueda evaluar características humanas como las emociones, este antes debe emular el pensamiento humano.

1.1 Antecedentes

Actualmente las medidas tomadas para regular y tratar de controlar el tema de la satisfacción del cliente van desde lo tradicional al empleo de nuevas tecnologías para lograr optimizar estos procesos a mayor escala.

Siendo lo tradicional el uso de encuestas de satisfacción y las nuevas opciones como el uso de algoritmos de inteligencia artificial para tratar de predecir las necesidades puntuales de los clientes. Hoy en día existen en el mercado algunas opciones en cuanto a las soluciones que emplean tecnologías de aprendizaje automático como métodos para minimizar la insatisfacción de los clientes prediciendo las posibles variables, tratando así de mejor el servicio al cliente.

Por otro lado, el servicio y la atención al cliente por medios digitales presenta un gran crecimiento y se pronostican aumentos exponenciales durante los próximos años en lo que respecta a llevar el servicio al cliente bajo estos medios, surgiendo así cada día nuevos requerimientos y la necesidad de crear nuevos controles para estos casos.

Es por ello que surgen varias interrogantes y tema de investigación:

- ¿Son los sistemas actuales realmente eficaces para tratar los temas de satisfacción del cliente?
- ¿Qué implica llevar una atención al cliente desde un medio digital y cuáles serían las métricas utilizadas para evaluar la experiencia?
- ¿Es posible medir por medio de los mecanismos actuales y de manera óptima la experiencia del cliente sin que exista una retroalimentación verbal o escrita por parte de este?

Son estas algunas de las interrogantes que impulsan esta propuesta y proyecto.

1.2 Caracterización de la problemática

La insatisfacción del cliente es una problemática constante en comercios, entidades públicas y privadas. Desde el pequeño comerciante hasta las grandes corporaciones no se encuentran exentos de este problema, el cual puede generar consecuencias de gran riesgo para las actividades comerciales y de otras índoles que pueda tener la empresa donde reflejen una insatisfacción de sus clientes al momento de recibir los servicios o productos que ofrecen.

Un mal servicio puede acarrear problemas de lealtad hacia una marca, la pérdida de clientes

actuales, se puede crear una mala imagen y hasta destruir un producto exitoso, ya sea por una mala atención hecha pública en redes sociales que puede provocar la pérdida inmediata de posibles clientes potenciales a causa del ya conocido poder de difusión en estas redes.

Dado el gran auge de las telecomunicaciones, la atención del cliente al ser realizada desde un ambiente digital, por ejemplo, los videollamadas, tampoco escapan de estas problemáticas ya mencionadas.

Al contrario, si no son manejadas y controladas de la manera correcta pueden desencadenar grandes bajas de clientes.

Cada día más empresas y comercios suman estas opciones tecnológicas a las comunicaciones con sus clientes y muchas de estas carecen de controles adecuados o ignoran por completo la toma de procedimientos bajo estas modalidades, dejando así canales mediocres, aumentando la insatisfacción y la pérdida de clientes, así como también un desmejoramiento en las actividades sociales y comerciales.

1.3 Justificación

Debido al auge en el empleo de los dispositivos móviles y sus aplicaciones para gran parte de las actividades cotidianas y las telecomunicaciones, es requerida la elaboración de nuevos controles en las áreas ya mencionadas, como la atención al cliente.

Las empresas y entidades que buscan mejorar sus estándares de calidad están presentando vacíos de información y la falta de controles en cuanto a las experiencias de clientes que se comunican con ellos por medios digitales.

Este proyecto busca servir de complemento a estos problemas y carencias, aprovechando las tecnologías ya existentes, su escalabilidad, accesibilidad y gran flexibilidad para ajustarse a cada requerimiento.

Brindándole así a estas entidades, nuevas opciones y formas de cómo mejorar su servicio al cliente.

Demostrándoles por otra parte, la importancia de mantener controles y de llevar un seguimiento minucioso de las experiencias de los clientes con respecto a las labores de servicios por medio de técnicas posiblemente desconocidas.

1.4 Restricciones y limitaciones

Las herramientas y plataformas escalables utilizadas para el desarrollo del proyecto deben ser utilizadas bajo las modalidades gratuitas y de planes por demanda.

Hardware y *software* mínimo requerido en dispositivos empleados.

Poseer conocimientos previos en el desarrollo móvil y web en un nivel intermedio-avanzado.

El desarrollo móvil es posible en múltiples plataformas, sin embargo, para este proyecto se desarrollará únicamente en la plataforma de Android para poder cumplir con el cronograma y los tiempos establecidos.

La plataforma y aplicativo web se montará en un servidor local a modo de pruebas. Evitando así costes de alojamiento.

Se emplearán los mecanismos de inteligencia artificial para cumplir con lo mínimo requerido por el proyecto. Se debe contar con velocidades de conexión que soporten al menos videollamadas por Internet.

Los algoritmos desarrollados serán en base a la lógica personal, usando conocimientos generales recaudados en la investigación. Para este proyecto no se cuenta con la ayuda de expertos en las áreas relacionadas.

1.5 Objetivos

1.5.1 Objetivo general

Implementar un sistema intérprete de la experiencia del cliente en tiempo real, empleando plataformas escalables de redes neuronales y aprendizaje profundo.

1.5.2 Objetivos específicos

- Identificar que herramientas para la medición y control de la satisfacción del cliente son utilizadas por las empresas y/o comercios en la actualidad.

- Analizar las ventajas y desventajas que estas herramientas pueden ofrecer.
- Proponer y demostrar el uso de nuevos controles en la detección y medición de la satisfacción del cliente.
- Identificar por medio de investigaciones cualitativas y cuantitativas los beneficios reales que pueda tener la propuesta e hipótesis formulada dentro de esta investigación, hacia las empresas, comercios y sistemas de atención al cliente.
- Integrar de manera óptima las herramientas *web* y tecnologías móviles adecuadas dentro de la implementación del proyecto.

2. Herramientas

2.1 Herramientas de satisfacción del cliente

La satisfacción de los clientes ha sido desde siempre un pilar importante dentro de las políticas comerciales en las organizaciones.

Desde la elaboración de las normas ISO 9000 para la gestión de la calidad, las organizaciones y empresas que las incluyen es sus estándares han tratado de seguir estos lineamientos a detalle, obteniendo en gran medida buenos resultados.

Si bien es cierto, el aplicar estos estándares y políticas pueden representar un gran costo, la inversión realizada en la mayoría de los casos le garantiza a la organización el poder continuar operando de manera estable.

El servicio al cliente desde medios digitales se ha tornado en una situación habitual hoy en día, a tal punto de hasta llegar a desplazar a los métodos tradicionales en ciertas áreas. Es por esto que las empresas han comenzado a invertir más recursos para poder cubrir esta necesidad que viene evolucionando.

Existen varios tipos de herramientas para medir la satisfacción del cliente que pueden ser aplicadas desde un medio tecnológico como los son: (a) buzón de sugerencias, (b) panel y las (c) encuestas.

Para llevar un control de la satisfacción de los clientes, las empresas han optado por invertir en el análisis de datos de sus clientes. Estos datos pueden obtenerse desde páginas web, redes sociales o aplicaciones móviles.

Entre las métricas más conocidas y empleadas en los medios digitales para poder calcular la satisfacción del cliente: (a) *Customer Satisfaction Score* (CSAT), (b) *Net Promoter Score* (NPS), (c) *Customer Effort Score* (CES), entre otras.

2.2 Herramientas de *software* como servicio

Para hablar de *software* como servicio es necesario introducir el concepto de tercerización.

Este concepto está basado en la transferencia de un servicio interno a un proveedor externo. SaaS (*Software as a Service*), es decir *software* como servicio, pertenece a un conjunto de soluciones en la nube para la administración de servicios.

La escalabilidad que ofrecen estos servicios es alta, esto los convierte en un factor clave para aplicaciones que requieran crecer de un momento a otro ya sea por mayor demanda o la inclusión de nuevos módulos.

Dentro de las herramientas de *software* como servicio se encuentras distintas infraestructuras y motores de comunicación disponibles ya preparadas para soportar cualquier tipo de aplicación móvil o *web* que requiera de los beneficios de estos servicios como el caso de este proyecto.

2.3 Herramientas para el reconocimiento de patrones y aprendizaje automático

El empleo de la tecnología en prácticamente todos los sectores sociales y comerciales ha impulsado el crecimiento del uso de la inteligencia artificial aplicada.

Es por esto que grandes empresas dentro del sector tecnológico han empezado a ofrecer sus soluciones SaaS y herramientas de desarrollo.

Entre ellas se encuentra la plataforma de *Google Cloud* donde se ofrecen toda una serie de herramientas pero en especial distintas APIs para el procesamiento de información mediante aprendizaje automático como: (a) *Cloud Machine Learning Platform,* (b) Vision API, (c) *Speech* API, (d) *Natural Language* API y (e) *Translate* API.

El aprendizaje profundo es una sub-rama del aprendizaje automático en donde se emplean algoritmos que simulan la estructura y funciones del cerebro, esto es mayormente empleado para el procesamiento del lenguaje natural.

2.3.1 *Mobile Vision* API

Esta API forma parte del conjunto de herramientas ofrecidas dentro del paquete de *Google Play Services,* que no son más que librerías de extras que ofrece *Google* para el desarrollo de aplicaciones Android.

Esta API permite la detección de rostros en imágenes y videos. Además también ofrece la lectura de códigos de barra y reconocimiento de caracteres en imágenes y videos para convertirlos en texto, todo en tiempo real.

Esta herramienta cuenta con la capacidad de detectar los rostros en distintas orientaciones y los rasgos de la cara como los ojos, nariz y la boca. Ver figura 1.

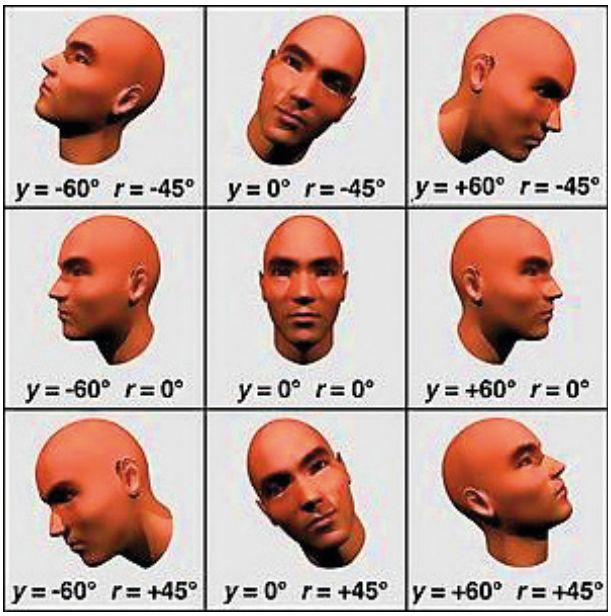

Figura 1. Orientación del rostro en Mobile Vision API.

Al momento en que la API detecta un rostro, este retorna un objeto Face. Este objeto provee toda una data espacial sobre el rostro detectado.

Algunos de los métodos que utiliza son:

- (a) *getPosition*(), que retorna la coordenada superior izquierda del área donde el rostro fue detectado.
- (b) *getWidth*(), que retorna el ancho del área donde el rostro fue detectado.
- (c) *getHeight*(), que retorna la altura del área donde el rostro fue detectado.
- (d) *getId*(), que retorna un identificador asociado al rostro detectado.

De igual manera existen los siguientes métodos para determinar la orientación del rostro:

- (a) *getEulerY*(), que retorna la rotación del rostro en el eje vertical.
- (b) *getEulerZ*(), que retorna la rotación del rostro en base al eje Z.

La clasificación dentro del objeto Face determina cuanto una característica facial está presente. Actualmente el API soporta dos tipos de clasificaciones: (i) ojos abiertos y (ii) sonriendo.

Estos valores van de 0 a 1 y pueden ser tomados como un porcentaje, por ejemplo, que tan sonreída se encuentra una persona.

Los métodos que se encargan de retornar lo mencionado son los siguientes:

- (a) *getsLeftEyeOpenProbability*(), que retorna la probabilidad de que el ojo izquierdo se encuentre abierto.
- (b) *getsRightEyeOpenProbability*(), que retorna la probabilidad de que el ojo derecho se encuentre abierto.
- (c) *getIsSmilingProbability*(), retorna la probabilidad de que el rostro se encuentre sonriendo.

3. Detección de emociones

3.1 Metodologías

Las emociones son características innatas en los seres humanos, es decir, siempre estarán presentes en distintas maneras y niveles en cada individuo.

Existen muchos parámetros físicos presentes en una emoción, como lo son la piel, las expresiones faciales, la distribución de la sangre, el ritmo cardíaco, la respiración, la respuesta pupilar, secreción salival y movilidad gastrointestinal son algunas muy comunes.

3.2 Sistema de codificación de acciones faciales (FACS)

Este sistema fue desarrollado por Ekman y Friesen en el año 1978, que asociaron movimientos en el rostro y marcaron ciertas variables.

El FACS o Sistema de codificación de acciones faciales fue elaborado para la detección de cambios en las expresiones faciales, sean las notorias o las micro expresiones. Estas acciones pueden ocurrir de forma individual o simultáneamente.

La combinación de estas acciones vendría representando una expresión específica y se deduce que ciertas expresiones o emociones son derivadas de otras. Véase figura 2.

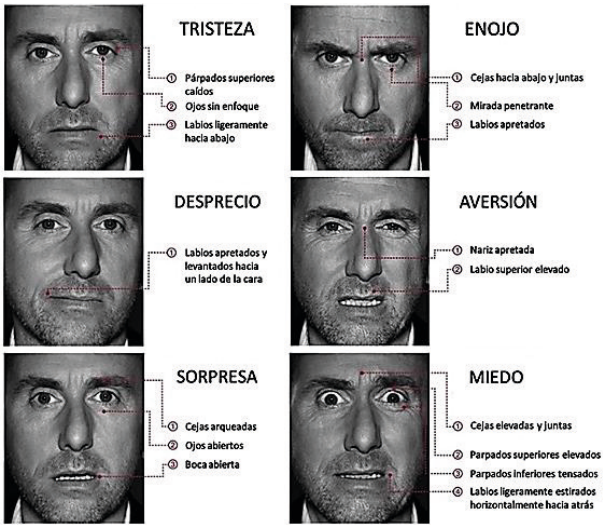

Figura 2. FACS – Sistema de codificación de las acciones faciales.

Cabe destacar que las investigaciones realizadas por Ekman demuestran que la asimetría en rostros denota una expresión simulada o incomodidad. La asimetría es producida en expresiones voluntarias.

4. Análisis y diseño del sistema intérprete

4.1 Análisis

Al escoger la metodología que se emplearía se tomaron en cuenta factores como los costos de tecnologías, las documentaciones existentes, si era de una tecnología de código abierto y el tiempo de desarrollo e integración que conlleva.

La detección de expresiones faciales por medio de las cámaras es una tecnología que se encuentra al alcance de los desarrolladores de *software* con multitud de plataformas y opciones para escoger. De igual manera, esta tecnología se ha logrado integrar de manera óptima en los dispositivos Android, haciéndola propicia para el desarrollo del proyecto.

Las herramientas utilizadas en la elaboración del proyecto fueron seleccionadas en base a su tipo de código, su escalabilidad, costo, documentación y fueron evaluadas en este preciso orden.

4.1.1 Infraestructura de comunicación

Para el desarrollo de la comunicación entre los usuarios del aplicativo en las plataformas *web* y móvil se decidió integrar *QuickBlox*.

Esta herramienta de código abierto permitió una integración sencilla entre los aplicativos y se agilizaron los desarrollos gracias a toda la documentación y soporte existente característico de los desarrollos con código abierto.

Su flexibilidad multiplataforma permitió el enlace entre el aplicativo móvil y la plataforma *web* sin mayor complicación, empleando los ya conocidos protocolos de *webrtc.* Tomando las bases de sus programas de ejemplo integrados en las plantillas de *Q-municate* y toda su documentación fue posible integrar el motor de comunicación y adicionalmente generar un almacenamiento en la nube con datos obtenidos desde el aplicativo móvil.

4.1.2 Plataforma de visión por computadora

Dentro del aplicativo móvil desarrollado se han integrado unos módulos con la capacidad de detectar los niveles de sonrisa y movimientos de la cabeza de los usuarios.

El API escogido para este propósito fue el de *Mobile Vision* ofrecido por *Google.* Se ha hecho así debido a que esta API no requiere de llamados constantes a servidores externos y emplea librerías para la detección facial de manera local.

Estas librerías han sido optimizadas para los dispositivos móviles y cuentan con

actualizaciones regulares. Su principal desventaja y limitante es la poca o inexistente integración con otras plataformas móviles como *iOS*.

Para estos casos que se podrían encontrar en una posible futura etapa las otras dos opciones mencionadas cuentan con toda la infraestructura requerida y su integración es posible para todas las plataformas móviles y *web*.

Para la selección del API de visión por computadora el factor de mayor relevancia fue los costos de los servicios, ya que la integración en la plataforma de *Android* era de una complejidad similar, contando *Mobile Vision* con la ventaja de tener una documentación exclusiva para esta plataforma, ser completamente gratuita y accesible para todos los desarrolladores del ambiente Android.

4.1.3 API de gráficos

En cuanto al desarrollo del lado web, adicional a la infraestructura de comunicación ofrecida por *QuickBlox* se integraron unas librerías gráficas con la capacidad de desplegar información en tiempo real y desplegar varios tipos de gráficos.

Para este propósito se ha trabajado con PubNub, los cuales ofrecen la librería de código abierto EON, ya mencionada anteriormente.

El principal motivo para ser escogida fue su sencilla integración y modelos de datos en el lenguaje de *JavaScript* además de ser una herramienta de código abierto con excelente soporte para el uso de documentos que manejen datos.

4.2 Arquitectura de las aplicaciones

4.2.1 Aplicación móvil

Para el desarrollo de la aplicación móvil se emplearon tecnologías de código abierto en la plataforma Android, usando código nativo y el entorno de desarrollo oficial de *Android Studio*.

La aplicación móvil cuenta con dos componentes principales desde la perspectiva del proyecto elaborado: (i) los módulos de comunicación de *QuickBlox* y *Q-municate* y (ii) los métodos y llamados a la API de *Mobile Vision*.

4.2.2 Plataforma *web*

La plataforma *web* fue desarrollada utilizando *JavaScript* tanto en el módulo de comunicación como en el módulo de gráficos.

Para la plataforma web se empleó al igual que el aplicativo móvil la estructura inicial de *Q-municate* la cual emplea Node.js como estructura principal, *Bower* como administrador de paquetes y *Grunt* para ejecutar las tareas escritas en JavaScript e iniciar los servicios.

Los gráficos en tiempo real fueron desarrollados de manera independiente al módulo de comunicación empleando la librería de código abierto EON ofrecida por *PubNub*.

Al igual fue desarrollado en *JavaScript* con una vista de gráficos en tiempo real y otra en donde se muestran registros con el propósito de servir para pruebas comparativas.

4.3 Flujo básico y casos de uso

En el caso de uso desarrollado para el proyecto, el "cliente", usuario del aplicativo móvil selecciona al contacto de servicio al cliente y realiza una video-llamada para luego ser atendida por el administrador o agente de servicio.

Al momento de establecer la conexión se crea una nueva sesión en los métodos empleados para la detección de rostros y paralelamente se registran cada segundo una instancia del objeto visión en la base de datos alojada en el servicio de *QuickBlox* para luego ser leída en tiempo real por la plataforma *web*.

4.4 Algoritmos empleados

Al realizarse el análisis de los casos de uso, de elegir las herramientas y plataformas adecuadas que se ajustarán al proyecto, seleccionar lo esencial de estas y evaluar la información recopilada en cuanto a las metodologías empleadas para la detección de las emociones se procedió a elaborar los algoritmos necesarios en cada parte del proyecto.

4.4.1 Algoritmos para la aplicación móvil

Dentro del aplicativo móvil fue requerido idear un conjunto de algoritmos para poder manejar los procesos necesarios del Sistema CXi.

En la captura del rostro desde la aplicación móvil se empleó la API de visión. Su algoritmo de funcionamiento es relativamente sencillo. Ver cuadro 1.

Cuadro 1. Algoritmo 1 empleado para cargar el número de sesión

Algoritmo 1: Número de sesión	
1	**Inicio**
2	**Declaración de variables**
	Entero Sesión = 0
3	Crear objeto consulta
4	Definir los parámetros de consulta
5	Enviar consulta
6	Leer respuesta
7	**Si** respuesta es exitosa **Entonces**
8	Sesión = Primer parámetro del campo sesión en respuesta
9	**De lo contrario**
10	Sesión = =
11	**Devolver** Sesión
12	**Fin**

El siguiente de los procesos involucrados al obtener la señal de la cámara con el detector de rostros ya habilitado fue el de obtener las características y rasgos faciales del objeto Face obtenido del API de *Vision.*

Para el desarrollo del algoritmo se declararon las variables correspondientes a las características y valores obtenidos dentro del objeto Face como lo es el nivel de sonrisa, probabilidad de apertura del ojo derecho así como del ojo izquierdo y los grados de movimiento del rostro en los ejes anteriormente mencionados, adicionalmente se inicializó una variable contador y otras variables como *sonrisaTotal* y *porcentajeInteress* encargadas de llevar un total de los datos obtenidos.

Posteriormente se le asignó el valor obtenido por el API a cada variable y a las variables encargadas de los totales de sonrisa e interés se les sumó su valor junto con métodos en procesos recurrentes.

Para el valor de *sonrisaTotal* se empleó un método sencillo que devolviese un porcentaje de los valores obtenidos del objeto *Face.*

En el caso de los valores totales de interés, se empleó un algoritmo basado en los estudios de la simetría en las expresiones faciales, como también en los movimientos del rostro y cabeza.

Se tomaron tres valores principales para evaluar el porcentaje de interés: (i) simetría en ojos, (ii) porcentaje de ojos abiertos y (iii) porcentaje de mirada fija a la pantalla.

Para el cálculo de la simetría en los ojos se tomaron los valores entregados por el API que indican el porcentaje de apertura de cada ojo, de esta forma si el valor obtenido en un ojo era mayor al otro se dividía el valor menor entre el mayor y se multiplicaba por cien para obtener el porcentaje de simetría en ojos.

En el cálculo del porcentaje de ojos abiertos se sumó el valor de ambos ojos y se dividió entre dos. Por último para el cálculo del porcentaje de mirada fija se tomaron los valores de rotación del rostro, tomando en cuenta que en eje Y se tenía un rango de -60 a 60 y de -45 a 45 para el eje Z, donde el valor de cero (0) era el valor medio y punto de referencia que indica un rostro centrado.

Se tomaron los valores del eje Y y eje Z y se estimó un nivel de atención, donde una mayor aproximación a cero (0) indica un mayor porcentaje de atención.

Para obtener el resultado que nos interesa, el porcentaje de interés total, se estimó un porcentaje para cada uno de los tres valores mencionados, siendo un 25% para la simetría en ojos, un 25% los ojos abiertos y un 50% la mirada fija a pantalla o porcentaje de atención.

Continuando el flujo de procesos dentro del aplicativo móvil, fue requerido elaborar un algoritmo que creara un objeto de los datos capturados por *Vision* y los guardara de manera

recurrente durante el preciso instante de la llamada.

Para esto primeramente se realizó un cálculo del promedio de los datos obtenidos, usando la cantidad de veces que se ejecutó la detección de características del rostro durante un segundo, asignando esta data recolectada a variables que se enviarían en un proceso continuo a la base de datos alojada en el servidor de *QuickBlox* con un número determinado de sesión.

Al terminar la llamada se reinicializan los valores y se aumentan en uno el número de sesión. Estos algoritmos mencionados son los principales dentro del aplicativo móvil.

4.4.2 Algoritmos para la plataforma *web*

Para la plataforma *web* el requerimiento del proyecto pide que se despliegue información sobre la experiencia del cliente.

Para esto la única data a manipular dentro de los parámetros de este proyecto serían los datos de sonrisa y de interés para entonces generar un valor que defina el nivel de satisfacción de los clientes.

Es requerido que el sistema decida cuál es nivel de satisfacción del cliente en tiempo real y adicionalmente interprete un resultado al finalizar cada sesión.

Para lo ya mencionado se elaboró un algoritmo basado en un árbol de decisión que pudiese predecir el nivel de satisfacción, tomando como entrada los valores de interés y de sonrisa.

4.5 Diseño

4.5.1 Diseño de la aplicación móvil

En cada plataforma fue requerido un diseño nativo y acorde al tema del Sistema CXi. En el aplicativo *Android* se usaron recursos de imágenes y las herramientas de diseño nativas de la plataforma bajo lenguaje *xml.*

Para el diseño de la aplicación móvil se mantuvieron ciertas estructuras presentes en el esquema de *Q-municate* y se agregaron los recursos personalizados. Ver Figura 3.

Adicionalmente el aplicativo móvil cuenta con la pantalla de video-llamada, la cual cuenta con varias capaz de diseño en donde se alojan los controles y los marcos de video local y remoto. Ver figura 4.

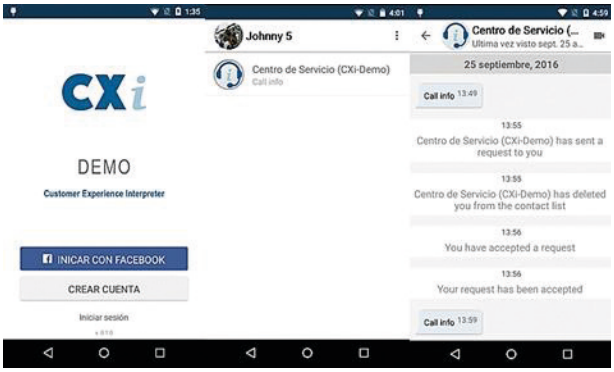

Figura 3. Vista de la aplicación móvil.

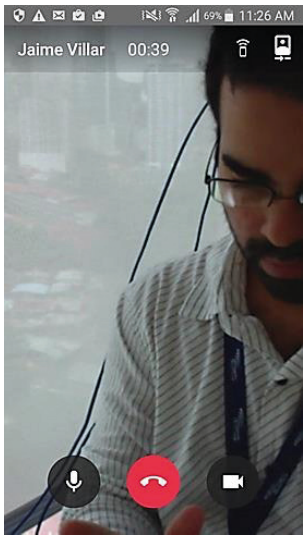

Figura 4. Vista de videollamada.

4.5.2 Diseño de la plataforma *web*

Se emplearon los mismos recursos de imágenes diseñados para la aplicación móvil, adecuándolo a un formato *web.*

Se trabajó en base al diseño predeterminado de *Q-municate.* Esta plantilla tiene por defecto el esquema de estilos de *Materialize,* el cual se modificaron partes de las hojas de estilo para ajustar el proyecto al esquema necesario por el sistema CXi.

Materialize es un esquema de trabajo CSS para el diseño y maquetación de páginas web creado por *Google.*

La pantalla de inicio de sesión cuenta con el formulario básico y registro en la plataforma de *QuickBlox,* así como los logos y colores del sistema CXi.

El diseño de la página principal del módulo de comunicación cuenta con un diseño básico.

Una barra superior contiene el logo y nombre del sistema en el lado izquierdo y al derecho un ícono con las opciones de la cuenta.

En el lado izquierdo de la pantalla se presenta el listado de contactos recientes y a su derecha si se selecciona alguno de los contactos se puede ver un contenedor que despliega un listado de los últimos registros de llamada y en su parte superior el botón que corresponde a la video-llamada. Véase figura 5.

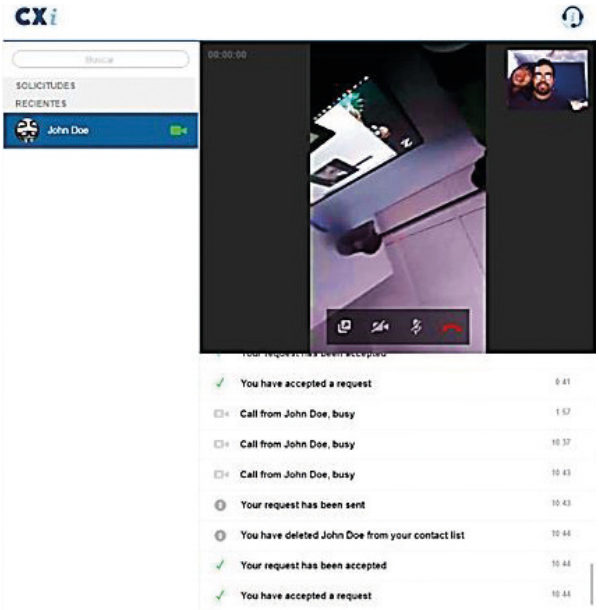

Figura 5. Vista principal del módulo de comunicaciones *web.*

Lo siguiente es la pantalla principal de los gráficos en tiempo real, la cual lee los datos almacenados en *QuickBlox* y los despliega en pantalla al momento empleando JavaScript, por medio de la librería de EON ofrecida por *PubNub.*

Cuenta con un gráfico lineal principal que muestra los valores de sonrisa en color azul, de interés en color naranja y la satisfacción estimada en color verde.

Posteriormente cuenta también con tres gráficos independientes para cada uno de estos valores, mostrando el porcentaje captado al momento.

En la figura 6 se muestra una captura de los gráficos descritos en donde se muestran los porcentajes y valores captados en una línea de tiempo.

Figura 6. Vista de la experiencia del cliente en tiempo real.

5. Implementación del sistema intérprete

Dentro del ambiente de desarrollo se emplearon distintos *software* y plataformas. Primeramente se tomó en cuenta el repositorio de los códigos y la herramienta de versionado. Para este propósito se utilizó *SourceTree* en sus versiones de Windows y Mac, así como una cuenta alojada en *BitBucket* para el fácil manejo de los códigos entre las distintas plataformas usadas.

El ambiente para el desarrollo móvil fue plenamente nativo usando el lenguaje de *Java* y el *software* oficial para el desarrollo en Android que es el Android Studio en su versión 2.1.x – 2.2.

En el caso de la plataforma web se empleó la estructura en *JavaScript* conocida como *Node.* js y otras herramientas de código abierto para el manejo de paquetes y ejecución de código mencionadas anteriormente.

Al integrar esta plataforma de servicios fue requerido crear una cuenta en *QuicBlox* para obtener unas credenciales y llaves del API que posteriormente serían incluidas en los aplicativos *web* y móvil.

5.1 Implementación de la aplicación móvil

Para incluir el módulo de comunicación dentro del proyecto en su variante móvil se clonó el repositorio de *Q-municate* desde *Git-hub,* se empezó a trabajar desde esa base y los módulos cargados por defecto. Se incluyeron las credenciales personalizadas al proyecto para poder validar las pruebas de conexión. Al integrar los métodos y librerías de *Mobile Vision* fue requerido agregar una de las dependencias de *Google Play Services* para acceder al API de *Vision.*

Tomando como base la arquitectura definida anteriormente se programó un modelo para el manejo de la data de la clase *Vision.*

En este modelo está compuesto de tres variables que son: (a) sesión, (b) sonrisa e (c) interés.

En esta clase también se realizó la conversión de los datos leídos del tipo *QBCustomObject* de la clase *Vision*. Ver figura 7.

```
public class Vision {

    public interface Contract {
        String SESION = "sesion";
        String SONRISA = "sonrisa";
        String INTERES = "interes";
    }

    private String id;
    private int userID;
    private int sesion;
    private int sonrisa;
    private int interes;
    private long date;
```

Figura 7. Fragmento del código de la clase *Vision.*

La clase *FaceGraphic* básicamente aloja el algoritmo 2 ya descrito en donde se declaran las variables que se usarán con el API de *Vision.* Ver figura 8.

```
float sonrisa = 0;
float ojoDerecho = 0;
float ojoIzquierdo = 0;
float rostroY = 0;
float rostroZ = 0;
public static int sonrisaTotal = 0;
public static int porcentajeInteres = 0;
public static int count = 1;
```

Figura 8. Variables empleadas en los métodos FaceGraphic.

Posteriormente se cargan los valores obtenidos del objeto *Vision.* Ver figura 9.

```
sonrisa = face.getIsSmilingProbability();
ojoDerecho = face.getIsRightEyeOpenProbability();
ojoIzquierdo = face.getIsLeftEyeOpenProbability();
rostroY = face.getEulerY();
rostroZ = face.getEulerZ();
```

Figura 9. Datos del objeto *Face.*

El porcentaje de simetría en ojos se calcula mediante la codificación de la figura 10.

```
int simetriaOjos;
Double sOjos;

if (ojoDerecho > ojoIzquierdo){
    sOjos = (ojoIzq / ojoDer) * 100.0;
} else if (ojoDerecho < ojoIzquierdo) {
    sOjos =  (ojoDer / ojoIzq) * 100.0;
} else {
    sOjos = 100.0;
}

simetriaOjos = sOjos.intValue();
```

Figura 10. Cálculo de simetría en ojos.

Una vez calculados estos valores, junto con el de mirada fija y finalmente el porcentaje de interés, se procede a crear el objeto *Vision* y cargar la sesión en *QuickBlox.*

Otro de los métodos elaborados para aplicar el tercer algoritmo fue el de *VisionData()*. En este método se implementa un hilo, que no es más que un proceso paralelo dentro del sistema Android, en donde cada segundo verifica si la conexión de video sigue activa, ejecuta el método *VisionObj()* y reinicia los valores de las variables usadas por el objeto visión.

5.2 Implementación de la plataforma *web*

Al igual que el proyecto móvil, se clonó inicialmente el repositorio del código de *Q-municate* para integrar las librerías y dependencias necesarias para la comunicación del sistema CXi.

Seguidamente se incluyeron las configuraciones iniciales dentro del archivo config.js, como los son las credenciales de la

cuenta en *QuickBlox* y el ID de la aplicación alojada en esta plataforma.

Adicionalmente se configuraron los URL de acceso al API. Para el desarrollo de este proyecto se implementó la plataforma *web* en un servidor local.

Para esto fue requerida la instalación de algunas librerías y dependencias como *Node*.js y *Ruby* por medio de la terminal.

La parte principal del módulo CXi es la página que muestra la experiencia del cliente en tiempo real. La estructura principal está basada en varios contenedores, el superior que muestra el gráfico lineal y tres inferiores que muestran los estados leídos de las variables de sonrisa e interés.

Para desplegar los gráficos se utilizaron ciertos métodos en *JavaScript,* los cuales se configuraban con los credenciales del API de *PubNub* y con los ajustes requeridos de acuerdo al gráfico que se necesitase. Ver figura 11.

La función encargada de ejecutar el algoritmo del árbol de decisión es la llamada *arbolCXi(),* esta recibe los parámetros de sonrisa e interés y estima un valor para la satisfacción.

Una vez ejecutado el algoritmo, inmediatamente se despliegan los valores en los respectivos gráficos.

5.3 Interpretación de los resultados

El sistema CXi pudo descifrar de manera óptima los niveles de sonrisa en los clientes y usuarios del aplicativo móvil, sin embargo, los estados emocionales del ser humano van más allá de las métricas captadas.

```
package com.quickblox.q_municate.model;

import ...

/**
 * Created by Jaime Villar on 08/17/16.
 */
public class Vision {

    public interface Contract {
        String SESION = "sesion";
        String SONRISA = "sonrisa";
        String INTERES = "interes";
    }

    private String id;
    private int userID;
    private int sesion;
    private int sonrisa;
    private int interes;
    private long date;


    public Vision(QBCustomObject qbCustomObject) {
        id = qbCustomObject.getCustomObjectId();
        userID = qbCustomObject.getUserId();
        sesion = Integer.parseInt(QBVisionObjectsUtils.parseField(Contract.SESION, qbCustomObject));
        sonrisa = Integer.parseInt(QBVisionObjectsUtils.parseField(Contract.SONRISA, qbCustomObject));
        interes = Integer.parseInt(QBVisionObjectsUtils.parseField(Contract.INTERES, qbCustomObject));
        date = qbCustomObject.getCreatedAt().getTime();
```

Figura 11. *Script* usado para el gráfico principal del módulo CXi.

Para este proyecto se tomaron en cuenta los niveles de sonrisa y de interés dando resultados positivos al momento de monitorear los niveles en tiempo real.

Esto le daba la posibilidad al sistema de conocer si el cliente está prestando la debida atención a la conversación y si mostró alguna sonrisa sin la necesidad de verlo físicamente.

En los cálculos de los niveles de satisfacción como ya se ha mencionado se sintió la carencia de datos para mejor la precisión de los datos resultantes, sin embargo, el alcance que tuvo el proyecto arrojó resultados que demuestran la posibilidad de que un sistema tenga la capacidad de detectar y mostrar en tiempo real los estados de ánimo.

Para los resultados obtenidos en el módulo de pruebas, en donde se hacía la comparativa entre el sistema tradicional de encuestas y el sistema CXi se descubrieron ciertas carencias que podrían afectar el análisis de los datos como:

- La disposición que tenga el usuario que se le toma la muestra, es decir, estas encuestas se llenan en ciertos casos sin el debido interés, solo para cumplir o salir rápido de eso.
- Siendo situaciones ficticias las evaluadas, es difícil que el usuario demuestre expresiones reales a ser captadas en el sistema.
- La poca cantidad de muestras, estudio y pruebas limitadas por el tiempo del proyecto. Esto imposibilita en cierta medida poder comparar entre sistemas.

6. Conclusiones

Las herramientas y plataformas de *software* como servicio tienen una gran variedad de ofertas y aplicaciones que se ajustan a un sinnúmero de modelos y negocios. Cuentan con la ventaja de brindar soluciones instantáneas, con buena fiabilidad, seguridad, flexibilidad y altos niveles de escalabilidad.

Sin embargo estas herramientas pueden significar una compleja integración dependiendo de los sistemas utilizados.

Las plataformas escalables le ofrecen a los sistemas la capacidad de crecer acorde a lo que se necesite en el momento, sin tener que

incurrir en gastos mayores para acondicionar infraestructuras que soporten futuras demandas.

Entre mayor sea el número de características tomadas de las expresiones faciales, mejor podrá alimentarse el sistema y evaluar estos datos para obtener resultados dentro de una escala aceptable de comparación con los valores reales.

Para poder realizar una evaluación, ajustes de las características del sistema y estudiar la eficacia del método propuesto es requerido un estudio estadístico que involucre a expertos en las múltiples áreas relacionadas.

Al tomar solo características del cliente se le niega al sistema la opción de evaluar las entradas producidas por los agentes de servicio.

Obtener valores que representen la satisfacción real de un individuo es difícil tanto para los métodos tradicionales como para los métodos más modernos.

Esto se debe a la complejidad que existe en las expresiones humanas, sin embargo, empleando algoritmos que emulen la naturaleza humana, es posible obtener resultados prometedores y muy cercanos a la realidad.

Recomendaciones

Buscar la ayuda y colaboración de expertos en la conducta humana, como psicólogos o científicos especializados. Esto para poder definir mejor los parámetros dentro del sistema.

Incluir el mayor número de características posibles dentro del sistema. Pueden ser expresiones faciales, tonalidades en la voz o cualquier otra característica cuantificable por medido de sensores presentes en los dispositivos empleados por los usuarios, clientes en este caso.

Evaluar en qué otros campos de aplicación tendría eficacia el sistema.

Tomar en cuenta todas las entradas y salidas posibles del sistema, como es el caso de los agentes de servicios. Si se conocen las entradas de los agentes y luego se comparan y evalúan con las entradas del cliente, el sistema puede llegar a proporcionar métricas y acciones que beneficiarían al servicio en general, como también podría aprender y poder predecir posibles fallas e indicadores de riesgo.

REFERENCIAS

[1] AFA. (2008). Automated Face Analysis. Septiembre 16, 2016, de AFA. Recuperado de http://www.cs.cmu.edu/~face/index2.htm

[2] Beard, R. (2014). 3 Ways to Use Data and Technology to Enhance the Customer Experience. Septiembre 9, 2016. Recuperado de http://blog.clientheartbeat.com/technology-enhancing- customer-experience/

[3] Brewster, S. (2016). Customer Service Bots Are Getting Better at Detecting Your Agitation. Septiembre 16, 2016, de MIT Technology Review. Recuperado de https://www.technologyreview.com/s/602352/customer-service- bots-are-getting-better- at-detecting- your-agitation/

[4] Campamá. G. (2005). 10 métodos para medir la satisfacción de los clientes. Septiembre 9, 2016, por EuQuality Networks. Recuperado de http://www.euquality.net/zonaprivada/descargas/Octubre%202005%20-%20Satisfaccion%20del%20Cliente.pdf

[5] Casalboni, A. (2016). Google Vision API: Image Analysis as a Service. Septiembre 14, 2016, por CloudAcademy Blog. Recuperado de http://cloudacademy.com/blog/google-vision- api-image- analysis/

[6] Castrillón, W. (2008). Técnicas de extracción de características en imágenes para el reconocimiento de expresiones faciales. Septiembre 12, 2016, de la Universidad de Pereira. Recuperado de https://www.researchgate.net/publication/44131123_TECNICAS_DE_EXTRACCION_DE_CARACTERISTICAS_EN_IMAGENES_PARA_EL_RECONOCIMIENTO_DE_EXPRESIONES_FACIALES

[7] Doerrfeld, B. (2015). 20+ Emotion Recognition APIs That Will Leave You Impressed, and Concerned. Septiembre 15, 2016, de Nordic Apis. Recuperado de http://nordicapis.com/20-emotion-recognition-apis-that-will-leave-you-impressed-and-concerned/

[8] Eclass. (2011). Tecnologías para medir la satisfacción del cliente. Septiembre 8, 2016, de eclass. Recuperado de https://comunidad.eclass.com/articulo/16423/tecnologias-para-medir-la-satisfaccion-del-cliente

[9] Ekman, P., & Friesen, W. (1978). FACS – Facial Action Coding System. Recuperado de http://www.cs.cmu.edu/afs/cs/project/face/www/facs.htm

[10] Ekman, P., & Oster, H. (1979). Facial expressions of emotion. Annual Reviews in Psychology. 30, 527-554.

[11] Ekman, P. (2001). Telling Lies. Berkley Books, Nueva York.

[12] Gartner. (2014). Gartner Surveys Confirm Customer Experience Is the New Battlefield. Septiembre 6, 2016, de Gartner for Marketers. Recuperado de http://blogs.gartner.com/jake-sorofman/gartner-surveys-confirm-customer-experience-new-battlefield/

[13] Intersect. (2016). Emotion Detection and Artificial Intelligence. Septiembre 16, 2016, de Intersect. Recuperado de http://www.intersect.org.au/cases/emotion-detection-and-artificial-intelligence

[14] ISO. (2013). Satisfacción del cliente, Norma ISO 9001:2008. Septiembre 6, 2016, de ISO.org. Recuperado de http://www.iso.org/iso/home/store/catalogue_ics/catalogue_detail_ics.htm?ics1=03&ics2=120&ics3=10&csnumber=63027

[15] Rourke, J. (2016). How Machine Learning Will Improve Retail and Customer Service. Septiembre 9, 2016, de Data Informed. Recuperado de http://data-informed.com/how-machine-learning-will-improve-retail-and-customer-service/

[16] Yegulalp, S. (2014). 11 open source tools to make the most of machine learning. Septiembre 13, 2016, de InfoWorld. Recuperado de http://www.infoworld.com/article/2853707/machine-learning/11-open-source-tools-machine-learning.html