

TRABAJO DE GRADO MODALIDAD PASANTÍA, PARA OBTENER EL TÍTULO  
DE INGENIERO ELECTRÓNICO

APLICACIÓN DE LA MINERÍA DE DATOS EN LA DETECCIÓN AUTOMÁTICA DE FRASES  
DE FALLOS Y SOLICITUDES DE INGRESO AL TALLER PARA EL DEPARTAMENTO DE  
MANTENIMIENTO DE MASSEQ.



PRESENTADO POR:  
LUIS FELIPE TAMAYO CUMBRE  
Cod- 9702

CORPORACIÓN UNIVERSITARIA AUTÓNOMA DEL CAUCA  
FACULTAD DE INGENIERÍA  
PROGRAMA DE INGENIERÍA ELECTRÓNICA  
POPAYÁN- CAUCA  
2022

TRABAJO DE GRADO MODALIDAD PASANTÍA, PARA OBTENER EL TÍTULO  
DE INGENIERO ELECTRÓNICO

APLICACIÓN DE LA MINERÍA DE DATOS EN LA DETECCIÓN AUTOMÁTICA DE FRASES  
DE FALLOS Y SOLICITUDES DE INGRESO AL TALLER PARA EL DEPARTAMENTO DE  
MANTENIMIENTO DE MASSEQ.



PRESENTADO POR:  
LUIS FELIPE TAMAYO CUMBRE  
Cod- 9702

DIRECTOR:  
MGR. YESID ENRIQUE CASTRO CAICEDO

CORPORACIÓN UNIVERSITARIA AUTÓNOMA DEL CAUCA  
FACULTAD DE INGENIERÍA  
PROGRAMA DE INGENIERÍA ELECTRÓNICA  
POPAYÁN- CAUCA  
2022

## NOTA DE ACEPTACIÓN

Aprobado por el comité de grado en cumplimiento de los requisitos exigidos por la Corporación Universitaria Autónoma del Cauca para optar al título del Ingeniero Electrónico.



---

**MGR. YESID ENRIQUE CASTRO CAICEDO**

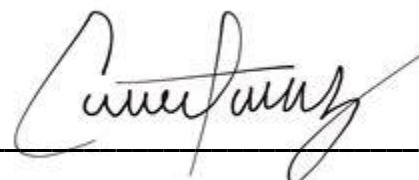
**DIRECTOR**



---

**MGR. GLORIA LILIANA LOPEZ MUÑOZ**

**JURADO**



---

**MGR. CARLOS FELIPE LOPEZ CORDOBA**

**JURADO**

## Resumen

Los avances en las técnicas de Deep Learning han exhibido una serie de ventajas competitivas que ha llamado la atención de empresas y organizaciones de todo el mundo en distintos sectores económicos, en búsqueda de ello se realiza el convenio académico-empresarial con el objetivo de tratar uno de los problemas que requiere solución en esta área del conocimiento, se trata de la implementación de un sistema capaz de identificar mensajes donde se expone información de fallos en equipos y maquinaria, temática que concierne al procesamiento de lenguaje natural (PLN), uno de los campos de las ciencias de la computación. Para ello se consideraron distintas tecnologías donde son aplicadas técnicas de tratamiento de lenguaje, prevaleciendo el diseño de un algoritmo desarrollado en Python que a su vez emplea la arquitectura *Transformers* (Vaswani y otros, 2017), modelo que ha demostrado gran efectividad en el tratamiento de lenguaje humano.

En este documento se presentan bases teóricas y las fases de desarrollo cuya implementación requiere extraer las oraciones referentes al área de mantenimiento y notificarlas por un medio de transmisión dedicado, un grupo alojado en el servicio de mensajería WhatsApp.

**Palabras clave:** procesamiento de lenguaje natural, redes neuronales, arquitectura transformers, inteligencia artificial.

## **Abstract**

The advances in the techniques of Deep Learning have exhibited a series of competitive benefits that have drawn the attention of companies and organizations from all over the world in different economic business. In search of it, the academic-business agreement is realized with the aim of dealing with one of the problems that requires solution in this area of knowledge. The implementation of a system capable of identifying messages where information on failures in equipment and machinery is exposed, a subject that concerns natural language processing (NLP), one of the fields of computer science. For this, different technologies were considered where language processing techniques are applied, prevailing the design of an algorithm developed in Python that uses the Transformers architecture (Vaswani et al., 2017), a model that has shown great effectiveness in the treatment of human language.

This document presents theoretical bases and the development phases whose implementation requires extracting the sentences referring to the maintenance area and notifying them through a dedicated transmission medium, a group hosted on the WhatsApp messaging service.

**Keywords:** natural language processing, neural networks, transformers architecture, artificial intelligence.

# Contenido

<b>1. INTRODUCCIÓN</b>	<b>1</b>
<b>2. MARCO CONTEXTUAL</b>	<b>3</b>
<b>3. OBJETIVOS</b>	<b>6</b>
4.1. OBJETIVO GENERAL	6
4.2. OBJETIVOS ESPECÍFICOS	6
<b>4. MARCO REFERENCIAL</b>	<b>7</b>
5.1. APRENDIZAJE AUTOMÁTICO – MACHINE LEARNING	7
5.1.1 <i>Aprendizaje supervisado</i>	7
5.1.2 <i>Aprendizaje no supervisado</i>	8
5.1.3 <i>Aprendizaje por refuerzo</i>	8
5.1.4 <i>Aprendizaje profundo – Deep Learning</i>	9
5.1.5 <i>Neurona Artificial – Perceptrón</i>	9
5.2. REDES NEURONALES RECURRENTES	13
5.3. PROCESAMIENTO DE LENGUAJE NATURAL	13
5.3.1 <i>En base a reglas</i>	14
5.3.2 <i>En base a datos</i>	14
5.4. REDES TRANSFORMERS	15
5.4.1 <i>Arquitectura del modelo</i>	18
5.4.2 <i>Tokenización</i>	19
5.4.3 <i>Word Embedding</i>	21
5.4.4 <i>Codificación posicional</i>	26
5.4.5 <i>Bloques codificador y decodificador</i>	29
<b>6. METODOLOGÍA</b>	<b>34</b>
6.1. RECOLECCIÓN DE INFORMACIÓN DEL OBJETO DE ESTUDIO	35
6.2. DEFINICIÓN DE LOS PARÁMETROS DE CATEGORIZACIÓN DE PRIORIDAD EN EL TRATAMIENTO DE LOS REPORTES DE CONTROL DIARIO	37
6.3. ANÁLISIS Y ELECCIÓN DE TECNOLOGÍAS DE EXTRACCIÓN DE INFORMACIÓN	40
6.3.1 <i>Aplicaciones de procesamiento de lenguaje natural basadas en la nube</i>	44
6.3.1.1. Amazon Comprehend	44
6.3.1.2. Microsoft Azure	46
6.3.1.3. Watson Natural Language Understanding	47
6.3.1.4. Lenguaje estructurado de programación Python	49
6.3.2 <i>Modelos y tareas de los Transformers</i>	51
6.3.2.1. Text generation	53
6.3.2.2. Text Clasification	54
6.4. ETAPAS DE DESARROLLO DEL APLICATIVO	55
6.4.1 <i>Preparación de los datos</i>	55
6.4.2 <i>Pruebas de Tokenización y Aprendizaje por Transferencia</i>	65
6.4.3 <i>Preparación del entorno de entrenamiento</i>	68
6.4.4 <i>Establecimiento de hiperparámetros de entrenamiento.</i>	70
6.4.4.1. Los tamaños de los lotes	71
6.4.4.2. Épocas de entrenamiento	71

6.4.4.3. Longitud máxima de las oraciones	71
6.4.4.4. Métrica de evaluación	72
6.4.5. <i>Reajuste de los datos de procesamiento</i>	74
6.4.6. <i>Entrenamiento</i>	75
6.4.7. <i>Evaluación del modelo</i>	80
6.5. DESPLIEGUE	82
6.5.1. <i>Integración a la plataforma Maseq</i>	82
6.5.2. <i>Notificación de las alertas de fallos al equipo de mantenimiento</i>	83
6.5.3. <i>Ejecución programada</i>	84
<b>7. RESULTADOS</b>	<b>86</b>
<b>8. CONCLUSIONES</b>	<b>93</b>
<b>9. TRABAJOS FUTUROS</b>	<b>95</b>
<b>10. REFERENCIAS</b>	<b>96</b>

## Contenido de Tablas y Gráficos

Fig. 1 Diagrama de una neurona artificial	10
Fig. 2 Función Sigmoide: relaciona valores negativos grandes a 0, y valores positivos grandes a 1	11
Fig. 3 Función ReLU: Permite el paso de valores positivos, mientras asigna 0 a todo valor negativo.	11
Fig. 4 Red neuronal artificial	12
Fig. 5 Análisis secuencial de lenguaje en RNR	16
Fig. 6 Análisis paralelo en redes transformes	17
Fig. 7 Los Transformer.	18
Fig. 8 Extracción de texto a tokenizar	19
Fig. 9 Representación gráfica Tensorboard, internet como núcleo relacional	25
Fig. 10 Sumatoria del codificador posicional e incrustación de palabra.	27
Fig. 11 Codificación Posicional de 5 elementos $L(i) \in [0,4]$ , dimensión del Word embedding $d_{\{model\}} \in [0,7]$	29
Fig. 12 Bloques Codificador-decodificador	30
Fig. 13 Bloque codificador	30
Fig. 14 Múltiples cabezas de atención	31
Fig. 15 Pila de bloque Codificador	33
Fig. 16 Pila Decodificadora	34
Fig. 17 Etapas de desarrollo del proyecto	35
Fig. 18 Función del procesamiento de lenguaje	41
Fig. 19 Disponibilidad de aplicaciones , (imagen propia).	42
Fig. 20 RESUMEN DE TAREAS DISPONIBLES EN HUGGING FACE	52
Fig. 21 Text generation huggingface	53
Fig. 22 Text Clasification huggingface	54
Fig. 23 Informe generado por plataforma administrativa Maseq	57
Fig. 24 Importación de .xlsx a dataframe	58
Fig. 25 Reducción de dimensiones dataframe	59
Fig. 26 Patrones de mensajes encontrados en control diario.	61
Fig. 27 Diagrama resumen del preprocesamiento de datos de entrenamiento.	62
Fig. 28 Distribución de los mensajes de control diario	65
Fig. 29 Importación del modelo pre-entrenado	66
Fig. 30 Definición del número de etiquetas.	69
Fig. 31 Balanceo de datos de alerta e informativos.	75
Fig. 32 Establecimiento de hiperparámetros en el algoritmo	76
Fig. 33 Distribución segmentos de entrenamiento	77
Fig. 34 Ventana de visualización de la salida del modelo	79
Fig. 35 Múltiples cabezas de atención.	80
Fig. 36 Extracto de código de la predicción del modelo.	81
Fig. 37 Envío de notificaciones vía WhatsApp Web.	84
Fig. 38 programador de tareas en virtualización de Windows.	85
Fig. 39 Mensajes notificados por el Script en el grupo de WhatsApp.	92



## 1. Introducción

La explosión en la generación de contenidos en línea impulsada principalmente por la masificación del internet ha traído consigo varios retos para las ciencias de la computación, se trata de la deficiente manejabilidad limitada por los sistemas administrativos de información convencionales, especialmente cuando se trata de extensos volúmenes de información, esta problemática tiene respuesta en un área del conocimiento, la inteligencia artificial, un término que se hace cada vez más popular dentro del mundo informático, liderada por gigantes tecnológicos como Google, Facebook, OpenAI, Microsoft, Oracle, entre otros tantos, donde se aplican soluciones tan diversas como lo son el tratamiento imágenes para el reconocimiento de objetos, personas, los convertidores y moduladores de voz, como los que usan los asistentes virtuales de los teléfonos inteligentes, pasando por la administración especializada de los datos como el *big data*, la minería de datos, y el procesamiento de lenguaje natural.

No hace falta hacer una búsqueda muy profunda para encontrar información de las millonarias inversiones que se realizan en estas tecnologías. Tortoise Intelligence por ejemplo, reporta en un reciente informe que empresas a nivel mundial han hecho millonarias inversiones en los campos de la inteligencia artificial, donde se alcanzó la suma de 77500 millones de dólares para el 2021, batiendo un nuevo récord y superando el del año anterior, que llegó a los 36000 millones de dólares (tortoisemedia, 2021).

Este es un fenómeno generalizado que se ha transformado en más que un objetivo, en una necesidad. Empresas de sectores económicos tan distantes como lo

puede ser una compañía del sector de las obras civiles y transporte quien se ve desbordada por la cantidad de información que genera cada día, entra en la búsqueda de soluciones que se basen en modelos de inteligencia artificial. Para el caso particular, Maseq proyectos e ingeniería requiere detectar los mensajes que reportan fallos en sus activos, almacenados en un sistema de control de equipos y maquinaria que es generado diariamente, estos sirven para alertar internamente a todo un equipo quien realiza mantenimientos y reparaciones en equipos de distintas categorías mecánicas.

En base a la literatura de las ramas de aplicación de la inteligencia artificial, el método más efectivo para el reconocimiento de ese tipo de expresiones se encuentra en el procesamiento de lenguaje natural (PLN), hallando suficiente documentación de uso y sus las aplicaciones, como la que se nos enseña en el libro ***Introduction to natural language processing*** por Jacob Eisenstein (EISENSTEIN, 2019), quien nos explica que el procesamiento de lenguaje natura es una combinación de computación lingüística y aprendizaje automático (***Machine Learning***) donde se es capaz de aproximar el lenguaje humano a una lógica que utilizan las computadoras.

Este documento aspira guiar por las etapas y rutas que se tomaron para lograr el reconocimiento de lenguaje y clasificarlo bajo ciertos parámetros, un reto que tiene varias particularidades, fundamentalmente por el tipo de lenguaje que se utiliza en los reportes de control diario, quien puede verse cargado de jerga local con matices que tienden a ser específicas a la ubicación geográfica en la que se encuentra, se trata del centro y sur de Colombia. En él, se propuso adaptar un modelo que cumpla con los retos que presenta un caso tan específico. El propósito se satisface cuando estos mensajes sean notificados

directamente al personal encargado de manera automática, utilizando un medio de transmisión exclusiva, a través de la plataforma de mensajería WhatsApp.

## **2. Marco contextual**

La modalidad de pasantía es una ventana de apertura al mundo laboral donde son aplicados conocimientos adquiridos durante la etapa académica de los estudiantes universitarios, con motivo de ello se hizo la solicitud de un convenio académico donde se ven involucradas la Corporación Universitaria Autónoma del Cauca, la compañía Maseq proyectos e ingeniería, y el propio estudiante. Para la realización de este convenio fue necesario el planteamiento de una solución a un problema donde se viera involucrado el desarrollo de destrezas típicas de la ingeniería electrónica y sus especialidades, se trata de la codificación de un sistema basado en inteligencia artificial que con la capacidad de reconocer el lenguaje humano para la extracción de información puntual.

Para entender el contexto de lo que exige la compañía es necesario describir el entorno en el que se está ubicado. Maseq es una compañía con base en el Huila, quien está involucrada en distintos tipos de negocio principalmente en el desarrollo de obras civiles, además de ello entre sus oficios también destaca la extracción directa de materiales para la construcción, transformación, y el servicio complementario del transporte del material, destacando siempre por las buenas prácticas en el ejercicio.

Para este tipo de operación es requerido una gran variedad de equipos de distintas categorías mecánicas y eléctricas, por mencionar algunos, se usan desde equipos para la

construcción como reglas vibratorias, cortadoras de concreto, equipos especializados de topografía, unidades para el desplazamiento de personal como buses, automóviles, camionetas, pasando por equipos de transporte como tracto camiones, volquetas, y finalizando en maquinaria pesada. Esta multitud de equipamiento requiere de un control muy exhaustivo para permitir la operación sin contratiempos, en sus actividades hay una serie de normativas internas que establecen cómo deben realizarse los procesos de mantenimiento y reparación de estos. De la intervención se ocupa un área dedicada a ello, se trata del departamento de mantenimiento, sin embargo, este precisa de la notificación del estado funcional de todos los equipos, por ejemplo, cuando son requeridos ciertos servicios que pueden estar por fuera de las previsiones, fallos y daños en el equipamiento, desgaste por uso, etc. Por tal motivo se estableció un servicio de control diario donde todo equipo, maquinaria y activo operativo en la compañía tiene la obligación de hacerlo.

El control diario es un formato físico en el que se registran distintos datos relevantes al equipo: la identificación del elemento en cuestión, la fecha, tiempo de horas trabajadas, para el caso de equipos móviles el kilometraje, y horómetro para maquinaria pesada, del mismo modo de unos espacios denominados observaciones y novedades, en estos últimos los operadores y conductores registran información que puede, o no ser dirigida para el área de mantenimiento. Por ejemplo, puede mencionar las actividades que el equipo realizó durante la jornada laboral, las rutas de desplazamiento que usó, asimismo como reportar alteraciones en el funcionamiento habitual.

La información que es reportada en control diario es recolectada por un área dedicada a ello, aquí inicia la intervención del área de minería de datos, quien consta

principalmente de dos partes. Una que está centrada en la recolección de la información a través de la recepción de ciertos formatos, que posteriormente son digitados en una plataforma web, donde administran algunos de los servicios de la compañía, este personal está ubicado en los distintos proyectos y obras; la otra parte está encargada de recibir esta información, examinarla, y retransmitirla a las áreas pertinentes. Este fue un sistema muy efectivo en sus orígenes, puesto que el volumen de los equipos era mucho menor, además de una distribución geográfica reducida. Como es natural la operación y el número de proyectos fue evolucionando y expandiéndose, por lo que cada vez se requería más personal que se dedicara a examinar de manera manual el control diario, convirtiéndose en un ciclo insostenible, con un trabajo muy mecánico y a veces ineficiente, dado que se pasaban por alto, por ejemplo, mensajes donde se requería intervención para mantenimiento y nunca se notificaban.

En base a esta problemática las áreas administrativas decidieron buscar alternativas para la extracción de estas frases de una manera automatizada, es allí donde se vio la oportunidad de mejora y se accedió al convenio entre las tres partes mencionadas al principio.

Como requerimientos funcionales para el trabajo a ser realizado se solicitó la creación de un sistema de detección automático de frases de fallos y solicitudes de mantenimiento, extracción de la información directamente de las bases de datos y análisis de los mensajes relacionados para su respectiva notificación.

### **3. Objetivos**

#### **4.1. Objetivo General**

Diseñar una herramienta computacional de extracción de frases de alerta dentro de texto desestructurado, que notifique el estado funcional de los equipos para el departamento de mantenimiento de Maseq SAS.

#### **4.2. Objetivos específicos**

- Reconocer expresiones de los fallos en la maquinaria desde el compilado de reporte de control diario.
- Parametrizar por relevancia las solicitudes de mantenimiento de los equipos.
- Identificar patrones en la sintaxis de los reportes de fallos en el equipamiento.
- Automatizar la extracción de frases de competencia del área de taller electromecánico.
- Enlazar las bases de datos del sistema Maseq al algoritmo de interpretación de frases de alerta.
- Generar informes en tiempo real de los equipos con requisición de mantenimiento.

## 4. Marco Referencial

### 5.1. Aprendizaje automático – Machine Learning

El aprendizaje automático es uno de los enfoques de estudio del campo de la IA (Inteligencia artificial) en el que los sistemas informáticos cuentan con la capacidad de: (a) Aprender a partir de grandes volúmenes de información, (b) Encontrar patrones y (c) Generar estructuras para atender problemas de distinta índole, entre los cuales se encuentra la automatización de procesos que superen los retos típicos que traen consigo la generación de los macrodatos (**Big Data**). El *Big data* es una técnica de administración de grandes volúmenes de datos que no puede ser procesada fácilmente por el software convencional, en cambio, se logra extraer información que permite resolver problemas que antes no podrían siquiera abordarse. (ORACLE, 2022). Se dota los sistemas de algoritmos computacionales con principalmente dos habilidades que son propias de organismos inteligentes, la adquisición de nuevo conocimiento, y la autonomía en la toma de decisiones entre dos o más alternativas para la solución de problemas. Basados en el tipo y el volumen de los datos, el aprendizaje de maquina (**Machine Learning**) ofrece diferentes técnicas para alcanzar los objetivos dependiendo del enfoque de estudio. (ibm, 2021) El aprendizaje maquina se divide principalmente en 3 categorías, el aprendizaje supervisado, el no supervisado, además, del aprendizaje por refuerzo.

#### 5.1.1 Aprendizaje supervisado

Se refiere a un tipo de aprendizaje fundamentado en descubrir las relaciones existentes entre unas variables de entrada y unas variables de salida, se asocia un sistema de etiquetas que pueda inferir las características de mayor importancia para que un

modelo haga análisis predictivo y sea capaz de tomar decisiones de manera autónoma, es decir toma ciertos elementos representativos de muchos ejemplos, los compara extrayendo los patrones que le representen, y si luego se dan las condiciones el algoritmo será capaz de clasificar el resultado al conjunto más aproximado, incluso si se le suministra valores que no haya considerado antes.

### **5.1.2. Aprendizaje no supervisado**

Esta técnica de aprendizaje se usa cuando existe una gran cantidad de información que no tiene clasificación aparente ni contexto de los datos de entrada, logra la agrupación de datos en clústeres a pesar de la falta de conocimiento previo. A diferencia del aprendizaje supervisado, aquí no se cuenta con un sistema de etiquetas que indiquen las características más relevantes, en cambio este se enfrenta a grandes volúmenes de información de forma iterativa con el objetivo de tener una representación de las variables que le permitan cualificar el objeto de estudio en uno u otro ámbito. Ejemplo de ello son los clasificadores de spam que poseen los gestores de correos de Gmail, Outlook, etc. (Gbenga Dada y otros, 2019). Considerando la gran cantidad de correo que se envía diariamente con cientos de varianzas en la forma, contenido, redacción u origen, estos logran asociarlos eficazmente como correo no deseado a su categoría correspondiente; esto suele hacerse de manera no supervisada, puesto que encontrar los patrones con tal variedad información es humanamente imposible.

### **5.1.3. Aprendizaje por refuerzo**

La manera en la que se estructura este tipo de algoritmos es a partir de la experiencia, es un método heurístico en el cual el conocimiento se obtiene a partir del



ensayo prueba y error, consiste en probar una alternativa y verificar su funcionamiento, si el resultado es de utilidad esta variante se elige de tal manera que se recompensen las decisiones correctas, fortaleciendo así el proceso de toma de decisiones, todo ello de forma cíclica hasta que se satisfagan los requisitos planteados por el problema original.

#### **5.1.4. Aprendizaje profundo – Deep Learning**

El aprendizaje profundo es el resultado de años de investigación en las ciencias de la computación en la que se busca simular las capacidades y el funcionamiento del cerebro humano desde su componente elemental: la neurona, hasta las complejas relaciones que estas establecen, prestando especial atención en cómo interacciones de este tipo provocan un razonamiento lógico.

El primer modelo de red neuronal propuesto por McCulloch y Pitts (s.), tenía el principio de la frase “todo o nada” cuya salida resultaba en un modelo binario condicionado por un umbral escalón con valores prefijados, fue el origen de lo que hoy se conoce como Neurona Artificial.

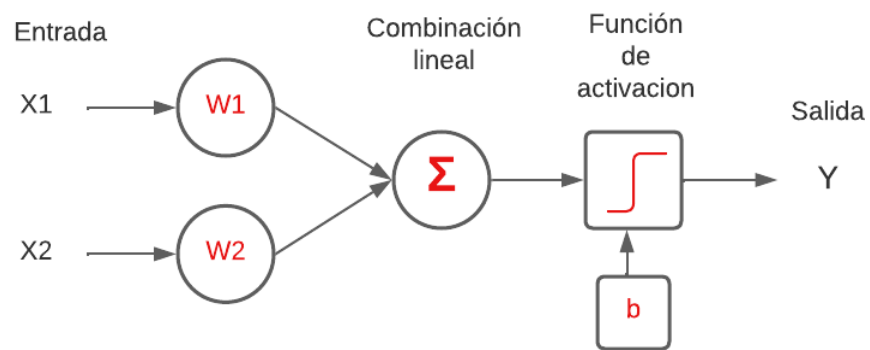
#### **5.1.5. Neurona Artificial – Perceptrón**

Una neurona artificial está formada por una serie de operaciones con dos o más entradas  $x_i$  cada una de ellas asociada a unos pesos  $w_i$  que operan entre si como el producto de las entradas por los pesos correspondientes, resultando en una combinación lineal a la que más tarde que se le adiciona un sesgo (bias [b]) que predispone a la neurona a activarse. (Lin, 2017)

*Diagrama de una neurona artificial*

Fig. 1

*Diagrama de una neurona artificial*



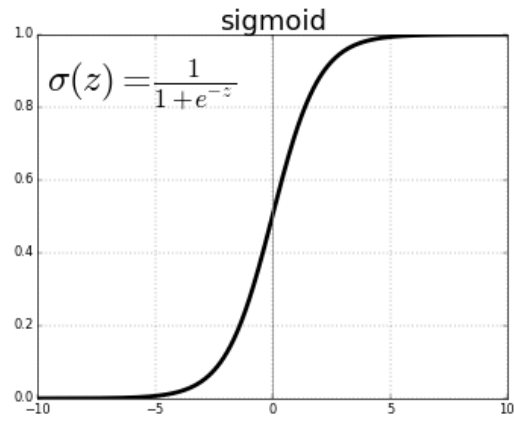
La salida de la neurona exige una función de activación  $y = f(n)$  que se encargue de normalizar la combinación lineal anteriormente descrita.

$$n = b + \sum_i w_i x_i \quad (1)$$

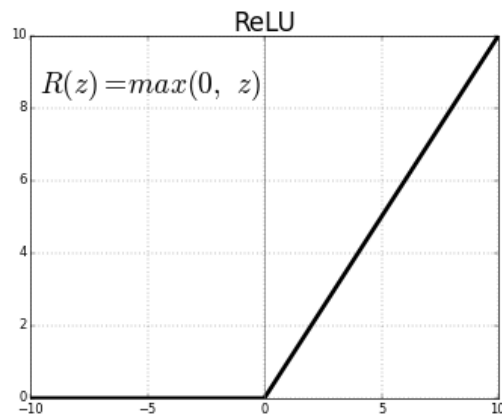
Para la normalización de una función de activación se utilizan distintos métodos, generalmente funciones como la sigmoide o la ReLU (Unidad Lineal rectificada).

**Fig. 2**

***Función Sigmoide: relaciona valores negativos grandes a 0, y valores positivos grandes a 1***

**Fig. 3**

***Función ReLU: Permite el paso de valores positivos, mientras asigna 0 a todo valor negativo.***



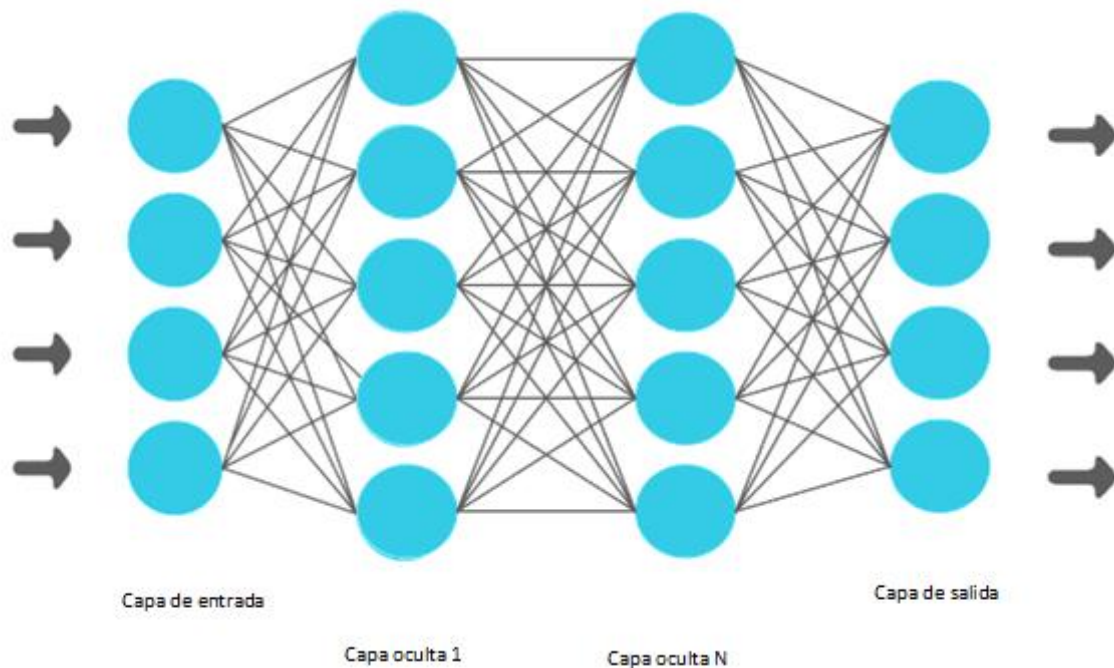
De manera similar a como lo hace el cerebro humano una red neuronal artificial se compone de cientos de neuronas interconectadas en distintas capas; la habilidad de

autoaprendizaje reside en el gran número de cómputo que ofrece la distribución del procesamiento en distintas neuronas, es decir juntar una cantidad de neuronas que por sí sola no tiene mayor capacidad de solución de problemas, pero que combinadas ofrecen un potente poder de cómputo.

### 5.1.6 Redes neuronales Multicapa

**Fig. 4**

**Red neuronal artificial**



Existen 3 tipos de capas: de entrada, las ocultas y de salida. La razón por la que se da el nombre de Deep Learning se debe a que dentro de las capas ocultas pueden darse un sinnúmero de capas profundas que conjuntamente contribuyen al procesamiento

especializado, de esta manera se puede superar retos de distinta índole con gran efectividad, mejorando procesos tales como reconocimiento de voz, visión maquina o procesamiento de lenguaje natural.

## **5.2. Redes neuronales recurrentes**

Una de las características que hicieron la redes neuronales recurrentes populares es la capacidad de memoria con la que fueron concebidas, estas son capaces de almacenar estados de lectura de capas anteriores a la suya, diferente a como lo hace una red convolucional, el tipo de red más básica que existe. Esta última consta de una función de activación que actúa exclusivamente en una dirección, mientras que la RNR es capaz de realimentarse a sí misma. En una red neuronal con varias capas, este comportamiento es replicado, ejecuta la función de activación, realimenta su salida, y si antes suyo se ejecutaron otras funciones de activación, estas también se usan para reajustar la salida.

## **5.3. Procesamiento de Lenguaje Natural**

Los conjuntos de texto y el lenguaje en general no es una tarea que un sistema informático pueda comprender en sí mismo, el lenguaje humano presenta muchos matices que pueden variar por motivos culturales, sociales e idiomáticos. La capacidad de análisis de texto en informática está facultada por un proceso de modelización matemática en el que las palabras son convertidas a representaciones numéricas combinando modelos estadísticos, reglas lingüísticas o de aprendizaje automático, de esta manera se faculta el procesamiento del lenguaje.

Existen muchos tipos de algoritmos para el procesamiento de lenguaje natural, no obstante, dos de ellos son los más usados:

### 5.3.1. En base a reglas

Dependiente de la noción de expertos lingüistas, se basa en una serie de reglas idiomáticas donde el lenguaje es dividido según las categorías gramaticales, el desarrollo de un algoritmo en base a reglas requiere de la inversión de mucho tiempo puesto que debe crearse una especie de diccionarios manualmente donde cada palabra debe tener una asignación gramatical, a la vez que la relación entre las distintas categorías lingüísticas, si se trata de verbos, sustantivos, adjetivos, etc. a esto se le denomina etiquetado gramatical (*Part Of Speech Tagging*).

**Tabla 1. Etiquetado gramatical**

<b>La</b>	<b>escuela</b>	<b>se</b>	<b>inauguró</b>	<b>el</b>	<b>año</b>	<b>pasado</b>
<b>Artículo</b>	<b>sustantivo</b>	<b>pronombre</b>	<b>verbo</b>	<b>artículo</b>	<b>sustantivo</b>	<b>adjetivo</b>

La mayor limitación se da en el tamaño del diccionario puesto que cada una de las palabras debe tener su categoría gramatical asignada, de lo contrario los algoritmos basados en *POS tagging* pueden disminuir su efectividad. La desventaja más evidente es su limitación debido a que la combinación entre palabras debe estar previamente definida y la composición de una oración es válida únicamente si se cumple con las reglas que han sido establecidas originalmente.

### 5.3.2. En base a datos

Este tipo de algoritmos requiere de la colección de grandes corpus de información que sirven como datos de muestra para la aplicación de modelos estadísticos y de aprendizaje automático, la finalidad es que las redes neuronales se usen para encontrar

patrones que describan las características que hace una oración elocuente, el detalle es interpretar las relaciones existentes entre palabras desde una perspectiva matemática evitando el uso de parámetros que tienen más sentido para el razonamiento humano como lo es la división gramatical, por lo que se acude enteramente a valores y operaciones numéricas, es aquí donde la arquitectura Transformers cimienta sus bases.

#### 5.4. Redes Transformers

Los Transformers son la evolución más reciente del campo del Deep Learning, se trata de una arquitectura novedosa introducida por Google a finales del 2017 con un artículo titulado “Attention is all you need” (*Vaswani y otros, 2017*). La propuesta llegó con el objetivo de superar las falencias que afectan los rendimientos de las redes neuronales recurrentes en cuanto a tiempos de entrenamiento, limitaciones resultantes de la naturaleza secuencial aplicada en arquitecturas anteriores incluyendo las **Long short-term memory** (LSTM) de 1 o más dimensiones (*Gernot A. Fink, 2022*).

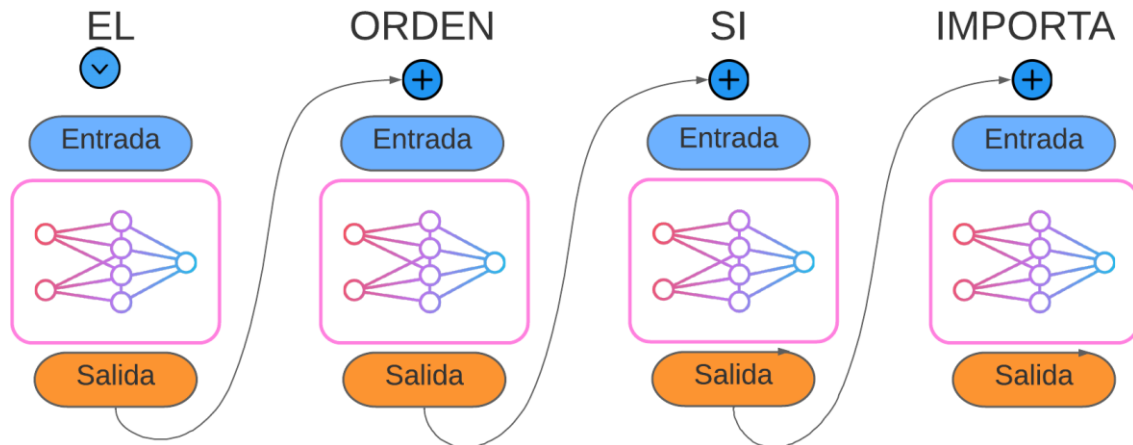
El lenguaje humano cuenta con una serie de reglas que dictan la sintaxis de las oraciones, en donde el orden de las palabras es importante para dar contexto de lo que quiere expresarse, de esta manera han sido diseñados los primeros algoritmos para el procesamiento de lenguaje natural, en ellos se parte de la lógica humana donde la lectura e interpretación de las oraciones y texto en general se hace de manera progresiva, analizando palabra por palabra. A grandes rasgos, el análisis de segmentos de texto en este tipo de arquitecturas suele tomar forma de bucle finito donde las iteraciones son la cantidad de palabras en la oración, y cada iteración tiene de entrada tanto las palabras

que componen la frase, como los valores resultantes del análisis anterior (esto último cuando se analiza de la segunda palabra en adelante). Resultando un análisis encadenado que ralentiza el entrenamiento de los modelos.

A modo de ejemplo se tiene de entrada la frase EL ORDEN SI IMPORTA, en redes neuronales recurrentes se haría de manera progresiva como lo muestra la figura 5.

**Fig. 5**

***Análisis secuencial de lenguaje en RNR***



Entonces, el razonamiento que usan anteriores arquitecturas a las Transformers es tomar cada palabra, cuando se analice la primera de ellas, se pasaría a la segunda no solo recibiendo como entrada la palabra misma, sino también el análisis e información de la palabra anterior, de esta manera hasta que se complete la secuencia de la oración.

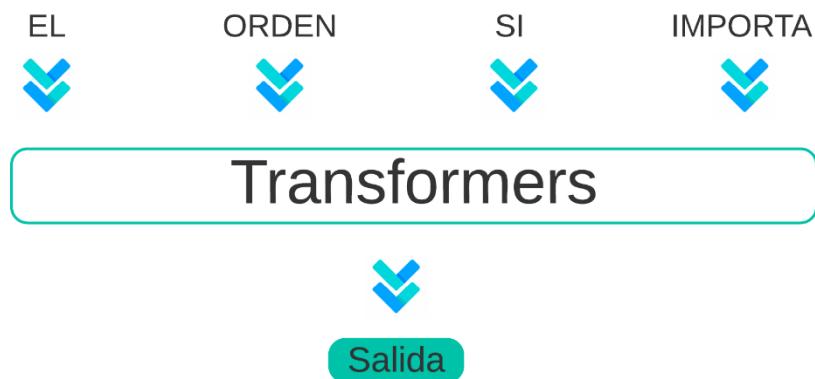
Pese a que los Transformers tiene en cuenta la importancia del orden en las oraciones, contrario a como lo hacen otras implementaciones, esta es capaz de procesar toda la información al tiempo (figura 6), es decir, analizar el texto paralelamente,



posibilitando mayor rendimiento, principalmente cuando se trata de entrenar cuerpos que pueden contener millones de palabras en información.

**Fig. 6**

***Análisis paralelo en redes transformes***

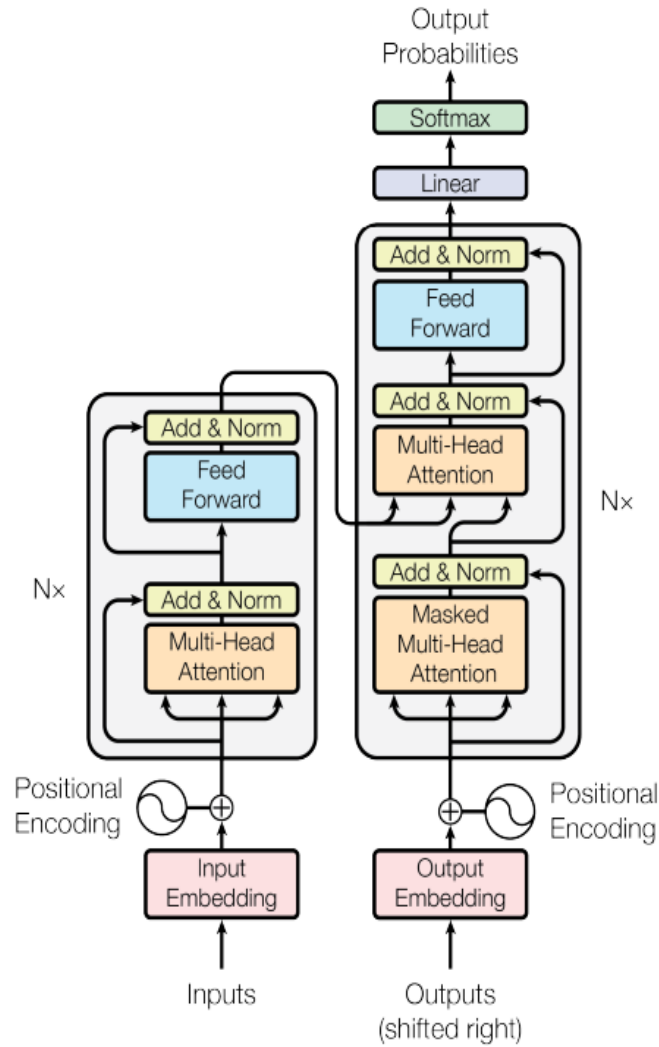


No se puede pasar por alto que aquí son tomadas ciertas técnicas ya usadas en las redes neuronales recurrentes, sin embargo, la mayor efectividad se consigue cuando se introduce el concepto de “autoatención” que fortalece entre otras, tareas de procesamiento de lenguaje natural.

### 5.4.1. Arquitectura del modelo

Fig. 7

Los Transformer.



Para comprender el funcionamiento de la arquitectura es necesario abordar algunos temas esenciales: La tokenización, los “*embedding*”, el codificador posicional y los bloques codificador y decodificador de donde son inherentes los “mecanismos de atención”.

Como la mayoría de los modelos de inteligencia artificial, lo primordial es preparar la información de entrada, esta se transforma a valores numéricos, de otra manera no puede ser interpretada, a este proceso se le denomina tokenización.

#### **5.4.2. Tokenización**

Para alimentar modelos Transformers se toman grandes volúmenes de lenguaje donde exista la mayor representación de vocabulario como si de un diccionario se tratara, este proceso se hace de manera intensiva puesto que se deben recorrer corpus lingüísticos completos en búsqueda de palabras. La tokenización es la acción de tomar lenguaje escrito, dividirlo por palabras(tokens) y relacionar cada una de estas a unos valores.

#### **Fig. 8**

##### ***Extracción de texto a tokenizar***

Los instrumentos en la sonda están diseñados para que en el breve paso sobre Plutón y Caronte se obtenga la mayor información posible, como por ejemplo la composición y comportamiento de la atmósfera, la forma en que el viento solar interactúa con la misma, los elementos geográficos.<sup>12</sup>

Suponiendo que se quiere tokenizar una frase en particular, extraída de un conjunto de texto: 'la composición y comportamiento de la atmósfera', el resultado que se espera son las palabras individuales y sus índices, que son los valores que tendrá realmente en cuenta el modelo como lo muestra la tabla 2.

Tabla 2

## Representación de los inputs

Texto de muestra	Índice
a	0
.	.
andrajo	26
.	.
<b>atmosfera</b>	<b>128</b>
.	.
<b>composición</b>	<b>346</b>
.	.
<b>comportamiento</b>	<b>402</b>
.	.
costas	460
.	.
<b>de</b>	<b>589</b>
.	.
<b>la</b>	<b>1023</b>
.	.
<b>y</b>	<b>5400</b>
.	.
zumbido	6012

La numeración de los tokens no son valores aleatorios, estos responden a los índices de cada una de las palabras que el modelo encuentra dentro de los corpus de entrenamiento. Cuando se hace una representación de las entradas (*inputs*), lo que busca el modelo en primera instancia es evaluar cada una de las palabras, si ya ha pasado antes por el modelo, se recurrirá al índice asociado que estará almacenado en memoria (Tabla 2). En cambio, si en el entrenamiento una palabra no se encuentra, se le asignará un nuevo índice quedando registrada para uso posterior. De esta manera, cuando se quiera procesar nuevas oraciones, se hace una consulta a los índices ya encontrados.

Tabla 3

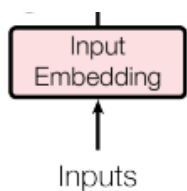
## Tokenización

<b>Palabra:</b>	la	composició n	y	comportamient o	de	la	atmosfer a
<b>Índice:</b>	1023	346	5400	402	589	1023	128

También es de notar que dentro de una frase es posible que se repitan los índices como el ejemplo representado en la (Tabla 3) donde se encuentra el mismo valor 1023, debido a que en efecto se refiere al mismo término “la”.

Lo anterior explica cómo se representan los inputs en el modelo, sin embargo, estos índices no contienen ninguna información de las relaciones que puedan tener las palabras entre sí, o de la posición en la frase, por lo que se hace necesario el concepto de Word Embedding que en español traduciría incrustación de palabras.

### 5.4.3. Word Embedding



La técnica de Word Embedding o incrustación de palabras tiene la finalidad de dar contexto a la aplicación de las similitudes existentes, o cercanía entre palabras. Con los índices asignados y almacenados se da paso a dar dimensionalidad a cada token generando un vector a partir de ellos, con un tamaño estandarizado en cuanto a sus componentes. En función del modelo que se aplique estos pueden ser de 100, 300, o 500 dimensiones, valor personalizado por el diseñador. Descrito por el artículo original (*Ashish Vaswani, et al., 2017*), el valor que se tomó en las implementaciones de las redes Transformers es de  $d=512$  (figura 4), en

principio estos almacenan valores que no tienen mayor utilidad, sin embargo, luego del entrenamiento el algoritmo debe encontrar las similitudes semánticas y sintácticas que le caracterizan en múltiples contextos, esta vez en una representación más densa. El mayor beneficio de la creación de las representaciones densas es la generalización que se puede lograr, las palabras están relacionadas por distintos patrones, la mayoría de ellos imperceptibles para los humanos, sin embargo, si encontráramos estos patrones, podríamos capturar esta información y posteriormente clasificarla de alguna manera. (Goldberg, 2017).

**Tabla 4**

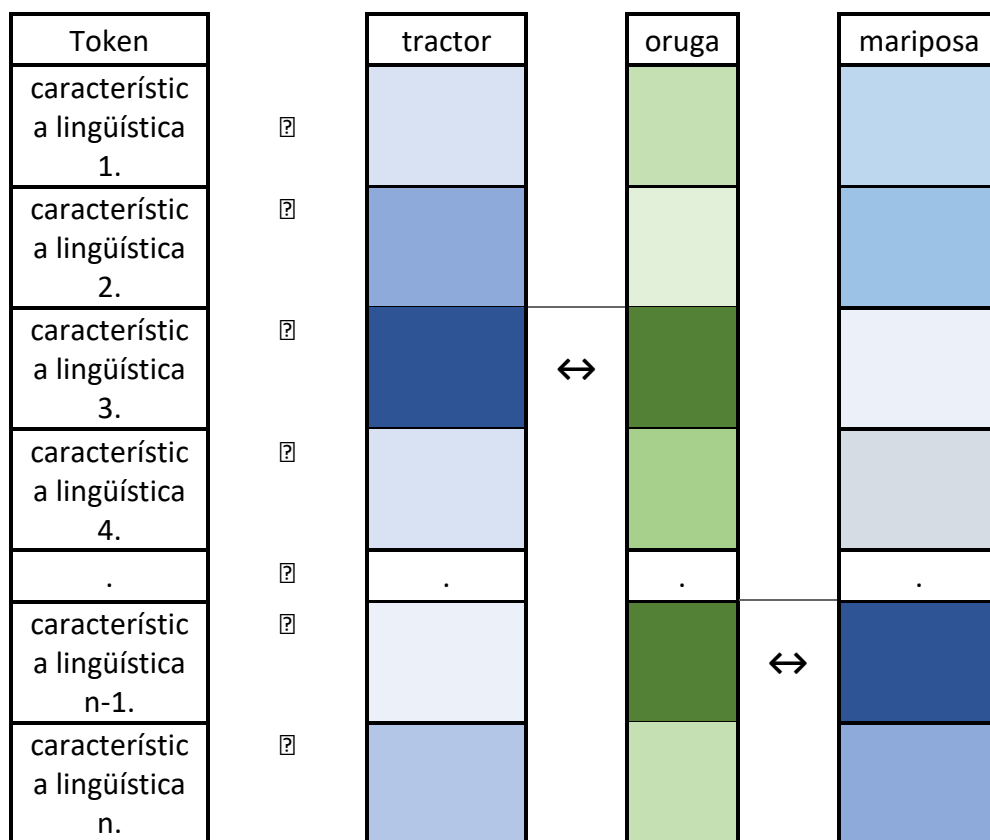
**Componentes vectoriales.**

	la	comp osición	y	com port amie nto	de	la	atm osfe ra
	<b>1023</b>	<b>346</b>	<b>5400</b>	<b>402</b>	<b>589</b>	<b>1023</b>	<b>128</b>
d = 1	0,37	0,29	-0,13	0,36	-0,04	0,37	0,15
d = 2	-0,10	0,19	0,55	-	-0,09	-0,10	-
d = 3	-0,79	0,04	0,87	-	0,38	-0,79	0,69
d = 4	0,18	-0,94	-0,31	-	0,14	0,18	0,62
d = 5	-0,51	0,37	-0,74	-	-0,69	-0,51	0,02
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
d = 511	-0,66	-0,66	0,51	0,45	-0,59	-0,66	-
d = 512	0,10	-0,02	0,90	0,03	-0,02	0,10	0,85

El objetivo de capturar la semántica y sintáctica en espacios vectoriales lleva a que el término oruga, por ejemplo, se conecte con palabras tan distantes como tractor o mariposa como es relacionado en la figura 5, dependiendo del contexto en el que se encuentre, bien cuando se hace referencia a oruga la larva de la mariposa, o en un contexto muy distinto a rodamiento oruga, el sistema de tracción que utilizan algunos tractores y otros vehículos como bulldozer y tanques militares.

**Tabla 5**

**Incrustación de palabras en escala de colores.**



Los pesos típicamente ajustados post entrenamiento nos pueden relacionar oruga y tractor en un espacio (característica lingüística 3) mientras oruga y mariposa en otro espacio (característica lingüística n-1), sin que esto signifique que entre tractor y mariposa estén relacionados directamente.

Al tratarse de lenguaje, un modelo que tome las posibles combinaciones entre el vocabulario y la forma en que se relacionan entre todas y cada una de las palabras es virtualmente imposible, además de que resultaría en una cantidad absurda de información, por ello se han implementado distintos métodos de Word embedding que densifican las características más significativas, se debe tener presente que la etapa de procesamiento Word embedding no es exclusiva de las redes Transformers, esta etapa de procesamiento es análoga en múltiples arquitecturas de redes neuronales (*Tomas Mikolov, Kai Chen, Greg Corrado, Jeffrey Dean, 2013*). Actualmente existen distintas técnicas y algoritmos de Word embedding:

#### **Basado en frecuencia**

Como:

- COUNT VECTOR
- TF-IDF VECTOR
- GLOVE

Técnicas típicamente aplicadas usando la factorización matricial.

#### **Basado en predicciones**

Como:

- CBOW
- SKIP-GRAM

Técnicas donde se usan las redes neuronales prealimentadas (Feedforward).

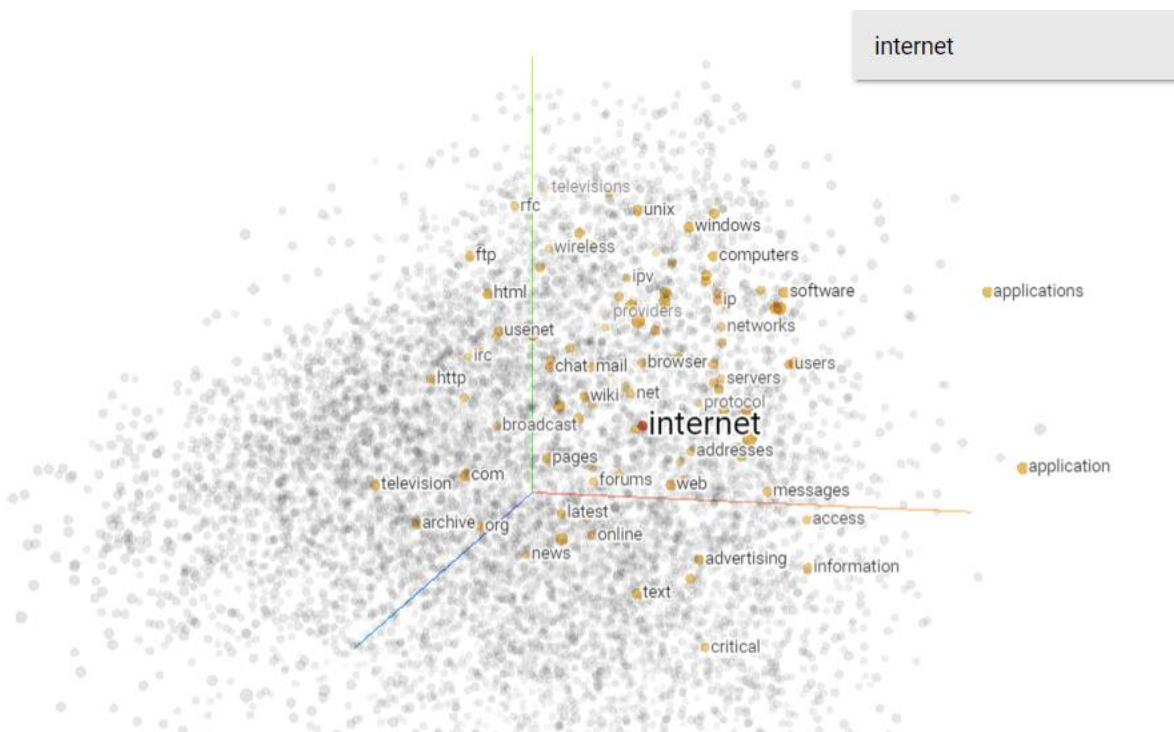
Todas ellas con el mismo objetivo extraer información que pueda ser de utilidad en el tratamiento del lenguaje.



Gráficamente podemos ubicar las palabras en un sistema de referencia con limitaciones, puesto que solo podemos hacer representaciones en un espacio tridimensional, la Fig. 6 muestra el tablero generado por Tensorboard (*tensorflow, 2021*). Una herramienta para visualizar e interpretar incrustaciones de palabras, esto nos da una idea de cómo se localizan las palabras a partir de unos valores numéricos, en el ejemplo se toma la palabra internet y sus pares calculados en el idioma inglés.

**Fig. 9**

**Representación gráfica Tensorboard, internet como núcleo relacional**



#### 5.4.4. Codificación posicional



Luego de la intervención de la capa de incrustación se da paso a la capa de codificación posicional, como es aclarado antes los Transformers no procesan la información de manera secuencial, esto ofrece una gran ventaja en disminución de tiempos de análisis y entrenamiento, por contrapartida, se pierde información crítica del orden por cómo está conformado el texto de entrada, el orden de las palabras es muy importante. Por ejemplo, no es lo mismo tener la oración “el perro se ha comido la galleta”, que “la galleta se ha comido el perro”, el modelo de inteligencia artificial podría confundir la semántica de ambos, después de todo las dos frases tienen exactamente las mismas palabras, la diferencia es el orden de estas, cuya semántica hace que cambie radicalmente.

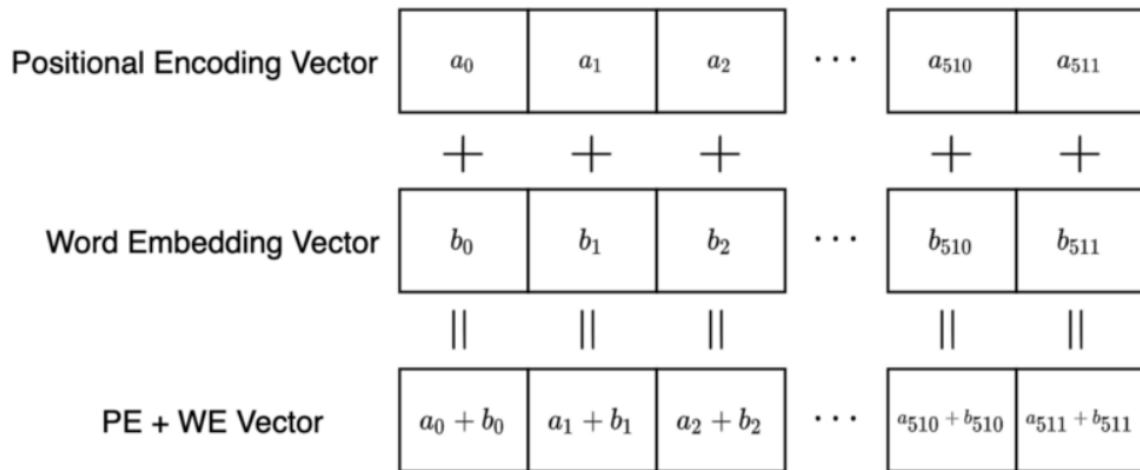
Se podría intuir entonces agregar en la entrada un valor numérico a cada token y así se identifica la posición en la frase, sin embargo, existen distintas razones para evitar hacer este proceso, la principal se debe a que en secuencias largas la magnitud de los valores puede aumentar en exceso.

Así como en la incrustación de palabras se usan 512 dimensiones, la codificación posicional debe tener la misma cantidad de elementos. Los codificadores posicionales tienen la misma dimensión  $d_{model}$  que la incrustación de palabras, entonces, ambos pueden ser

sumados. (Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaise., pag. 6, 2017).

**Fig. 10**

**Sumatoria del codificador posicional e incrustación de palabra.**



Para informar al modelo de las posiciones de las palabras, el principio que se usa en la arquitectura está basado en las funciones seno y coseno a diferentes frecuencias.

$$PE_{(pos,2i)} = \text{sen} \left( \frac{pos}{10000^{\frac{2i}{d_{model}}}} \right) \quad (2)$$

$$PE_{(pos,2i+1)} = \text{cos} \left( \frac{pos}{10000^{\frac{2i}{d_{model}}}} \right) \quad (3)$$

Considerando una frase de longitud L donde,

- $pos$  es la posición de la palabra en la frase desde 0 hasta L-1
- $i$  se usa para mapear los valores del Word embedding, 512 para el modelo propuesto.
- $10000$  es la longitud de onda desde  $2\pi$  hasta  $10000.2\pi$
- $d_{model}$  es el tamaño del Word embedding, valor que va desde 0 hasta 511

Por cada par de elementos se calcula el ángulo en radianes  $\frac{pos}{10000 \frac{2i}{512}}$  alternando

entre la función seno y coseno; debido a que se tiene un tamaño de  $d=512$  en total se calcularían 256 senos y 256 cosenos por cada posición en la frase.

En un ejemplo de implementación de longitud L, la codificación de la primera palabra  $PE(0)$  sería:

$$\begin{aligned} PE(0) &= \left( \sin\left(\frac{0}{10000 \frac{0}{512}}\right), \cos\left(\frac{0}{10000 \frac{0}{512}}\right), \sin\left(\frac{0}{10000 \frac{2}{512}}\right), \cos\left(\frac{0}{10000 \frac{2}{512}}\right), \dots, \sin\left(\frac{0}{10000 \frac{510}{512}}\right), \cos\left(\frac{0}{10000 \frac{510}{512}}\right) \right) \\ &= (\sin(0), \cos(0), \sin(0), \cos(0), \dots, \sin(0), \cos(0)) \\ &= (0, 1, 0, 1, \dots, 0, 1) \end{aligned}$$

Mientras que la segunda palabra opera de la siguiente manera:

$$\begin{aligned} PE(1) &= \left( \sin\left(\frac{1}{10000 \frac{0}{512}}\right), \cos\left(\frac{1}{10000 \frac{0}{512}}\right), \sin\left(\frac{1}{10000 \frac{2}{512}}\right), \cos\left(\frac{1}{10000 \frac{2}{512}}\right), \dots, \sin\left(\frac{1}{10000 \frac{510}{512}}\right), \cos\left(\frac{1}{10000 \frac{510}{512}}\right) \right) \\ &= (0.8414, 0.5403, 0.8218, 0.5696, \dots, 0.0001, 0.9999) \end{aligned}$$

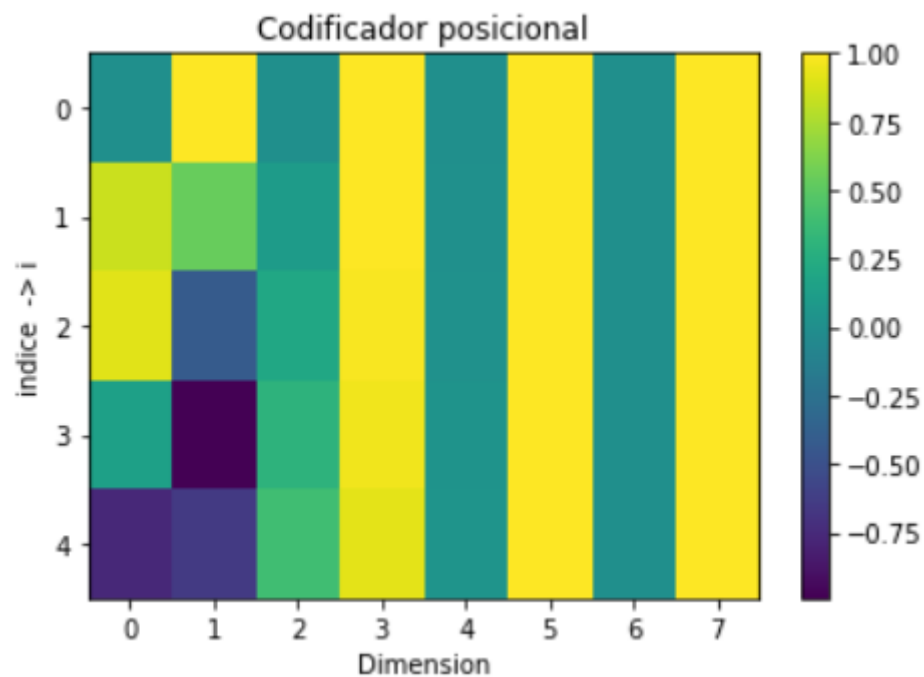
Así sucesivamente hasta que sea calculada la totalidad del texto de longitud L.

Estos resultantes muestran del porqué se eligieron las funciones seno y coseno como indicadores posicionales, quienes varían entre los valores [-1,1] esto es un tipo de normalización, donde no se tiene el riesgo que estos aumenten sin control; además que se genera un patrón que le modelo es capaz de interpretar (Anlin Qu, Jianwei Ni, Shasha Mo.,

2021). La figura 11 da una perspectiva de las variaciones que presentan estos resultados con un ejemplo muy simplificado con una oración de 5 palabras.

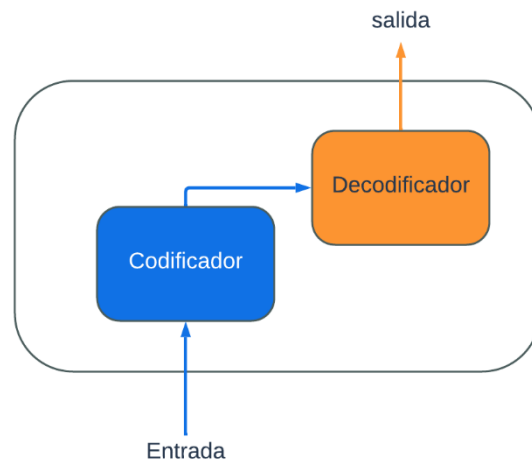
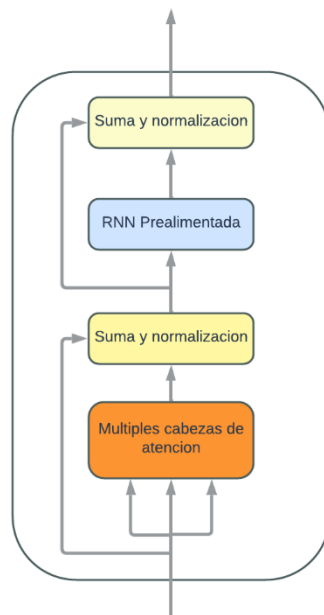
**Fig. 11**

**Codificación Posicional de 5 elementos  $L(i)$ -  $[0,4]$ , dimensión del Word embedding  $d_{\{model\}}$   $[0,7]$**



#### 5.4.5. Bloques codificador y decodificador

Dentro de los bloques codificador y decodificador se aloja el mayor trabajo que los Transformers hacen, las múltiples cabezas de atención. En otras palabras, qué parte del conjunto de texto debería recibir más atención, estos parten de la información generada por la incrustación de palabras junto con la posición relativa adicionada por el codificador posicional.

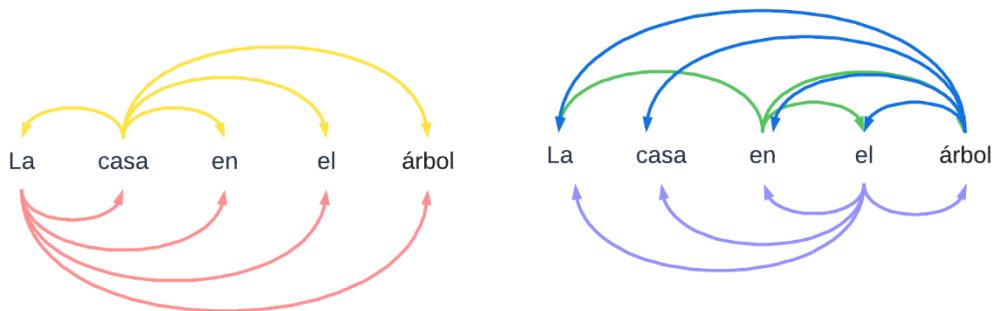
**Fig. 12****Bloques Codificador-decodificador****Fig. 13****Bloque codificador**

Aunque similares, el bloque codificador y decodificador tienen ciertos elementos que lo diferencian por la manera en que operan.

Este bloque cumple la función de tomar un token recorrer toda la secuencia y detectar a cuál de ellos dar mayor atención, esto es, en un fragmento de texto se toma cada token se evalúa el mismo, al tiempo que lo hace respecto al resto de los tokens que conforman la oración, donde se le da unos pesos para identificar a que tokens dar mayor atención.

**Fig. 14**

***Múltiples cabezas de atención***



Para encontrar estas relaciones se usaron 3 distintas capas lineales, los valores Query[Q], Key[K] y Value[V], Consulta, llave y valor respectivamente.

Un razonamiento intuitivo de esta lógica es descrito en el foro StackExchange, "Cuando se busca videos en Youtube, el motor de búsqueda mapeará su consulta [Q] (texto en la barra de búsqueda) contra un conjunto de claves [K] (título del vídeo,

descripción, tags etc.) asociadas con videos candidatos en su base de datos, luego le presentará los mejores videos coincidentes [V] (valores)." (stats, 2019).

Es el producto punto de escalares aplicado a todo el conjunto de consultas de manera simultánea:

$$attention(Q, K, V) = softmax\left(\frac{QK^T}{\sqrt{d_k}}\right)V \quad (4)$$

Donde:

Teniendo el vector consulta[Q] de tamaño m, junto con el vector llave[K] de tamaño n, el producto punto entre ambos genera una matriz m x n.

El elemento  $\sqrt{d_k}$  es la división de la matriz entre la dimensión del vector [K], quien tiene la misma cantidad de elementos al vector [Q], esta función escalar busca disminuir los valores que resultan del producto entre los vectores Q, K; esta se hace antes de aplicar softmax para contrarrestar la magnitud de los valores grandes que pueden afectar la función.

Se normalizan los valores con la función softmax (ecuación 5), escalando los valores entre 0, y 1 para que puedan interpretarse como probabilidades.

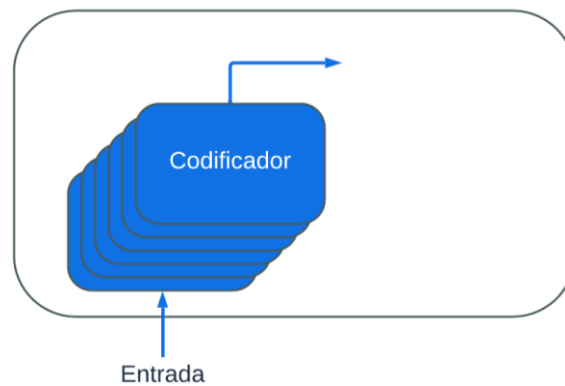
$$\sigma(\vec{z})_i = \frac{e^{z_i}}{\sum_{j=1}^K e^{z_j}} \quad (5)$$



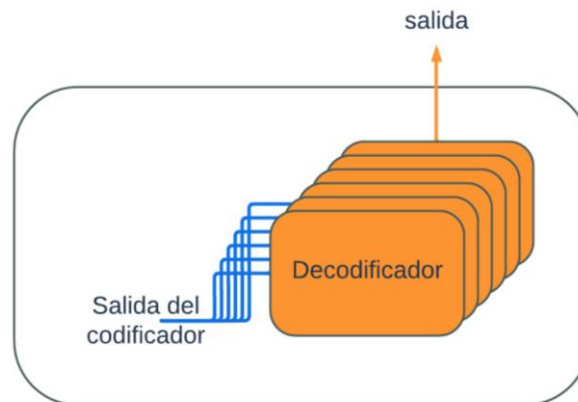
Esta codificación en la práctica es una pila de codificadores, la salida de un bloque codificador se utiliza secuencialmente para otro codificador hasta completarse la totalidad de los 6 (figura 15), todos ellos generando una codificación cada vez más abstracta. Esta es la definición de múltiples cabezas de atención, cada una de las capas codificadoras se ejecuta en un entorno independiente, analizando de manera diferente el conjunto de texto por lo que se arrojan relaciones de atención distintas para cada caso, esta es utilizada para computar finalmente la estructura del texto.

**Fig. 15**

***Pila de bloque Codificador***



Por su parte el decodificador usa la única salida que resultó de los bloques codificadores, esta vez generando una copia y distribuyéndola para los 6 distintos bloques decodificadores (figura 16).

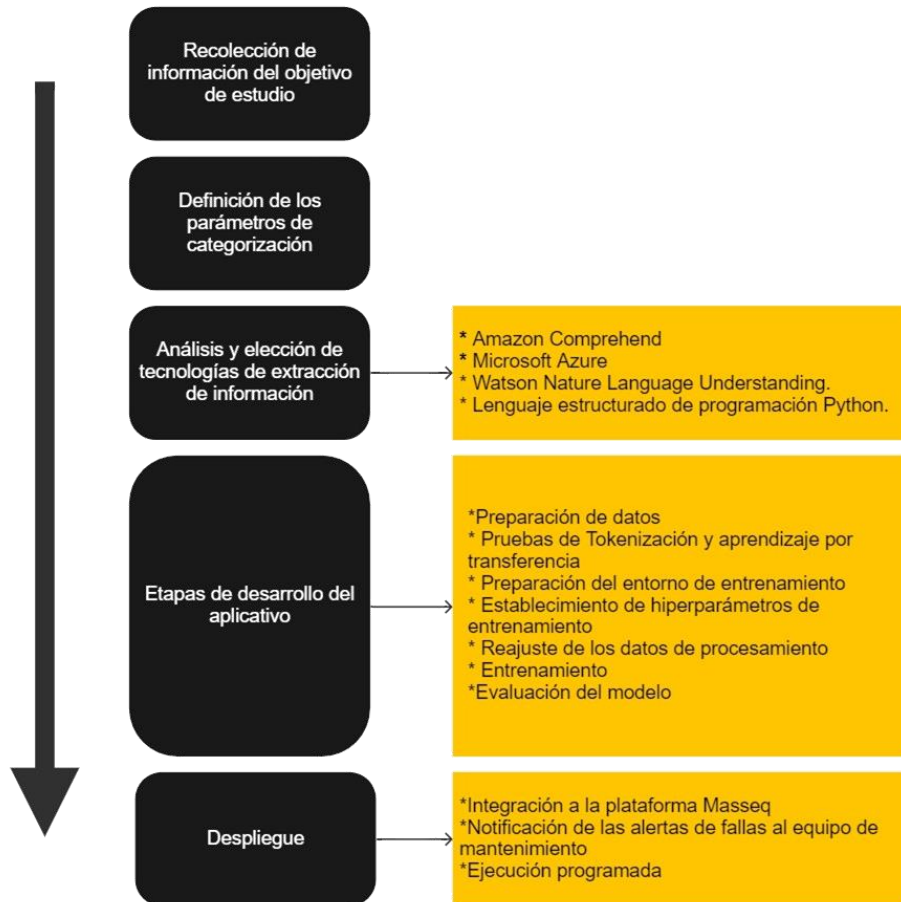
**Fig. 16*****Pila Decodificadora***

De la secuencia codificadora resultan valores abstractos que por fuera de la red neuronal no existe interpretación humanamente comprensible, pues bien, el decodificador traduce esta información y la interpreta para generar una salida que tenga sentido, al finalizar linealiza los datos extraídos y aplica la función softmax para generar el valor ya normalizado.

## **6. Metodología**

El trabajo fue desarrollado siguiendo una serie de fases (Etapas) que fueron establecidas desde el planteamiento del problema, estas, están basadas en métodos de extracción de información y minería de datos. Las fases están resumidas en la figura 17.

Fig. 17

**Etapas de desarrollo del proyecto****6.1. Recolección de información del objeto de estudio**

El desarrollo de la primera etapa propuesta consiste en la recolección de la información desde las distintas fuentes que provee la compañía. El planteamiento por solucionar compete particularmente a dos áreas, el área de mantenimiento y el área de minería de datos; con el fin de ilustrar la relación que tienen, es necesario precisar que mientras el área de mantenimiento es mayormente operativa, minería de datos se encarga de procesos administrativos.

Mantenimiento tiene una serie de actividades que comprende la prevención y reparación de fallos mecánicos en equipos de todo tipo, por mencionar algunos, volquetas, tractocamiones, cargadores, bulldozer, camionetas, entre otros. Las remisiones de equipos que requieran de la intervención mecánica se hacen a través de la gestión del área de minería de datos, como ya se ha mencionado antes, los parámetros que agrupan mayor cantidad de información son los campos de Novedades y Observaciones, así mismo los más complejos de tratar.

Existe un par de herramientas que permite extraer la información ya consolidada, la primera es recurrir a la base de datos en donde se almacenan los históricos del control diario de la compañía, contiene datos relacionados a las fechas, identificación de equipos, comentarios, y demás documentación de carácter privado, para ello el ingreso requiere de la autorización de permisos administrativos, si bien se consideró desde la formulación del proyecto la extracción de la información directamente desde la base de datos, en primera instancia se recurre a informes generados desde la interfaz gráfica.

La alternativa rápida para disponer de la información se hace por medio de la plataforma web a la que se accede a través de las credenciales proporcionadas por el administrador, allí se posibilita la generación de archivos.xlsx, formato compatible Excel, donde están contenidos igualmente campos predefinidos que pueden ser de utilidad, entre ellos la fecha del reporte, la placa, el operador, y naturalmente los campos de observaciones y novedades.

Tabla 6

**FORMATO DE CONTROL DIARIO**

Fecha	Activo	Operador	Km	Hr	...	Observaciones	Novedades
1/10/2021	ABC123	Gordon Moore	10000	2222	...	Trabaja con normalidad	Solicitud de Mtto.
2/10/2021	ABC123	Gordon Moore	10100	2223	...	Trabaja con normalidad	Solicitud de Mtto.
.	.	.	.	.	.	.	.

Este fichero se puede descargar delimitando un intervalo de fechas específico, individualizar cierta maquinaria o la totalidad de esta, así mismo como su ubicación geográfica. A efectos prácticos, debido a que el prototipo contempla todo tipo de equipos, el único filtro establecido es el rango de fechas lo suficientemente amplio que ofrezca una panorámica del contenido que es digitado en los reportes de control diario. El formato contiene una tabla cuyas filas corresponden a un único reporte de control diario, por tanto, de filtrarse la fecha cronológicamente, y de un equipo en concreto es posible apreciar el balance del funcionamiento de los equipos a través del tiempo.

## **6.2. Definición de los parámetros de categorización de prioridad en el tratamiento de los reportes de control diario**

Dentro de los requisitos plasmados originalmente uno de los objetivos para la implementación del algoritmo es que cada una de las notificaciones de fallos tenga cierta categorización dependiendo de su relevancia, la finalidad de esto sería dar prioridad a uno u otro equipo en la entrada al taller, por lo que es necesario aplicar un modelo de

tratamiento de datos que sea capaz de clasificar los reportes, después de todo, procesar tal cantidad de información hace inviable cualquier intento de revisión.

Discriminar el tipo de fallo de los equipos implica que el área de mantenimiento defina detalladamente buena parte de los defectos que puedan incidir en los equipos, es una de las primeras discusiones que se tienen al hacer contacto con el área.

**Tabla 7**

**PROPUESTA DE ESCALA DE PRIORIDADES**

<b>Descripción del fallo</b>	<b>Prioridad</b>
Fuga de aceite en el motor	1
Stop izquierdo para cambio, cristalizado.	3
Aire acondicionado no enfría bien.	5
Silla del conductor en muy mal estado.	4
Hoja de resorte partida.	2

La propuesta que se dio de ejemplo en la tabla 7, se basa en comentarios reales que se encuentran en los reportes, y se definen 5 categorías arbitrariamente para examinar la factibilidad de implementación, en donde 1 es de mayor prioridad, y 5 de menor. En este punto luego de plantear esta posibilidad se tiene una serie de observaciones que rechazan la idea por las razones descritas a continuación:

- a) Definir fallos comunes que se abordan en el taller involucra horas de trabajo de mecánicos y técnicos que no pueden ser asumidos.
- b) Categorizar al menos los principales fallos de un único equipo llevaría un tiempo desproporcionado.
- c) Existen muchos fallos que difícilmente se podrían ponderar en cualquier escala.

- d) Las políticas de la compañía dictan que los activos que estén operando deben estar en perfectas condiciones, por tanto, la prioridad entre “una hoja de resorte partida”, y “el aire acondicionado con fallas” tiene una escala muy similar para las directrices empresariales.

Adicionalmente la documentación que se encuentra en internet de clasificación de fallos bajo estos criterios es limitada, más allá de la solución de problemas a partir de la aplicación de redes neuronales a equipos y el mantenimiento de ellos, como la que plantean Wilmer Contreras et al., donde se diagnostican fallas mecánicas en motores con gran efectividad (CONTRERAS URGILES y otros, 2019). Este tipo de aplicaciones son muy puntuales y no son aplicables para los requerimientos establecidos, debido a la diversidad de fallos, y tipo de equipos que administra una compañía en este sector económico, además de ello, un análisis independiente que definiera estos parámetros como parte de la pasantía se sale del objetivo central.

En opinión del jefe del área de minería de datos, la clasificación se debe dar de una manera simplificada, el debate se orienta esta vez por las categorías que ya están establecidas por la compañía para diferenciar el tipo de reportes (figura 8), donde cada uno de los equipos tiene una codificación única en su campo “Activo”.

**Tabla 8**

### **CODIFICACIÓN DE ACTIVOS**

<b>MA-CA-LF130</b>		
<b>MA</b>	<b>CA</b>	<b>LF130</b>
Maquinaria Amarilla	Cargador	Código o Placa

Separadas por guiones simplifica las siglas que representan el tipo de vehículo. Además del ejemplo de la tabla existen otras 3 categorías, VT- vehículos de transporte, ES – equipos de soporte, EM – equipos menores. Extraer esta información de las bases de datos permite un tipo de clasificación efectiva, por lo que se tendrá en cuenta en la implementación del algoritmo.

En síntesis, las alertas deberían anexar información que notifique el o los mensajes de fallo, la fecha del reporte, y diferenciarlo por la categoría, si se trata de maquinaria amarilla [MA], vehículos de transporte [VT], equipos de soporte [ES] y equipos menores [EM].

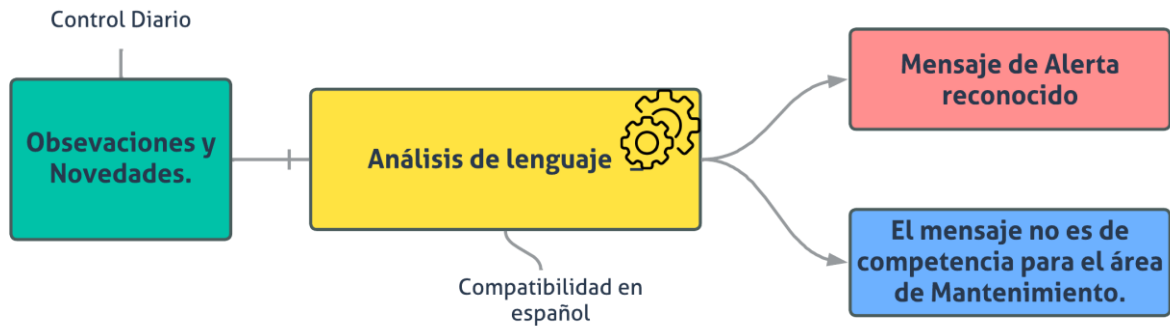
### **6.3. Análisis y elección de tecnologías de extracción de información**

Antes de exponer la comparativa de las tecnologías capaces de procesar lenguaje es imprescindible identificar los requisitos y limitaciones con los que se cuenta en base a los recursos que dispone la compañía.

En primer lugar, en concordancia con el objetivo general el sistema efectivamente debe identificar los patrones que constituyen los mensajes que requieren atención, que reconozca la sintaxis en la escritura en el idioma español con el propósito de retener todos los mensajes que sean de importancia y definitivamente descartar todos aquellos ajenos al asunto de estudio.



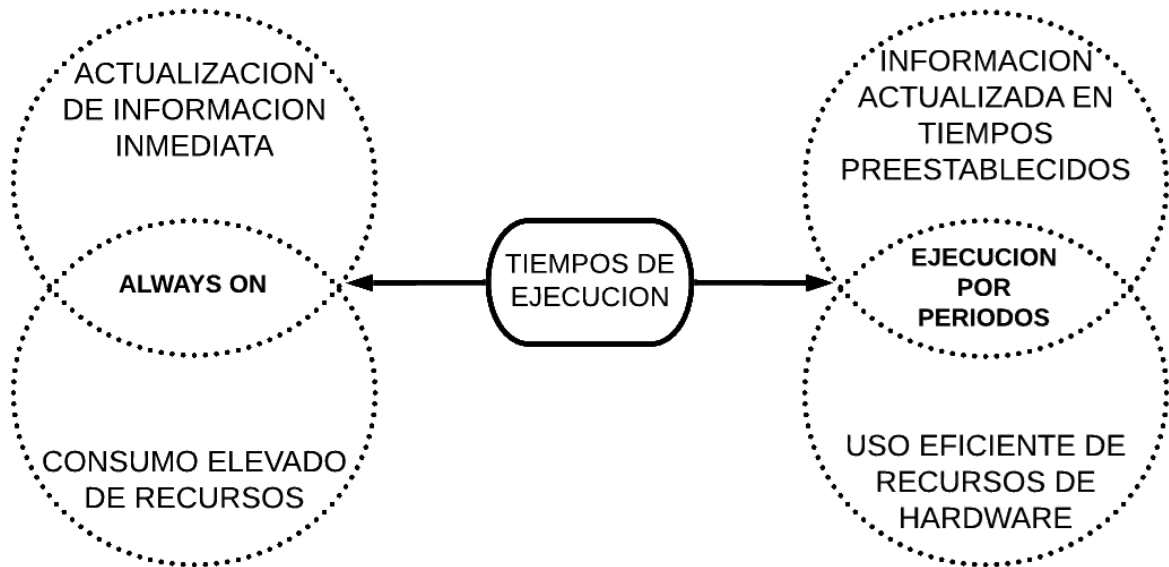
Fig. 18

**Función del procesamiento de lenguaje**

La autonomía en los tiempos de ejecución es importante, el poder iniciar de manera automática sin la necesidad de asistencia de un administrador es uno de los requisitos establecidos para el aplicativo. En función de la arquitectura existen dos premisas de ejecución, la primera que se haga en instantes de tiempo preestablecidos dependiendo de los requerimientos de la operación, por otro lado, está la premisa *always on* o siempre encendida, La disponibilidad de un servicio es la noción de un recurso siempre activo, accesible para ser consultado de manera inmediata, sin que los usuarios puedan percibir retrasos (IBM, s.f.). De igual manera, teóricamente es posible implementar el inicio automático en ambos casos, ahora bien, seleccionar una de las alternativas está condicionada a los requisitos disponibilidad y uso de recursos.

Fig. 19

*Disponibilidad de aplicaciones, (imagen propia).*



Por un lado, se tiene un sistema que siempre está disponible para procesar la información, esto ofrece tiempos de respuesta con retrasos muy cortos, un análisis constante de la información, igualmente un consumo elevado de los recursos de hardware disponible; en cambio la ejecución por periodos define ciertos horarios en los que se debería ejecutar todo el conjunto de datos pendientes por procesar de manera que acumula información y en un instante determinado los evalúa y genera los resultados pertinentes. Esta es una dinámica que debe establecer la compañía conforme a los requerimientos una vez el prototipo esté en funcionamiento.

Es requisito que el servicio sea integrable en la infraestructura que posee la empresa, desde la consulta de información en las bases de datos, hasta el almacenamiento y ejecución del propio sistema, por ello se elige la arquitectura orientada a servicios SOA, esta última es definida como puntos de computación interconectada diseñada para la comunicación entre distintos componentes de software (*IGI Global, 2018*). El SOA se trata de un conjunto de servicios independientes que tienen una función especializada, se podría inferir entonces como una serie de tareas que se ejecutan de forma consecutiva; pero es mucho más que eso, los servicios son micro aplicaciones que permanecen en entornos diferentes, incluso pueden estar programados en lenguajes distintos, sin embargo logran trabajar de manera conjunta sin que se vean afectados unos con otros, por ende deben compartir un protocolo estandarizado de red para el acceso de los datos, con el que se garantiza el intercambio de información, por ejemplo, JSON, REST, SOAP (*ANONIMO, redhat, 2020*). La ventaja de este sistema es que permite que servicios se integren posterior a que el software central ya esté en producción, en pocas palabras, todas las aplicaciones que ya estén ejecutándose permanecerán de esta manera, no necesitan detenerse o suspenderse para poder integrar una herramienta complementaria, por esto es importante que la propuesta de análisis planteado en este informe se fundamentara en la SOA, el algoritmo debió adaptarse al mismo protocolo de transmisión de información, para posteriormente hacer una incorporación sin generar traumatismos.

Eventualmente, es indispensable que la información gestionada por el aplicativo no se vea expuesta a terceros debido a fallos de seguridad, así mismo los datos debieron permanecer integrales en todo momento.

En este punto no hace falta aclarar que el NLP es la tecnología más idónea para el tratamiento del lenguaje en los sistemas informáticos, por ello las consideraciones que se adoptaron se fundamentan en este campo de las ciencias, ahora bien, se pudo recurrir a tecnologías diseñadas por compañías como Amazon, Microsoft, IBM que entre su portafolio de servicios contiene soluciones NLP, sin embargo, desarrollar el algoritmo que cumpliera con los requerimientos respaldado en lenguajes de programación como Python, C++ o R, da flexibilidad y adaptabilidad para proyectos futuros, aun así, se analizaron las primeras alternativas.

Los requisitos y las limitaciones del prototipo fueron definidos en esta etapa, a partir de allí se procedió al análisis de tecnologías en las que podría desarrollarse, entre las posibles candidatas, se compararon las características de los principales servicios escalables a la nube.

### **6.3.1. Aplicaciones de procesamiento de lenguaje natural basadas en la nube**

#### **6.3.1.1. Amazon Comprehend**

De las tecnologías candidatas al tratamiento del lenguaje, se tuvo en cuenta Amazon Comprehend, el pilar de Comprehend es el procesamiento del lenguaje natural en el que se analizan textos en una variedad de servicios como el reconocimiento de entidades personalizadas, clasificación personalizada, extracción de frases clave, análisis

de opiniones, reconocimiento de entidades (AMAZON, 2022); Esta tecnología provee un punto de acceso que puede ser llamada desde otros servicios. A tal efecto debe proporcionársele el documento que contenga lenguaje escrito a analizar, tiene una ventaja es que si se dispone del servicio de almacenamiento en la nube (Amazon S3) directamente para disponer los ficheros que deseen procesarse (AMAZON 2, 2022), sin necesidad de transformar a formatos de archivo compatible.

Tabla 9

#### CARACTERÍSTICAS DE AMAZON COMPREHEND

Característica	Descripción	Idioma
<b>Detección del idioma dominante</b>	Examinar el texto para determinar el idioma dominante.	alemán, inglés, español, italiano, portugués, francés
<b>Detección de entidades</b>	Detecta referencias textuales a los nombres de personas, lugares y artículos, así como referencias a fechas y cantidades.	alemán, inglés, español, italiano, portugués, francés
<b>Detección de frases clave</b>	Encuentra frases clave como «buenos días» en un documento o conjunto de documentos.	alemán, inglés, español, italiano, portugués, francés
<b>Detección de información de identificación personal</b>	Analiza documentos para detectar datos personales que podrían utilizarse para identificar a una persona, como una dirección, un número de cuenta bancaria o un número de teléfono.	alemán, inglés, español, italiano, portugués, francés
<b>Determinar sentimientos</b>	Analiza los documentos y determina el sentimiento dominante del texto.	alemán, inglés, español, italiano, portugués, francés
<b>Análisis de sintaxis</b>	Analiza las palabras del texto y muestra la sintaxis de voz de cada palabra y permite comprender el contenido del documento.	alemán, inglés, español, italiano, portugués, francés
<b>Temática</b>	Busca el contenido de los documentos para determinar temas y temas comunes.	alemán, inglés, español, italiano, portugués, francés

Una característica destacada es que posee AWS KMS, una tecnología propietaria en donde los datos están cifrados lo que proporcionan mayor seguridad; por contraparte Amazon Comprehend tiene un costo individual denominado unidades, en donde 1 unidad equivale a 100 caracteres, además de costos relacionados a la administración.

### 6.3.1.2. Microsoft Azure

Microsoft Azure es una plataforma que provee una serie de herramientas que permite a desarrolladores adentrarse en aplicaciones de la Inteligencia Artificial, provee una serie de puntos de acceso que posibilita la creación de aplicaciones inteligentes, en donde no se requiere de habilidades y conocimientos especializados en IA o ciencia de datos (*microsoft, 2022*). Cuenta con una serie de características como el etiquetado personalizado, extracción de frases o el reconocimiento de entidades con nombre NER (Tabla 10). Estos han sido denominados minería de soluciones, dispone igualmente de un repositorio abierto para su estudio (*microsoft, 2022*).

**Tabla 10**

#### **CARACTERÍSTICAS DE MICROSOFT AZURE NLP**

Escenario	Modelo	Descripción	Idioma
Clasificación de texto	BERT, DistillBERT, XLNet, RoBERTa, ALBERT, XLM	La clasificación de texto es un método de aprendizaje supervisado para aprender y predecir la categoría o la clase de un documento dado su contenido de texto.	Inglés, chino, alemán, francés, japonés, español, holandés
NER	BERT	El reconocimiento de entidades con nombre (NER) es la tarea de clasificar palabras o frases clave de un texto en entidades predefinidas de interés.	Inglés

Resumen de texto	BERTSumExt BERTSumAbs UniLM (s2s-ft) MiniLM	El resumen de texto es una tarea de generación de lenguaje de resumir el texto de entrada en un párrafo de texto más corto.	Inglés
Vinculación	BERT, XLNet, RoBERTa	La implicación textual es la tarea de clasificar la relación binaria entre dos textos en lenguaje natural, texto e hipótesis, para determinar si el texto está de acuerdo con la hipótesis o no.	Inglés
Preguntas y respuestas	BiDAF, BERT, XLNet	La respuesta a preguntas (QA) es la tarea de recuperar o generar una respuesta válida para una consulta determinada en lenguaje natural, siempre que se proporcione un pasaje relacionado con la consulta.	Inglés
Similitud de frases	BERT, GenSen	La similitud de oración es el proceso de calcular una puntuación de similitud dado un par de documentos de texto.	Inglés
Embeddings	Word2Vec fastText GloVe	La incrustación es el proceso de convertir una palabra o un fragmento de texto en un espacio vectorial continuo de número real, por lo general, en baja dimensión.	Inglés
Análisis de sentimientos	Dependency Parser GloVe	Proporciona un ejemplo de entrenamiento y uso del análisis de sentimiento basado en aspectos con Azure ML e Intel NLP Architect.	Inglés

Todas estas pueden consultarse en el GitHub de código abierto de Microsoft (*microsoft, 2022*), adicionalmente de elegirse soporte, existe unos costos asociados a la administración de los servicios y por volumen de elementos analizados los cuales varían en función del escenario que se use.

### 6.3.1.3. Watson Natural Language Understanding

Con características similares a las que ofrecen Comprehend y Azure, Watson es capaz de comprender conceptos, entidades, palabras clave, sentimientos, también admite

la creación de modelos muy específicos a partir de datos no estructurados (Tabla 11).

Tiene gran variedad de compatibilidad con distintos SDK (kit de desarrollo de software)

entre los que destacan JAVA, Node.js, Python entre otros. (cloud.ibm, 2022), por lo que es

posible embeber fácilmente en software a otras plataformas.

**Tabla 11**

**CARACTERISTICAS WATSON NATURAL LANGUAGE UNDERSTANDING**

Característica	Descripción	Idioma
Categorías	Categoriza el contenido utilizando una jerarquía de clasificación de cinco niveles.	inglés, español
Conceptos	Identifica los conceptos de alto nivel que no están necesariamente referenciados de forma directa en el texto.	Inglés, español
Emociones	Analiza la emoción transmitida por frases de destino específicas o por el documento como un todo.	Inglés, español
Entidades	Busca personas, lugares, eventos y otros tipos de entidades mencionados en su contenido.	Inglés, español
Palabras clave	Busca en su contenido las palabras clave relevantes.	Inglés, español
Relaciones	Reconoce si dos entidades están relacionadas, e identifica el tipo de relación.	Inglés, español
Rol semántico	Analiza sintácticamente las frases en forma de sujeto-acción-objeto, e identifica entidades y palabras clave que son sujetos u objetos de una acción.	Inglés, español
Modelos personalizados	Identifica las entidades y relaciones personalizadas exclusivas de su dominio con Watson Knowledge Studio.	Inglés, español



Para el uso de esta herramienta se tiene una serie de tarifas calculadas mediante la cantidad de ítems analizados o unidades de datos, dónde 1 unidad de datos equivale a 10.000 caracteres o menos.

Las 3 tecnologías anteriormente descritas tienen la característica de no necesitar ser programadas totalmente, estas tienen una infraestructura ya montada de la que se puede replicar y hacer adaptaciones a los requerimientos del usuario, cada una de ella necesita una serie de capacitaciones específicas para hacer una implementación satisfactoria.

#### **6.3.1.4. Lenguaje estructurado de programación Python**

Python es un lenguaje de programación de alto nivel ampliamente utilizado en la actualidad, cuenta con una de las sintaxis más simples de programar con una curva de aprendizaje muy corta, es lenguaje interpretado, esto es, que no hace falta compilar su código para que pueda ser ejecutado, por lo que pruebas de ensayo y error resultan ser instantáneas. Es uno de los lenguajes más populares actualmente debido a su versatilidad en el análisis de datos y distintos campos de la Inteligencia artificial, su polivalencia es tal que es posible programar desde una neurona artificial hasta complejos modelos de Deep Learning. Cuenta con un gran número de librerías para el tratamiento del lenguaje en su mayoría de código abierto, se puede mencionar algunas de las más populares como NLTK (*Millstein, 2018*), Spacy (*VASILIEV, 2020*), o el mismo Transformer (*Ashish Vaswani, et al., 2017*).

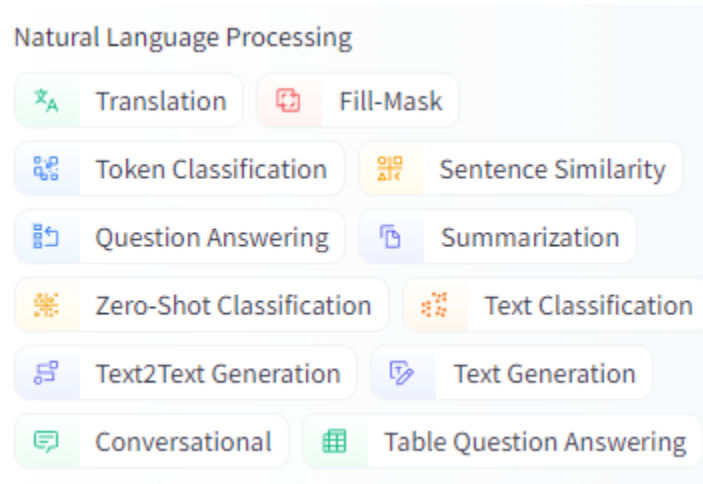
Las tecnologías descritas anteriormente se tomaron en consideración bajo distintos parámetros: La noción de la herramienta, el soporte activo y la suficiente documentación fueron razones de peso, además de ser compatibles a las tecnologías ya instaladas en la compañía, después de todo no sería conveniente formular una arquitectura que pueda ser respaldada por poco personal técnico. La compañía ya cuenta con asistencia proporcionada por externos como Microsoft o Amazon en servicios como correo electrónico, aplicaciones de ofimática, servidores web; se estudió esta posibilidad precisamente por la facilidad que supondría incluir otro servicio más de darse el caso; por otro lado, aunque no se tuviera relación con la multinacional IBM para la implementación de watson knowledge studio, dentro del área reside personal con experiencia en tecnologías relacionadas. Respecto al lenguaje Python es habitual encontrarse con desarrolladores que tengan habilidades en este lenguaje, el área de minería de datos no es la excepción, buena parte de la plantilla ejerce funciones relacionadas con la programación de scripts por lo que ya están habituados a tratar con las instrucciones típicas del lenguaje.

Al margen de las características que tiene cada tecnología en cuestión, en el fondo son muy similares, se toma en cuenta principalmente el desarrollo en Python por encima de las otras debido a que no tiene ningún costo de desarrollo, admitir una arquitectura en constante evolución como los Transformers, y en particular por la enorme variedad de modelos que se pueden implementar; en cualquier situación diseñar una solución en este lenguaje, favorece la creación de elementos complementarios de manera modular, agregar características nuevas e incluso de ser necesario darle nuevo enfoque a futuro sin

que implique rediseñar totalmente el algoritmo o invertir tiempo y esfuerzo en tecnologías distintas. Por estas razones se eligió Python como plataforma de desarrollo del prototipo.

### **6.3.2. Modelos y tareas de los Transformers**

Aunque Transformers es una arquitectura muy poderosa no es la herramienta que manipula los datos con apenas instrucciones, el reconocimiento de frases tiene limitaciones en cuanto a tamaño, idioma o tarea de preentrenamiento. Posterior a que Google compartiera BERT (*DEVLIN y otros, 2018*), han surgido un sin número de soluciones, como RoBERTa (*YINHAN y otros, 2019*), DistilBERT (*SANH y otros, 2019*), diseñadas para aplicaciones en inglés. Todo modelo tiene una serie de tareas para las que fue diseñado dentro del procesamiento de lenguaje natural PLN (Fig. 20), clasificación de texto, traducción, clasificación de tokens, generación de texto, son algunas tareas tomadas en consideración para la implementación en la detección de fallos de la maquinaria.

**Fig. 20****RESUMEN DE TAREAS DISPONIBLES EN HUGGING FACE**

Las consideraciones que se tomaron entre los modelos y las tareas disponibles permitieron hacer pruebas con algunas de ellas empleando el sitio web *HuggingFace* (figura 20) (*huggingface, s.f.*) que reúne una extensa biblioteca de modelos Transformers predefinidos, algunos soportados para el procesamiento de lenguaje natural. El éxito de esta plataforma se debe a que dispone de una gran cantidad de código abierto para su experimentación, donde se encuentran abiertos modelos muy populares desarrollados por empresas como Google, Facebook, OpenAI (cofundada por Elon Musk) entre otros. Esta plataforma ofrece herramientas que permiten aplicar aprendizaje por transferencia, proporcionando las características y capacidades del modelo, así mismo como la ruta de acceso para importación de este, de allí se consideraron distintos tipos de tareas candidatas a cuál podría adaptar más al objetivo de este trabajo, la detección y diferenciación de las frases de fallos en maquinaria y equipos.

### 6.3.2.1. Text generation

La generación de texto es una tarea de inteligencia artificial con la capacidad de redactar párrafos enteros con buen nivel léxico y de estructura, tanto así, que puede asemejarse en gran medida al lenguaje a escrito por personas. El mejor ejemplo de generación de texto es GPT-2 (*openai, 2015-2022*), desarrollada por OpenAI, lograría redactar páginas completas de texto de manera congruente, tuvo tanto éxito al punto que sus desarrolladores anunciaron que sería liberada una versión recortada de la misma, por temor a que fuera mal usada.

El ejemplo de la figura 21 da una idea de las capacidades de esta tarea, en este en particular se da parte de la información con la frase *difficulties in writing articles* (dificultades en la escritura de artículos) y este es capaz de generar una sección completa de texto a partir de ella a manera de complementación. (2, s.f.)

**Fig. 21**

#### Text generation huggingface

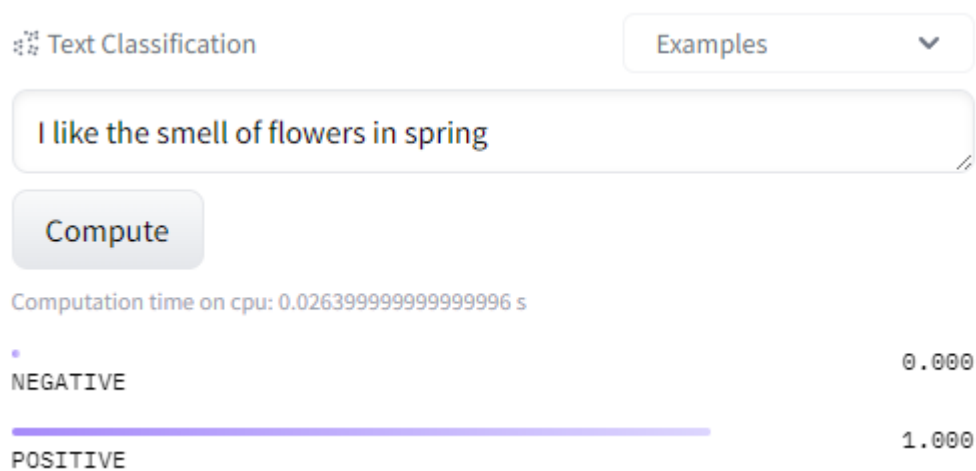


### 6.3.2.2. Text Classification

Funciona estructurando el texto automáticamente bajo ciertas reglas donde es capaz de ubicar los datos en clústeres de información; son comunes las aplicaciones en análisis de sentimientos, como las que encontramos en los comentarios en redes sociales, donde se busca detectar patrones que infieran el estado de ánimo de una persona al momento de hacer una opinión, si se trata de felicidad, tristeza, cansancio, alegría. Ha sido usado, además, para analizar las reseñas como las que contiene *yelp*, una aplicación donde usuarios pueden calificar restaurantes, con la ayuda del *text classification*, el set de datos logra categorizarse efectivamente diferenciando opiniones buenas de las malas, logrando una precisión de hasta el 97.41% (*LI y otros*)

Fig. 22

#### Text Classification huggingface



En la figura 22 se usa un modelo derivado de DistilBERT (huggingface, s.f.), cuya tarea recibe oraciones y este la califica si se trata de una frase positiva o negativa.

En ambos casos las implementaciones pueden servir de base para prototipo a diseñar el planteamiento original de este documento.

#### **6.4. Etapas de desarrollo del aplicativo**

La estructuración del algoritmo resulta muy afín al desarrollo planteado originalmente en la arquitectura Transformer (Ashish *Vaswani*, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaise., 2017).), desde luego, el proyecto al estar sustentado en un tratamiento de información específico es necesario adoptar ciertas medidas para ajustar al requerimiento planteado. A continuación, se describen las etapas y procesos que llevaron al desarrollo del modelo.

##### **6.4.1. Preparación de los datos**

En la ciencia de datos es crucial hacer un preprocesamiento de la información para alimentar los modelos de machine learning. Si bien el origen de los datos con los que se hará el prototipo no está totalmente desestructurado, estos necesitan ser manipulados antes de pasarse como texto de entrenamiento, con ello se espera que la entrada (input) del modelo reciba la mayor cantidad de información útil, y descarte aquella que pueda entorpecer el aprendizaje del modelo.

La información que alimenta el algoritmo tiene dos orígenes principalmente, por un lado, se usa el aprendizaje por transferencia cuya información resulta del entrenamiento de trabajos realizados por terceros, información que será usada en el propio modelo disminuyendo el procesamiento necesario para la detección de texto, esta

será descrita más adelante; por otro lado, está la información originada por la compañía que sufrirá cierto tratamiento antes de introducirse como base de datos.

En primer lugar, se debió solucionar el origen y el formato del archivo que será objeto del procesamiento. Accediendo a la generación de informes directamente desde el aplicativo del que dispone la compañía es posible hacer una exportación de las tablas en formato .xlsx. Con la supervisión del coordinador del área de minería de datos e innovación se dieron los permisos administrativos que habilitan el ingreso al módulo requerido, para la generación de estos informes se precisan de ciertos parámetros de modo que se pueda filtrar:

- La selección de un rango de fecha que extraiga la mayor cantidad de datos útiles. Por análisis previos hechos en el área de minería de datos, lo idóneo es escoger un periodo largo del año en curso, entonces, se estableció el máximo posible que va desde el 01/01/2021, hasta el 25/09/2021 fecha en que se generó el primer informe.
- Filtrar por las categorías de los equipos que se requieran. Como se mencionó antes el procesamiento de los fallos se espera que sea aplicable a todo tipo de equipo y maquinaria que opere en la compañía, por lo que se ha seleccionado la totalidad de estos.

Bajo estos parámetros se exporta el documento que resulta en un formato similar al de la figura 23.



Fig. 23

## Informe generado por plataforma administrativa Maseq

	A	B	C	DEFGHIJKLMNP	Q	R
1	Fecha	Vehiculo	Operador	Observaciones	Novedades	Observación
2	2021-01-05	ES-CM-JNZ758	JAIRO ANDRES RAMOS LIZCANO	F11415		Dia 06/01/2021 parqueadero.
3	2021-01-05	VT-DT-THF879	CESAR AUGUSTO OCHOA OTALORA	F11514		Malas estrias del cuplin del cardan, humedad en el gato del volco, esta fallando a/c.
4	2021-01-05	MA-PT-TM-02	MARIO GARCIA LEGUIZAMO	F11514	48 viajes tamizado de rio y cantera x 13 mts	cierre pantallas molino, inspeccion y encendido generador. hora de inicio produccion 7:25 am. una parada 10 min se tapo bajante banda #4 y parada 2:25 pm se acabo material. Produccion arena natural, arena tipo 1, triturados 3/4", 1".
5	2021-01-05	ES-CQ-TGZ053	MILLER GARCES GUERRERO	F11515		humectacion vias planta y paicito, agua no potable.
6	2021-01-05	VT-TC-TGY999	MILTON ARTUNDUAGA BARRERO	F11514	Entra con Hr.13922 - Km.373908	se cambio bombona de cabina "suspension", votadera de aceite hidraulico, fallas en fanclos se capó en mal estado, volco en mal estado.
7	2021-01-05	VT-DT-TFR244	DUBIAN FREYDY PINTO GARCIA	F11514		Transporte de personal. Panoramico chitiado, no tiene gato hidraulico, no tiene pito, presenta humedad en el cárter.
8	2021-01-05	ES-CT-TGZ082	DINER ANDRES SANCHEZ	F11515		

La figura de referencia muestra el contenido relevante que genera la plataforma web, ciertos campos son ocultados debido a que no son de competencia en esta etapa, en total el archivo está constituido por 305.873 filas que responden cada una a un único registro de control diario de los distintos equipos; obteniendo este fichero se da paso a la transformación de los datos, valiéndose de librerías para el manejo de grandes volúmenes de información, como pandas y numpy.

Para la codificación del algoritmo se eligió el entorno de desarrollo visual studio code (VS code), un editor de código liviano que cuenta con gran variedad de plugins con soporte activo, donde se puede usar una de las herramientas más populares entre desarrolladores de Python, se trata de jupyter notebook, este permite el desarrollo y la ejecución segmentos de código, agilizando la digitación de funciones.

Una de las ventajas que ofrece la librería *Pandas* es la facilidad de llamado e importación de distintos formatos de archivo como .XLSX .CSV, este es capaz de

interpretarlas en un nuevo conjunto de datos denominado **dataframe** (OVHcloud., 2022) sin alterar los valores originales, a partir de allí se pueden aplicar multitud de filtros y transformaciones que permitan modelar la información a conveniencia.

**Fig. 24**

### Importación de .xlsx a dataframe

```

1 import pandas as pd
2
3 info = pd.read_excel('control_diario.xlsx')
4 info

```

	Fecha	Vehiculo	Operador	Proyecto	Lugar	Hora Hombre AM	Total Horas AM	Horas Hombre PM	Total Horas PM	Total Horas	Horometro Inicial	Horometro Final	Horometro Total	Kilometraje Inicial
0	2021-01-05	ES-CM-JNZ758	JAIRO ANDRES RAMOS LIZCANO	Planta Trituradora	Neiva	Inicio: 07:00 Final: 11:59	4 Hora(s) 59: Minuto(s)	Inicio: 13:00 Final: 18:01	5 Hora(s) 1: Minuto(s)	10.000000	0.0	0.00	0.00	8998.0
1	2021-01-05	VT-DT-THF879	CESAR AUGUSTO OCHOA OTALORA	Planta Trituradora	Acopio de Material	Inicio: 06:00 Final: 11:59	5 Hora(s) 59: Minuto(s)	Inicio: 13:00 Final: 17:31	4 Hora(s) 31: Minuto(s)	10.500000	17187.0	17197.00	10.00	327799.0

La información útil para el entrenamiento se reduce a los campos de novedades y observaciones, por lo que el resto de la información a excepción del índice que es inherente de los **dataframe** puede ser omitida. Mientras el fichero original tiene una dimensión 305.873x22, una vez se descarten los otros campos esta conserva la misma cantidad de filas, pero esta vez representada en 3 columnas, el índice las novedades y observaciones.

Fig. 25

## Reducción de dimensiones dataframe

```
1 info[['Novedades', 'Observación']]
```

✓ 0.5s

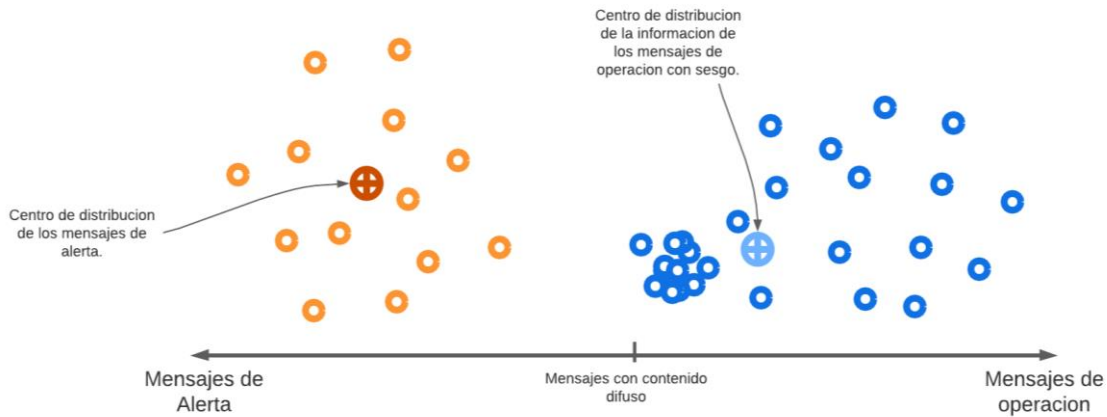
	Novedades	Observación
0	NaN	Dia 06/01/2021 parqueadero.
1	NaN	Malas estrias del cuplin del cardan, humedad e...
2	48 viajes tamizado de rio y cantera x 13 mts	cierre pantallas molino, inspeccion y encendid...
3	NaN	humectacion vias planta y paicito, agua no pot...
4	Entra con Hr.13922 - Km.373908	se cambio bombona de cabina "suspension", vota...
...	...	...

En la figura 25 se pueden notar espacios con valores *NaN*, es una representación que usa python para campos vacíos, este es un tipo de punto flotante “*Not A Number*”, cuya traducción del inglés sería no es un número. Estos valores se encuentran por todo el conjunto de datos de novedades y observaciones, se debe prescindir de ellos para evitar que una vez el modelo se esté entrenando tenga tiempos de lectura en campos nulos, que, aunque no parezcan significativos, cuando se trata de campos masivos estos tienden a acumular pausas y generar retrasos en los barridos de información. Para lograr su eliminación se juntaron dos pasos, el primero, la unión de los campos de novedades y observaciones, en una sola columna ‘mensajes’, posteriormente se descartan los valores *NAN*. Resultante de esta operación la contabilidad de mensajes llega a las 384.660 filas, es decir, la dimensión de la tabla de datos de entrenamiento es de 384.660x2, la columna índice y la columna mensajes.

Ahora bien, analizando el tipo de mensajes que pueden encontrarse dentro de los mensajes recopilados se detectó un patrón que también se repetía constantemente. Se trata de los mensajes con información duplicada, esto no ofrece mayor valor al modelo, en cambio sí representa trabajo adicional, incidiendo en mayores tiempos de entrenamiento. Debe aclararse que la información duplicada no es un error en el tratamiento o generación del informe, esto se debe a los reportes que hacen cuando el equipo está en operación, donde muchas veces se digita el tipo de trabajo realizado de manera consecutiva, por ejemplo la frase “equipo disponible por finalización de obra”, pudo haberse digitado en varios días consecutivos; como el objetivo del entrenamiento en el modelo Transformer es tener una representación de la mayor variedad de la sintaxis en las oraciones, repetir esta puede generar un sesgo en los pesos de entrenamiento del modelo. Típicamente en algoritmos de inteligencia artificial, que un elemento se repita muchas veces puede generar que se le de mayor peso, por lo que puede alterar la concepción del modelo.

Fig. 26

### Patrones de mensajes encontrados en control diario.



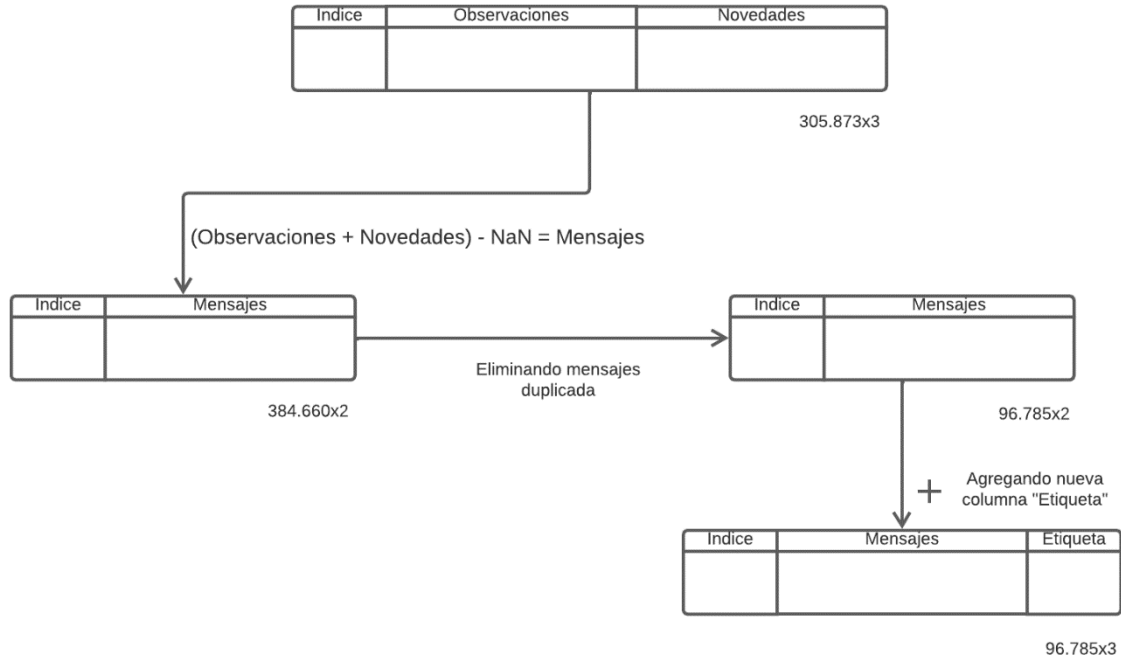
Los datos de la tabla de entrada sin procesar pueden inducir este error, por ejemplo, en la figura 26 se puede apreciar que el centro de distribución de los datos de la operación normal ha sido desplazado a la izquierda debido a gran cantidad mensajes duplicados, acercándolo en gran medida a los valores de mensajes de alerta, por lo que al momento de predicción y posterior puesta a prueba muchos de los mensajes de operación podrían ser detectados como mensajes de alerta. De esta manera se hizo necesario aplicar una función que removiera todos los mensajes con valores coincidentes.

Concluyendo en la preparación de la información se crea la nueva columna 'Etiqueta' con espacios vacíos, estos se hacen para establecer un etiquetado manual, e indicar al modelo de los datos de entrenamiento, específicamente qué mensajes corresponden a fallos para tener en cuenta, y cuales se refieren a información de operación habitual.

La figura 27 resumen las etapas de procesamiento que fueron aplicadas.

**Fig. 27**

**Diagrama resumen del preprocesamiento de datos de entrenamiento.**



El objetivo de esta última columna etiqueta es usar el método de aprendizaje supervisado, en efecto se requieren dos elementos para el entrenamiento, los mensajes y sus etiquetas asociadas. No sin antes definir los parámetros para diferenciar el tipo de mensaje, si deben tomarse en cuenta o no son de importancia al área de mantenimiento.

El sistema de etiquetas se definió de la siguiente manera inicialmente:

- Todo mensaje que contenga al menos una frase que requiera atención, es decir, que mencione problemas entre sus comentarios debe tener la etiqueta 0.
- El grupo de mensajes que informe normalidad en la operación se etiquetan con el valor 1.

Extraemos de ejemplo el mensaje “46 GLS. HR.21040 KM.429142. *“Bastante veriveri en la dirección, ya se paso para el cambio de filtros, revisar frenos, cuando se pasa de 5 a 6 chirrea muy feo.”* En la tabla 12 se puede apreciar que el mensaje se compone de dos partes, una informativa y una de tipo alerta, aun así, el sistema debe analizarla de manera integral. Por lo que se ha acordado el mensaje de alerta siempre va a tener más peso al momento de elección entre las dos opciones.

**Tabla 12**

**MUESTRA DE MENSAJES COMBINADOS**

46 GLS. HR.21040 KM.429142.	Bastante veriveri en la direccion, ya se paso para el cambio de filtros, revisar frenos, cuando se pasa de 5 a 6 chirrea muy feo.
<b>Informativa</b>	<b>Alerta</b>

Con el apoyo del personal de minería de datos se hace un etiquetado manual de todos los comentarios siguiendo el patrón de referencia que muestra la tabla 13, esta etapa es muy importante, puesto a partir de este set de datos se instruye el modelo de procesamiento de lenguaje. El personal del área al estar involucrado diariamente con el tipo de mensajes que exige mantenimiento puede asignar efectivamente estos valores a su categoría correspondiente.

Tabla 13

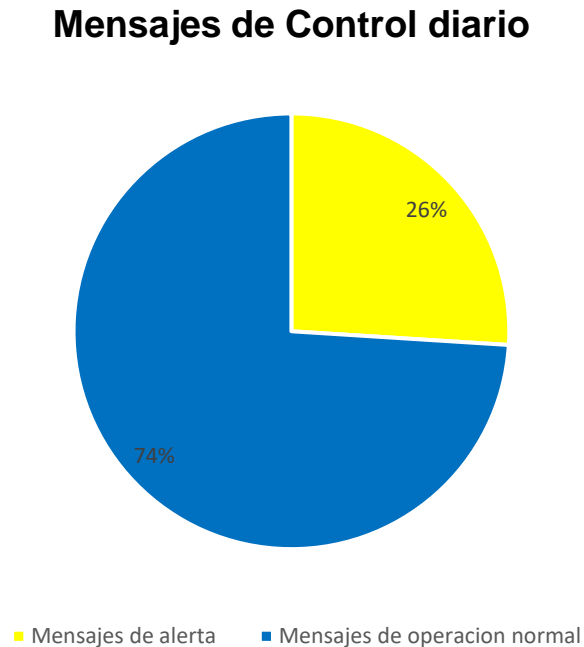
**Mensajes de ejemplo ajustados al sistema de etiquetas**

<b>Elemento</b>	<b>Mensaje</b>	<b>Etiqueta 0/1</b>
<b>0</b>	46 GLS. HR.21040 KM.429142. Bastante veriveri en la direccion, ya se paso para el cambio de filtros, revisar frenos, cuando se pasa de 5 a 6 chirrea muy feo.	0
<b>1</b>	ACPM 50 GLS HR 1225 - Grasa 2 Lbs.	1
<b>2</b>	HORA DE BRAZO GRUA HI.285-50 HF.285-50	1
<b>3</b>	Mezcla de base y cargue de volquetas.	1
<b>4</b>	GARZON-VIADUCTO.120	1
<b>5</b>		
<b>6</b>	52 gls. hr.1872. no le entra la gran primera, correas desajustadas.	0
<b>7</b>	Revisar a/c, revisar direccion tiene juego.	0
<b>8</b>	Llanta trasera en mal estado, lavado gral, licuadora no funciona.	0
<b>9</b>		
<b>10</b>	11 gls. hr.115583	1

La preparación de los datos culmina en la división por sets. De las filas que completan el informe de control diario resultan 96,785 espacios de entrenamiento, es decir, la sumatoria de todas las filas disponibles en Novedades más las filas de la columna Observación, de estas, la distribución ya etiquetada se encuentran 71.620 mensajes que informan de la operación normal, y 25.165 cuya sintaxis informa el contenido de mensajes que debieron ser notificados al área de mantenimiento. (ver figura 28)



Fig. 28

**Distribución de los mensajes de control diario****6.4.2. Pruebas de Tokenización y Aprendizaje por Transferencia**

En la preparación de la información se mencionó el origen de dos distintas fuentes de datos para el entrenamiento del modelo, el primero de ellos se trata de la información propia de la compañía cuyos datos serán el objetivo del algoritmo, y por el otro lado están los datos de terceros. Al importar modelos como BERT se obtiene una serie de complementos integrados como el propio tokenizador, el detalle está en que el idioma que nos confiere es el castellano, entonces, es conveniente elegir un modelo cuyo corpus lingüístico esté basado en el mismo idioma. La respuesta se encuentra en un artículo publicado por el departamento de ciencias de la computación de la Universidad de Chile

(CAÑETE y otros, 2020), se trata de un modelo pre-entrenado con bases en BERT en donde fueron usados recursos de distintas fuentes para su entrenamiento, entre los que se encuentran la Wikipedia, OPUS Project, Naciones unidas, diarios gubernamentales, Ted Talks, entre muchas fuentes de información disponible; lo atractivo del proyecto es que logra consolidar un extenso corpus logrando reunir más de 3 mil millones de palabras, un tamaño similar con el que fue alimentado BERT originalmente.

La tokenización es una etapa importante para adaptar la información en los Transformers, en base a esto el resultado puede variar considerablemente. De manera temprana se procedió a hacer varias pruebas para evaluar la efectividad en el proceso de tokenización que permite el modelo pre-entrenado. Al invocar la librería Transformer es posible importar directamente el autotokenizador, este no es necesario programarlo desde cero, por lo que rápidamente se pudo empezar experimentar con él.

**Fig. 29**

### ***Importación del modelo pre-entrenado***

```

1 from transformers import AutoTokenizer, AutoModelForMaskedLM
2
3 tokenizer = AutoTokenizer.from_pretrained("dccuchile/bert-base-spanish-wwm-uncased")
4
5 model = AutoModelForMaskedLM.from_pretrained("dccuchile/bert-base-spanish-wwm-uncased")

```

2m 4.3s

c:\Users\Felipe Tamayo\AppData\Local\Programs\Python\Python310\lib\site-packages\tqdm\auto.py:2: [https://ipywidgets.readthedocs.io/en/stable/user\\_install.html](https://ipywidgets.readthedocs.io/en/stable/user_install.html)  
 from .autonotebook import tqdm as notebook\_tqdm  
 Downloading: 76%|██████████| 319M/419M [01:42<02:02, 859kB/s]

Las primeras pruebas realizadas arrojaron una ejecución satisfactoria, este consigue extraer cada uno de los elementos que conforman el texto de entrada (los tokens), y su

4 [CLS]  
 4167 presenta  
 1700 mucho  
 11930 escape  
 1009 de  
 11069 humo  
 1040 y  
 11791 fuga  
 1009 de  
 8607 aceite  
 5516 motor  
 1040 y  
 1039 el  
 11904 vidrio  
 1149 esta  
 7151 chi  
 4462 ##tia  
 1050 ##do  
 5 [SEP]

representación numérica. Al tratarse de información que es digitada por conductores y operarios es natural encontrarse con palabras que el modelo nunca consideró, la palabra chitiado por ejemplo además de ser un elemento coloquial colombiano poco frecuente, tiene falta ortográfica, su forma de escritura aceptada es chiteado. El tokenizador al no poder relacionarlo con ninguno de los elementos con el que ha sido entrenado, aplica un método que es típico de los modelos basados en Bert, se trata de dividir el término en extractos más pequeños que ya reconoce.

chitiado	Chi ##tia ##do
----------	-------------------

Las primeras pruebas experimentales con los modelos lograron

tokenizar la mayoría de los términos efectivamente, como lo muestra la

figura de referencia. Además de los tokens reconocidos el modelo genera un tipo de token especial con los índices 4 y 5, por la documentación que ofrece el artículo BERT original (*DEVLIN y otros, 2018*) el primero de ellos es el token *CLS* se usa para modelos cuyo objetivo sea el reconocimiento de entidades como personas, ubicaciones geográficas fechas (*arcgis, s.f.*), diferente tarea de entrenamiento que se desea aplicar. Por su parte *SEP* indica la finalización de la frase, útil para trabajos que requieran modelos para generación de preguntas y respuestas.

El artículo publicado del Modelo pre-entrenado basado en BERT en español (*CAÑETE y otros, 2020*), nos enseña que han sido entrenados 2 modelos distintos, el *cased* y *uncased*, las diferencia está en que mientras el primero es capaz de procesar y diferenciar las palabras con mayúsculas y/o acentos, el segundo analiza las palabras en su versión más simple, sin considerar la puntuación.

<b>BERT-Cased</b>	<b>Habitación</b>
<b>BERT-Uncased</b>	Habitación -> habitacion

En vista de que no se requiere diferenciar frases que contengan mayúsculas como nombres propios ni acentos en las palabras, se ha utilizado el modelo BERT-*uncased* basado en español, sin embargo, se hicieron pruebas bajo ambas versiones del modelo.

Una de las virtudes del aprendizaje por transferencia es la capacidad de adaptación que se obtiene a partir de los datos de entrenamiento, la remodelación de los pesos para tareas distintas a las que ha sido diseñado originalmente. Puede que en el modelo original el objetivo fuese la traducción de texto, la detección de sentimientos, o incluso la generación de respuestas a preguntas, tareas muy diferentes a la que aquí compete, puesto que se desea hacer un tipo de clasificación de texto donde se obtiene comentarios que puedan inferir fallos en maquinaria y equipos, sin embargo, el contenido que extraemos del modelo Bert en español son los pesos y sesgos que arrojó el modelo, quien nos puede inferir las relaciones halladas en el lenguaje, las cabezas de atención que dio durante su entrenamiento, y los patrones que encontró en las relaciones entre palabras que tiene el lenguaje en español.

### **6.4.3. Preparación del entorno de entrenamiento**

Una vez se ha importado el modelo en el entorno de desarrollo local, se pueden hacer distintos tipos de pruebas. La primera estimación con la que se ha experimentado son los *logits*, que no son más que una probabilidad muy vaga que ofrece el modelo pre-entrenado respecto a una u otra etiqueta. Para dar un mejor contexto, cuando ya se

tienen los datos cargados, se instancian argumentos de la librería que permiten directamente operar con la cantidad de elementos que se quieren calcular. Debido a que los requerimientos solo necesitan diferenciar entre los 2 tipos de mensajes que encontramos en control diario [0 / 1], este es el valor que se definió, similar a como lo muestra figura 30. Si el área de mantenimiento hubiese elegido los 5 tipos de etiquetas para caracterizar los distintos tipos de fallos planteados en el ejercicio, este valor habría sido reemplazado.

**Fig. 30**

#### **Definición del número de etiquetas.**

```
from transformers import AutoModelForSequenceClassification  
  
num_etiquetas = 2  
modelo = AutoModelForSequenceClassification.from_pretrained(model_checkpoint, num_labels=num_etiquetas)
```

Una vez establecido el valor, y sin ningún entrenamiento previo podemos forzar el modelo a una estimación temprana, devolviendo dos valores que nos dice el potencial de cercanía con cada una de las etiquetas. Tomando muestra de uno de los mensajes que encontramos, podríamos generar unos *Logits*, un par de valores que se usan típicamente en redes neuronales para calcular las probabilidades que tiene una salida de pertenecer a uno u otro conjunto de datos, como se aprecia a continuación:

**“Mezcla de base y cargue de volquetas.”**

***Logits* : [0.54652, 0.45347]**

Esta estimación se basa en los pesos y material de entrenamiento del modelo original, pero no tienen absolutamente ningún contexto de lo que aquí se requiere, no sabe de las características que tienen los mensajes marcados con la etiqueta 0, ni la diferencia con las que incorporan el valor 1, se hace necesario entonces hacer un ajuste fino.

El ajuste fino es una manera de adaptar los datos que ofrece el modelo original a un caso particular, se trata de tomar la información que contiene los pesos calculados por el modelo Bert en español, y adaptarla para un nuevo caso de estudio, habituarla al contenido de los mensajes que pueden hallarse en control diario, con lo que se logra reajustar los pesos y sesgos con la finalidad de obtener una salida más acertada, es el principio fundamental del aprendizaje por transferencia.

#### **6.4.4. Establecimiento de hiperparámetros de entrenamiento.**

Transformers exige una serie de argumentos que modulan la manera en la que se hace el entrenamiento:

- a. Los tamaños de los lotes
- b. La cantidad de épocas de entrenamiento
- c. Longitud máxima de las oraciones
- d. La métrica de evaluación

Es un requisito obligatorio del que se debe estar familiarizado, de otra manera no se tendrá éxito en la implementación.

#### **6.4.4.1. Los tamaños de los lotes**

La arquitectura Transformers recibe el conjunto de datos en lotes divididos en partes iguales para analizarlos de manera iterativa, típicamente en valores como 2, 4, 8, 16, 32, 64, 128. En cada una de estas iteraciones se estima el error para cada ejemplo en el conjunto de datos, esta toma muestra del error y al finalizar entrenamiento actualiza el conglomerado del error detectado. Documentación de investigaciones previas en el tema recomiendan optar por tamaños estándar de 32 lotes (*Bengio, 2012*), donde un valor pequeño garantiza un entrenamiento rápido, en contraste, uno alto estimará con mayor precisión con el limitante que entre mayor sea el tamaño del lote, mayor será el uso de recursos de CPU o GPU, entonces, este valor es un número que estaría equilibrado para la mayoría de las tareas.

#### **6.4.4.2. Épocas de entrenamiento**

La época (*Epoch*) es la cantidad de veces que la red neuronal repasa todo el conjunto de datos de entrenamiento, cada época es una oportunidad de actualización de los pesos y sesgos del modelo, en otras palabras, es un ciclo completo de ejecución que habilita el aprendizaje; existe un problema relacionado con la cantidad de épocas, si bien un mayor número de éstas ayuda al modelo a disminuir el error de predicción, excederse puede generar un sobreajuste (*Variš & Bojar, 2021*), esto es, que el modelo puede aprender y hacer las predicciones de manera muy acertada especializándose en la información de entrenamiento, pero fallar estrepitosamente al analizar nuevos datos.

#### **6.4.4.3. Longitud máxima de las oraciones**

Para hallar la longitud del texto de entrenamiento se usaron métodos de barrido aplicados a la tabla depurada de control diario, se contabilizaron los elementos y se

definió la cantidad máxima de palabras del que estaban constituidos habitualmente. Estadísticamente se extrajeron dos valores importantes para definir la longitud del texto, la moda (valor que aparece con mayor frecuencia en el conjunto de datos), los máximos, y la relación que tiene los máximos con respecto a todo el conjunto de datos.

Se halló que la moda de la longitud de los elementos se encontraba en 30 tokens, los mensajes más extensos contienen un máximo de 38 palabras, y estos representan un 18.2%, es decir 17.615 mensajes de control diario, un valor que no es despreciable. De este modo por conveniencia y para que todo el set de datos pueda ser analizada pre y post entrenamiento el valor de este hiperparámetro se estableció en  $L=40$ . A diferencia de los parámetros anteriores que están expuestos a cambios, este se ha fijado puesto que no presenta mayores cambios.

#### 6.4.4.4. Métrica de evaluación

Cuando se trata de evaluar el desempeño de un modelo de IA hay una serie de métricas que analizan desde distintas perspectivas la efectividad del pronóstico que es posible lograr, para avanzar en la prueba se encontró la exactitud como métrica directa para ponderar los resultados del modelo, es efectiva, y sencilla de interpretar, y facilita la lectura por parte de los analistas de información. La métrica “*accuracy*” es la relación del número de predicciones correctas sobre el número total de predicciones.

$$Exactitud = \frac{PIA + PAA}{PIA + PAA + PID + PAD} \quad (6)$$



Donde,

- PIA - predicciones informativas acertadas
- PAA – predicciones de alertas acertadas
- PID – predicciones informativas desacertadas
- PAD – predicciones de alertas desacertadas

La desventaja que presenta esta medida se ve reflejada en la propia fórmula.

Cuando se trata de clasificación de texto los conjuntos de datos deben tener cierto balance, para obtener los mejores resultados, para comprender este fenómeno, considérese el siguiente ejemplo:

Suponiendo que se tiene un balance desproporcionado de los datos de entrenamiento donde existen 1000 mensajes que informan operación normal, y únicamente 10 de alerta, el PIA y el PAA posible tendrían una relación 1000 a 10 respectivamente.

En un caso hipotético, donde el modelo es muy efectivo detectando las predicciones informativas PIA alcanzando 990/1000, pero que falla estrepitosamente con las predicciones de alertas acertadas PAA logrando detectar únicamente 1/10, si se reemplazan estos datos en la fórmula tenemos que la exactitud nos da un valor de 0.98, donde exactitud=1 sería un modelo 100% efectivo. Por lo que el resultado que nos arroja es un valor muy alto que podría tomarse como una predicción casi perfecta. Sin embargo, debido a la mínima representación que tienen los mensajes de alertas, las predicciones que hagan en base a este conjunto de datos apenas modificarán el valor de exactitud, el

resultado se verá afectado exclusivamente cuando el conjunto de datos más numeroso disminuya en la cantidad de predicciones acertadas.

Pese a que en el ejemplo ha sido considerada una situación poco frecuente, un modelo que tenga datos desbalanceados en menor medida puede también verse afectado por este fenómeno, la diferencia es que el resultado no será tan intuitivo. Distintos artículos han plasmado este problema, como se hace en el documento “El impacto del desequilibrio de clases en las métricas de rendimiento de clasificación basadas en la matriz de confusión binaria”. (*Escuela Politécnica Superior. Universidad de Sevilla., 2019*), aun teniendo este obstáculo, se continuó con el uso de la métrica exactitud por la misma razón que fue expuesta anteriormente, es sencilla de interpretar y puede llegar a ser muy efectiva; por esta razón se debió agregar un paso extra al preprocesamiento de los datos, el balanceo de las etiquetas.

#### **6.4.5. Reajuste de los datos de procesamiento**

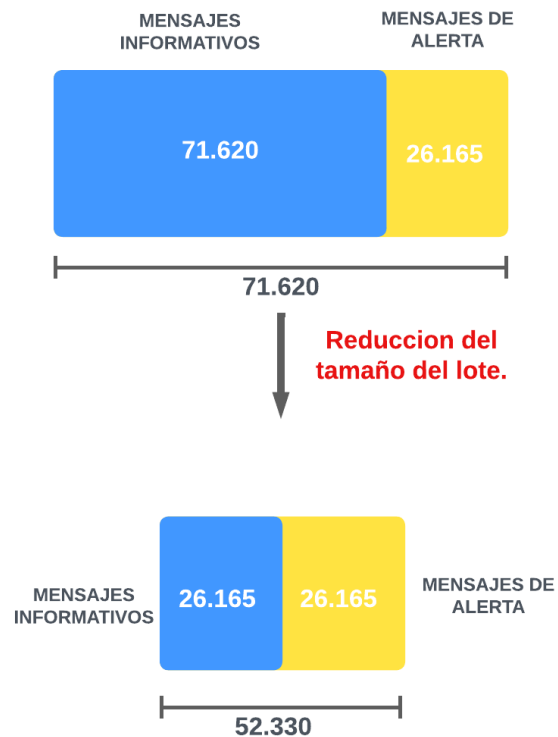
En el tamaño de los datos de entrenamiento anteriormente resultaron 97.785 frases, donde los mensajes que requieren atención llegan a los 26.165, mientras que los mensajes que no son de competencia a mantenimiento son 71.620, entonces, estos últimos debieron ser reducidos. Para alcanzar el balance se tendrían que descartar 45.455 mensajes ya clasificados.

En el set de datos se aplicaron una serie de filtros donde se extrajeron mensajes que podrían coincidir mucho en estructura con otras representaciones de la misma categoría, por ejemplo, donde se informan acontecimientos que pueden variar en una o dos palabras, esta aplicación, junto con la toma de muestra aleatoria, logró equiparar la

cantidad de mensajes informativos, con la cantidad de mensajes de alerta, como lo muestra la figura 31.

**Fig. 31**

**Balanceo de datos de alerta e informativos.**



#### 6.4.6. Entrenamiento

Uno de los primeros hiperparámetros experimentados se ven reflejados en el extracto de código mostrado en la figura 32.

Fig. 32

### Establecimiento de hiperparámetros en el algoritmo

```

from transformers import TrainingArguments

lote = 32
epoca_entr=8
logging_steps = len(train_dataset) // (2 * lote * epoca_entr)
split = 0.9
size = int((num_samples/batch_size)* split)

optimizador = tf.keras.optimizers.Adam(lr=1e-5, decay=1e-6)
loss = tf.keras.losses.CategoricalCrossentropy()
metrica = tf.keras.metrics.CategoricalAccuracy('accuracy')

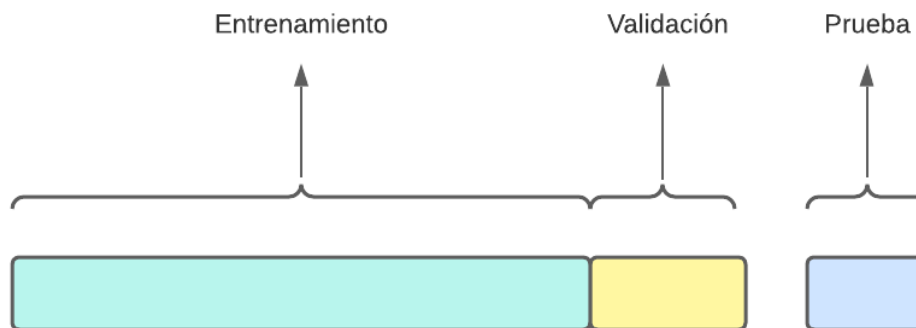
training_args = TrainingArguments(
    output_dir="results", num_train_epochs=epoca_entr, optimizador, opper_device_eval_batch_size=batch_size, load_best_model_at_end=True,
    metrica, weight_decay=0.01, evaluation_strategy="epoch", save_strategy="epoch", logging_steps=logging_steps,
)

```

Para el entrenamiento el modelo usa 3 sets de datos a partir del compilado de control diario, los propios datos de entrenamiento, datos de validación, y datos de prueba como se muestran en la figura 33, cada uno de ellos con mensajes distintos. Los datos de entrenamiento le indican al modelo la sintaxis y muestreo de los tokens que se repiten con frecuencia, sirven para ajustar los datos que se usaron de la transferencia de conocimiento del modelo Bert en español al entorno de reconocimiento de frases de fallo; por otro lado los datos de validación se usan para evaluar el modelo de manera parcial mientras se hace un entrenamiento, mientras que los datos de prueba se utilizan para calcular la precisión una vez que el modelo ha terminado de entrenarse, este último valor es la precisión que tenemos presente al finalizar el entrenamiento.

Fig. 33

### Distribución segmentos de entrenamiento



El entorno de desarrollo que se usó hasta este punto para la codificación del algoritmo difiere al entorno de entrenamiento, el motivo es la limitación en los recursos de hardware disponibles, se vio entonces en la necesidad de buscar hardware con mayor poder de procesamiento donde se consideró usar la herramienta Google colab (*GOOGLE, s.f.*), un producto diseñado por Google Research habilitado para el desarrollo en línea de código Python, cuenta con recursos gratuitos de CPU y la GPU para el entrenamiento de modelos de inteligencia artificial, no obstante, es un servicio externo basado en la nube en el que la privacidad de los datos no está asegurada, y puede verse expuesta información sensible. Por esta razón se buscaron alternativas, siempre teniendo prioridad la elección de recursos con GPU. Se optó por el uso de una serie de 6 computadores cuya unidad de procesamiento gráfico es la Nvidia GTX 2080 TI, donde se distribuyeron los sets de entrenamiento en 6 copias, estos entrenamientos fueron ejecutados paralelamente para obtener resultados en el menor tiempo, con ello daría tiempo de ajustar los hiperparámetros introducidos de ser necesario.

Los diferentes equipos tienen definidos hiperparámetros que varían entre sí, el objetivo es calibrar el algoritmo para destacar las salidas que retornen el mejor rendimiento. Debe aclararse que esta serie de entrenamiento no fue la única realizada, los hiperparámetros debieron variar multitud de veces para ajustar el modelo y que este nos arrojará valores más precisos, estas salidas pueden consultarse en la sección de Resultados.

**Tabla 14**

**Hiperparámetros definidos**

	Tamaño de los lotes.	Épocas de entrenamiento	Longitud de las oraciones	Dimensiones del Word embedding
<b>Primer entrenamiento</b>				
<b>Equipo 1</b>	16	3	40	512
<b>Equipo 2</b>	32	3	40	512
<b>Equipo 3</b>	64	3	40	512
<b>Equipo 4</b>	16	8	40	512
<b>Equipo 5</b>	32	8	40	512
<b>Equipo 6</b>	64	8	40	512
<b>Segundo entrenamiento</b>				
<b>Equipo 1</b>	16	3	40	128
<b>Equipo 2</b>	32	3	40	128
<b>Equipo 3</b>	64	3	40	128
<b>Equipo 4</b>	16	8	40	128
<b>Equipo 5</b>	32	8	40	128
<b>Equipo 6</b>	64	8	40	128
<b>Tercer entrenamiento</b>				
<b>Equipo 1</b>	32	8	40	128
<b>Entrenamiento final</b>				
<b>Equipo 1</b>	32	8	40	64
<b>Otro entrenamiento</b>				
<b>Equipo 1</b>	32	8	40	32

En la tabla 14 se visualizan los entrenamientos más concluyentes que se ejecutaron. En el primer entrenamiento colectivo se variaron los tamaños de los lotes en valores de 16,32,64 respecto a las épocas de entrenamiento que tuvieron valores de 3 y 8, sin variación de las dimensiones del Word embedding. Los resultados que se obtenían hicieron que se cambiaran las épocas esta vez a 4 y 12, con la misma secuencia de lotes 16,32 y 64. Inicialmente la dimensión del *embedding* se establecería como un valor fijo igual al que se aplicó en el modelo Transformers original, es decir 512, sin embargo, en búsqueda de disminución de tiempos de entrenamiento, estos se redujeron a partir del segundo entrenamiento a 128, el cambio se debió a que hay una relación directa en el tamaño del Word embedding y tiempo de entrenamiento, al final de la tabla se pueden ver los resultados más efectivos que se pudieron ajustar sin sacrificar la precisión. La forma como se visualiza el entrenamiento se puede ver en la figura 34.

**Fig. 34**

### Ventana de visualización de la salida del modelo

```

history = model.fit(
    train_ds,
    validation_data=val_ds,
    epochs=8
)

Epoch 1/8
62/62 [=====]
Epoch 2/8
62/62 [=====]
Epoch 3/8
62/62 [=====]
Epoch 4/8
62/62 [=====]
Epoch 5/8
62/62 [=====]
Epoch 6/8
62/62 [=====]
Epoch 7/8
62/62 [=====]
Epoch 8/8
62/62 [=====]

```

### 6.4.7. Evaluación del modelo

A medida que las etapas de entrenamiento finalizaban no solo se tomaban como valor de referencia la métrica de salida “precisión”; en cambio, también se evaluaba directamente con mensajes que se iban digitando en la plataforma, este detalle es importante, puesto que el modelo no estaba contextualizado de la nueva información que se iba generando. Con el modelo ajustado no solo se obtiene la clasificación de los mensajes, además se puede visualizar las cabezas de atención que se presentaron en distintas capas.

En la figura 35 se muestra cómo se evaluó el comentario “El aire acondicionado viene presentando fallos”, donde fue seleccionada la palabra **presentando**, y dos distintas capas de atención. Rápidamente podemos notar a que tokens le presto mayor atención por la intensidad de color, incluso en la figura de la izquierda se relacionó así misma en gran medida.

**Fig. 35**

#### ***Múltiples cabezas de atención.***





Evidentemente el parámetro de salida que más nos interesa es la predicción que logró, recordando que cuando se trata de mensajes con la etiqueta 0 corresponden a alertas de fallo para mantenimiento, mientras que la etiqueta 1 son los mensajes de operación no concerniente al departamento.

La primera parte de la salida (**Fig. 36**) nos muestra el tiempo que tardó en analizar el mensaje, este valor también está relacionado con el tamaño del *embedding*, cuanto mayor sea el tamaño del embedding mayor será el tiempo de la predicción.

**Fig. 36**

#### Extracto de código de la predicción del modelo.

```

1 val = "El aire acondicionado viene presentando fallos"
2 salida = np.argmax(model.predict(prepare_data(val))[0])
3 if salida == 0:
4     print("", val , "", "El equipo esta reportando fallos, notificar a mantenimiento!")
5 else:
6     print("El mensaje no contiene informacion reelevante para mantenimiento.")

```

✓ 0.3s

1/1 [=====] - 0s 228ms/step

' El aire acondicionado viene presentando fallos ', El equipo esta reportando fallos, notificar a mantenimiento!

Para tener una concepción más amplia del porcentaje de precisión que arroja cada una de las salidas ya entrenadas, se utilizó un dataset donde se reúnen los mensajes de una semana aproximadamente. A este no se le aplica filtro alguno, sencillamente se le entrega el documento y se le indican los campos a recorrer: novedades y observaciones; recursivamente realiza las predicciones de cada uno de los mensajes y los etiqueta en el mismo documento, al final obtiene el archivo con el cual los analistas de información corroboran la cantidad de mensajes correctamente clasificados; estas pruebas varían

dependiendo de los entrenamientos, en lo que siempre se coincide es en el porcentaje de precisión arrojado y el porcentaje de mensajes acertados que fueron revisados manualmente.

## **6.5. Despliegue**

### **6.5.1. Integración a la plataforma Maseq**

Una vez el modelo arroja resultados lo suficientemente estables para automatizar la extracción de la información se concedió el acceso a las bases de datos. Para consultar la información de control diario fue proporcionado un archivo tipo JSON (*JavaScript Object Notation*), este es un formato para el intercambio de datos que utiliza una estructura similar a un enlace web, donde adicionalmente se envía una cabecera, un parámetro que contiene un token de acceso especial, a través de él se puede acceder por completo a los reportes de control diario, con la seguridad de que no es posible alterar los datos almacenados; en bases de datos este está habilitado exclusivamente para hacer consultas y no inserción de información.

Conectar con el modelo con el JSON no resultó ser mayor reto, pandas tiene una instrucción dedicada para ello, por lo que la importación se hizo directamente con la librería, accediendo así, al control diario en tiempo real. De los requerimientos, la disponibilidad de la aplicación se exigió que fuese de tipo ejecutable por periodos, en este método se acumulan los mensajes de control diario que reportan en toda la jornada laboral, y al día siguiente se ejecuta el script en un horario establecido. Se determinó además que de encontrarse reportes de fallos en equipos se notifica a los encargados, al

contrario, sino se detectan reportes de control diario con fallos igualmente se reporta un mensaje que indica normalidad, de manera que los encargados se aseguran de que la tarea del algoritmo se ejecuta diariamente.

### **6.5.2. Notificación de las alertas de fallos al equipo de mantenimiento**

El medio de transmisión que se usó finalmente para notificar los mensajes de fallos en los equipos elegido es WhatsApp, la compañía cuenta con un entramado de grupos soportado en este servicio de mensajería por cada área y proyecto de la compañía, en consecuencia, el coordinador del área de minería de datos e innovación creó el grupo donde fueron involucradas los responsables de alertar los fallos en los equipos. Entonces se precisó la conexión del script con el api de WhatsApp. Directamente WhatsApp no provee conexión directa con los usuarios de su plataforma, este lo hace vía servicios pagos de terceros, entre ellos los más populares Twilio, Vonage o Podium, no obstante, a la fecha de implementación ninguno de ellos cuenta con el servicio de notificación a grupos, no por omisión suya, sino por parte de WhatsApp quien dejó de dar soporte a la transmisión de información a chats grupales desde el año 2020. La alternativa posible es el uso de una librería que utilice la interfaz gráfica de WhatsApp web, valiéndose de una librería para tal fin: pyautogui (figura 37), con ella el script abre el navegador y se dirige a la dirección web del grupo, una vez que esté cargada la interfaz, pega los mensajes que encontró respecto a los fallos en los equipos.

Fig. 37

### Envío de notificaciones vía WhatsApp Web.

```

344 web.open('https://web.whatsapp.com/send?g=grupobot')
345
346 time.sleep(5)
347 width,height = pg.size()
348 pg.click(width/2+width/4,height/2)
349 pg.press('enter')
350 pg.write('*REPORTE DE FALLOS DE EQUIPOS*')
351 pg.press('enter')
352 for c in imprime.index:
353     pg.write('*'+unicode.unidecode(imprime['vehiculo'])[c]+'* ' + unicode.unidecode(imprime['mensaje'])[c])
354     pg.press('enter')
355 time.sleep(10)
356 pg.hotkey('ctrl','w')

```

Es un proceso mecánico para la notificación de los mensajes, pero es funcional y la única alternativa para enviar mensajes grupales, este estaría activo dentro de la compañía hasta que se rehabilite la funcionalidad por parte del servicio de mensajería.

#### 6.5.3. Ejecución programada

La auto ejecución del script está soportada en un servidor local, este mantiene ciertos servicios activos 24/7 de manera que puede asistir la ejecución de la tarea ininterrumpidamente, a tal efecto se usó una de las virtualizaciones disponibles para la ejecución de scripts, esta virtualización contiene un sistema operativo Windows 10 donde se limitaron la mayoría de los servicios. Allí se debió crear un entorno virtual que permitiera la instalación de las librerías requeridas para la ejecución del script, evitando por consiguiente la incompatibilidad con otras librerías que usan desarrollos de terceros.

Windows tiene un servicio por defecto que se llama el programador de tareas (**Fig. 38**), donde es posible fijar tareas en tiempos predefinidos, por lo que se recurrió a él para desencadenar el servicio de detección.

**Fig. 38**

**programador de tareas en virtualización de Windows.**

The image shows the 'Editar desencadenador' (Edit Trigger) dialog box in Windows Task Scheduler. The window title is 'Editar desencadenador' with a close button (X) in the top right corner. The 'Iniciar la tarea:' (Start the task:) dropdown is set to 'Según una programación' (By a schedule). The 'Configuración' (Configuration) section has four radio buttons: 'Una vez' (Once), 'Diariamente' (Daily), 'Semanalmente' (Weekly), and 'Mensualmente' (Monthly). The 'Semanalmente' option is selected. The 'Inicio:' (Start:) field is set to '9/08/2022' and '7:00:00 a. m.', with a 'Sincronizar zonas' (Synchronize zones) checkbox. The 'Repetir cada:' (Repeat every:) field is set to '1' and 'semanas en:' (weeks in:). The days of the week are listed with checkboxes: 'Domingo' (unselected), 'Lunes' (selected), 'Martes' (selected), 'Miércoles' (selected), 'Jueves' (selected), 'Viernes' (selected), and 'Sábado' (selected). The 'Configuración avanzada' (Advanced configuration) section has several options: 'Retraso máx. (retraso aleatorio):' (Maximum delay (random delay):) set to '1 hora' (1 hour); 'Repetir cada:' (Repeat every:) set to '1 hora' (1 hour) and 'durante:' (for:) set to '1 día' (1 day); a checkbox 'Detener todas las tareas en ejecución al final de la duración de repetición' (Stop all tasks running at the end of the repetition duration) is unchecked; a checked checkbox 'Detener la tarea si se ejecuta durante más de:' (Stop the task if it runs for more than:) set to '30 minutos' (30 minutes); 'Expiración:' (Expiration:) set to '13/11/2023' and '8:50:27 p. m.', with a 'Sincronizar zonas horaria' (Synchronize time zone) checkbox; and a 'Habilitado' (Enabled) checkbox which is unchecked. At the bottom right, there are 'Aceptar' (OK) and 'Cancelar' (Cancel) buttons.

## 7. Resultados

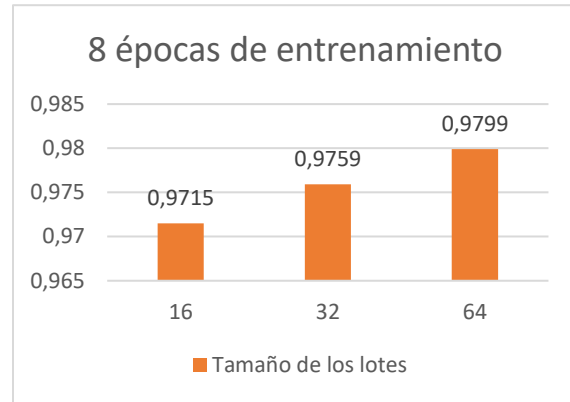
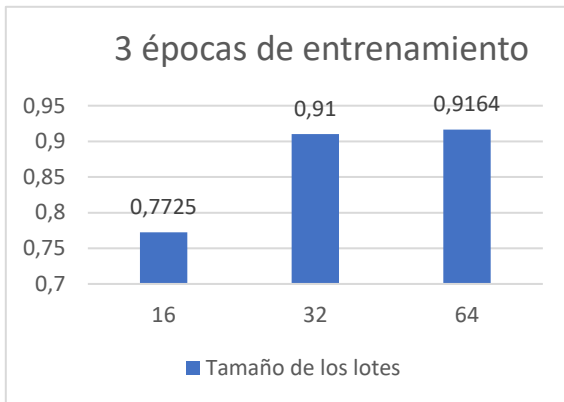
Los hiperparámetros usados para los datos de entrenamiento variaron conforme la métrica de precisión arrojó resultados favorables (Tabla 15), entonces, se replicaba el patrón variando otro hiperparámetros, con el objetivo de identificar el cambio que tiene mayor influencia en la precisión.

**Tabla 15**

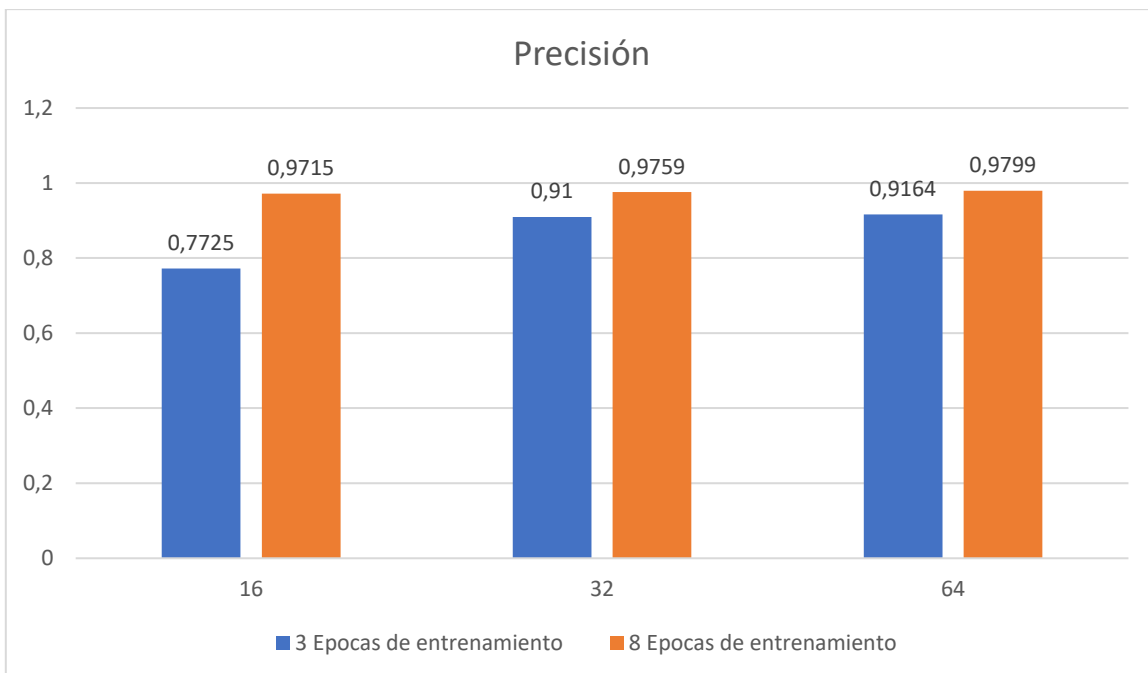
**Resultados del primer entrenamiento colectivo.**

	Tamaño de los lotes.	Épocas de entrenamiento	Dimensiones del Word embedding	Salida de la métrica Precisión
<b>Equipo 1</b>	16	3	512	0,7725
<b>Equipo 2</b>	32	3	512	0,91
<b>Equipo 3</b>	64	3	512	0,9164
<b>Equipo 4</b>	16	8	512	0,9715
<b>Equipo 5</b>	32	8	512	0,9759
<b>Equipo 6</b>	64	8	512	0,9799

Para el primer escenario se compararon las épocas de entrenamiento respecto al tamaño de los lotes, donde se encontró que la intuición es correcta al decir que entre mayor sea el tamaño de los lotes mayor será la precisión para los dos valores de épocas de entrenamiento 3 y 8.



De este también se concluye que existe una diferencia considerable entre la precisión calculada para el conjunto de datos con 3 épocas, respecto al conjunto de 8 épocas, sin embargo, la diferencia no se ve tan marcada entre los distintos tamaños de los lotes, teniendo presente que estos tienen la mitad del tamaño respecto a su siguiente valor,  $16 \cdot 2 = 32$ ,  $32 \cdot 2 = 64$ .



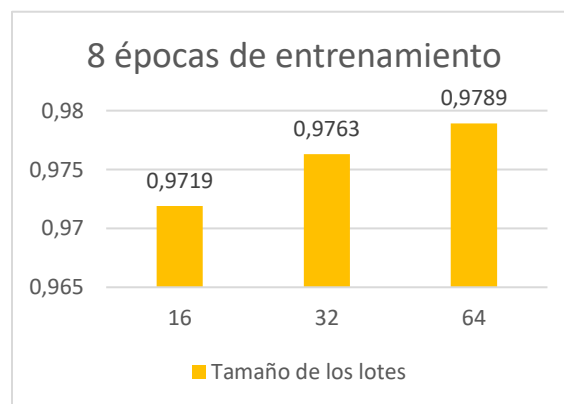
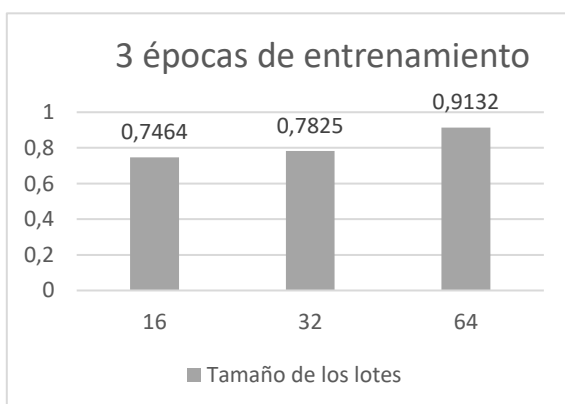
En el segundo entrenamiento colectivo se cambió la dimensión del Word embedding a  $\frac{1}{4}$  del valor original (**Tabla 16**), manteniendo los mismos tamaños de lotes y las épocas de entrenamiento, la razón, menores tamaños del *Word Embedding* inciden en menores tiempos de entrenamiento, así como menores tiempos clasificación.

**Tabla 16**

**Resultados del segundo entrenamiento colectivo pasando de 512D a 128D.**

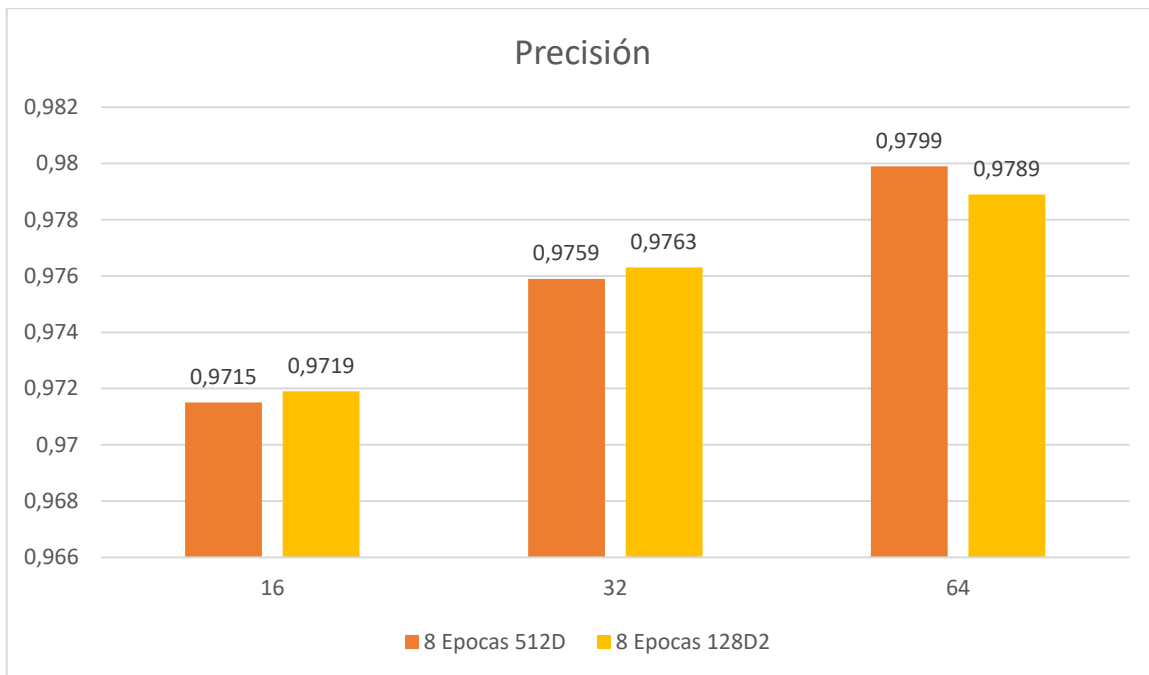
	Tamaño de los lotes.	Épocas de entrenamiento	Dimensiones del Word embedding	Salida de la métrica Precisión
<b>Equipo 1</b>	16	3	128	0,7464
<b>Equipo 2</b>	32	3	128	0,7825
<b>Equipo 3</b>	64	3	128	0,9132
<b>Equipo 4</b>	16	8	128	0,9719
<b>Equipo 5</b>	32	8	128	0,9763
<b>Equipo 6</b>	64	8	128	0,9789

Los datos una vez más confirman la premisa anterior, al menos para el set de datos ajustado a control diario, tiene una mayor influencia la cantidad de épocas de entrenamiento que el tamaño de los lotes en sí.





Si se traslada la comparativa para los conjuntos de entrenamiento con menor dimensión del Word embedding, se ve reflejado que disminuir de 512D a 128D apenas afecta la precisión; al contrario de lo que podría pensarse incluso podemos notar que a 16 y 32 lotes reducir el Word embedding generó un resultado levemente superior.



Se pudieron establecer unos hiperparámetros de tamaño estándar que presentaron un buen resultado, la configuración donde el tamaño de lote es 32, junto con 8 épocas de entrenamiento. Con estos valores se continuó reduciendo la dimensión del Word embedding, puesto que se aprecia que no conlleva mayor efecto en las anteriores pruebas, sin embargo, esta tiene un limitante a mínimo 64D, menor a ese valor la precisión empieza a perder efectividad significativamente, obteniéndose el peor resultado de la tabla 17.

Tabla 17

## Compilado de resultados

	Tamaño de los lotes.	Épocas de entrenamiento	Dimensiones del Word embedding	Salida de la métrica Precisión
Equipo 1	16	3	512	0,7725
Equipo 2	32	3	512	0,91
Equipo 3	64	3	512	0,9164
Equipo 4	16	8	512	0,9715
Equipo 5	32	8	512	0,9759
Equipo 6	64	8	512	0,9799
Equipo 1	16	3	128	0,7464
Equipo 2	32	3	128	0,7825
Equipo 3	64	3	128	0,9132
Equipo 4	16	8	128	0,9719
Equipo 5	32	8	128	0,9763
Equipo 6	64	8	128	0,9789
Equipo 1	32	8	128	0,9749
Equipo 1	32	8	64	0,9732
Equipo 1	32	8	32	0,7133

Por otro lado, es necesario presentar los tiempos que resultaron de cada entrenamiento, si se ordena la tabla por tiempos de ejecución se aprecia la relación directa que existe entre la dimensión del Word embedding y los tiempos de entrenamiento.

Tabla 18

Comparativa de hiperparámetros y tiempos de entrenamiento.

	Tamaño de los lotes.	Épocas de entrenamiento	Dimensiones del Word embedding	Salida de la métrica Precisión	Tiempo de entrenamiento (minutos)
Equipo 1	32	8	32	0,7133	92
Equipo 1	32	8	64	0,9732	189
Equipo 5	32	8	128	0,9763	312
Equipo 2	32	3	128	0,7825	317
Equipo 1	16	3	128	0,7464	321
Equipo 4	16	8	128	0,9719	329
Equipo 1	32	8	128	0,9749	337
Equipo 6	64	8	128	0,9789	340
Equipo 3	64	3	128	0,9132	361
Equipo 1	16	3	512	0,7725	518
Equipo 5	32	8	512	0,9759	518
Equipo 3	64	3	512	0,9164	522
Equipo 4	16	8	512	0,9715	529
Equipo 6	64	8	512	0,9799	538
Equipo 2	32	3	512	0,91	578

Esta última tabla pone en evidencia el mejor de los resultados que se obtuvo basado en dos características principalmente, un puntaje alto, y un tiempo de entrenamiento y por consiguiente un valor de clasificación bajo. Los hiperparámetros que consiguen cumplir con ambas premisas es mostrado en la tabla 19.

Tabla 19

Hiperparámetro seleccionado para despliegue del aplicativo.

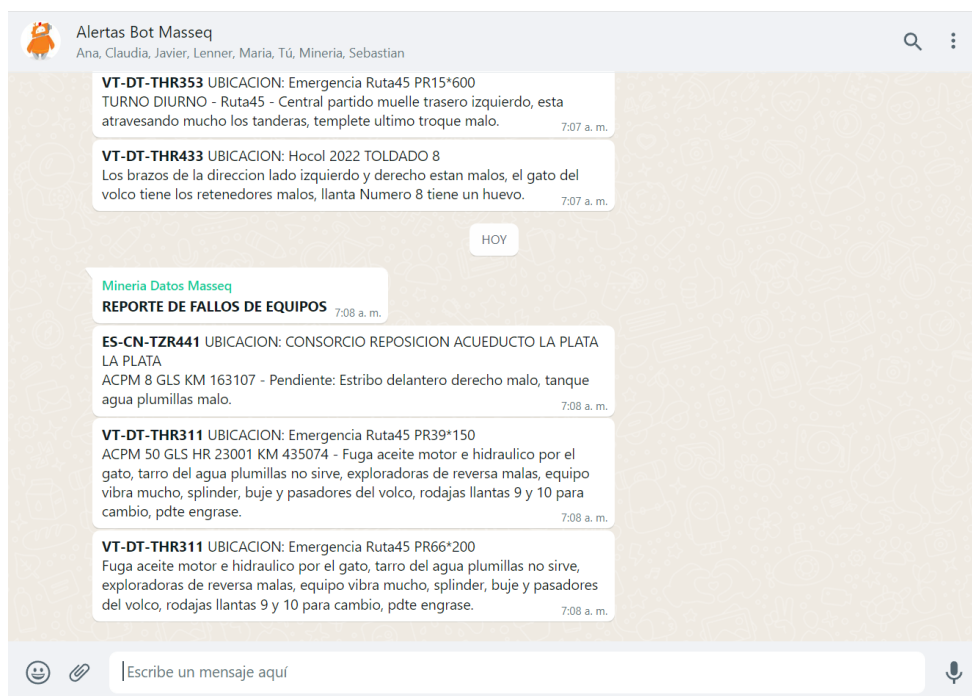
	Tamaño de los lotes.	Épocas de entrenamiento	Dimensiones del Word embedding	Salida de la métrica Precisión	Tiempo de entrenamiento (minutos)
Equipo 1	32	8	64	0,9732	189

Este nos arroja una precisión de 0.9732, un valor que está muy por encima del promedio para el conjunto de entrenamientos, y fue entrenado en solamente 189 minutos, en el segundo menor tiempo de entre todos los entrenamientos.

Las herramientas más populares no siempre son las que mejor soporte tienen, cuando se habla la red de mensajería WhatsApp se espera un sistema robusto y que cuente con la mayoría de las herramientas que exige el mercado, en cambio se encontró con restricciones que impiden la automatización de los mensajes, a diferencia de otros servicios menos populares como Telegram, o Signal. Aún es posible automatizar el envío de mensajes a grupos de WhatsApp utilizando otras herramientas, la librería pyautogui para tal efecto. El resultado de la notificación enviada se puede ver en la figura 39.

**Fig. 39**

### Mensajes notificados por el Script en el grupo de WhatsApp.



## 8. Conclusiones

Reconocer las expresiones de fallos en los equipos no es un proceso inherente de detectar palabras que estén relacionadas con términos como fallo, error, problema, etc. si se entendiera el problema de esta manera, el porcentaje de efectividad del algoritmo sería muy regular. En el ambiente constructor a menudo se encontró el término “atendiendo fallo en la vía”, una referencia a un fallo geológico, y no un fallo en el equipo en sí, además de este, existen multitud de ejemplos con expresiones que podrían confundir incluso a personas. La versatilidad de los Transformers no solo permitió capturar el significado de las palabras, sino a partir de un diseño adaptativo reconocer las expresiones de fallo que caracterizan los reportes de mantenimiento.

Asignar mayor o menor relevancia a los fallos reportados es una actividad interna del área de mantenimiento, la parametrización que se exigió se cumplió extrayendo las categorías de los vehículos halladas en el control diario, a estas se les da mayor transcendencia en función del tiempo que transcurra entre el reporte y la recepción del activo por parte del taller mecánico.

Uno de los aspectos más comunes encontrados dentro de los mensajes de control diario está relacionado con la gran cantidad de jerga regional que se usa, palabras como chitiado, veriveri, cardaneando, puede confundir a un hablante nativo del español que no esté familiarizado con estas expresiones, sin embargo, gracias al aprendizaje no supervisado se pueden detectar muchos patrones permitiendo clasificar en una u otra categoría el tipo de mensaje al que hacen referencia.

Gracias a la popularidad de los formatos de intercambio de información como el JSON utilizado para la extracción de los mensajes de control diario, fue posible recolectar la información desde las bases de datos de la compañía. Estos formatos garantizaron el acceso a los servicios web, y permitieron trasladar el algoritmo a un entorno distinto al de desarrollo, uno de los servidores empresariales, posibilitando la automatización en la extracción de los reportes de fallos en los equipos, esto sin la intervención humana, más allá de la que se requirió al hacer la primera configuración del servicio de notificaciones.

En cuanto a las notificaciones en tiempo real, el periodo que el equipo de mantenimiento considera oportuno es una jornada laboral completa; una notificación reportada inmediatamente después de registrarse en el aplicativo web no solo inútil, sino que además un desperdicio de recursos. Las circunstancias ideales, establecieron que el tiempo donde se pueden manejar las alertas es al principio de la jornada laboral, por lo que la recurrencia de las notificaciones está fijada, al menos durante el periodo de la pasantía, a primera hora del día ocupacional.

## 9. Trabajos Futuros

Uno de los interrogantes que se dieron en la formulación de algoritmo fue si un modelo de inteligencia artificial podría ser capaz de extraer información a partir de texto careciente de estructura. Mas tarde comprobando las capacidades que ofrecen las redes transformes, se abre camino para la creación de una variedad de aplicaciones donde el lenguaje puede contener representaciones de vocabulario informal, por ejemplo, categorizando la información que es consultada a través de los mensajes enviados vía WhatsApp del servicio al cliente, esta información es escasamente clasificada, sin embargo, puede contener información que retroalimente la calidad del servicio.

Si bien el modelo fue diseñado para el reconocimiento y clasificación entre dos tipos de oraciones, la estructuración de este puede adaptarse para agrupar por categorías no solo la información de los fallos encontrados en vehículos, sino también el reconocimiento de distintas expresiones que necesiten ser detectadas, dentro y fuera de la compañía. La forma en la que fue desarrollado permite realizar cambios que no alterarían en gran medida los mecanismos de funcionamiento, donde las variables extraídas ofrecen una representación de la sintaxis de las oraciones tradicionalmente usada en las regiones centrales de Colombia, esta puede servir para la implementación de modelos recurriendo al aprendizaje por transferencia.

## 10. Referencias

- 2, A. (s.f.). *huggingface*. huggingface:  
<https://huggingface.co/gpt2?text=My+name+is+Lewis+and+I+like+to>
- AMAZON 2. (2022). *aws.amazon*. aws.amazon: <https://aws.amazon.com/es/s3/>
- AMAZON. (2022). *aws.amazon*. aws.amazon: <https://aws.amazon.com/es/comprehend/features/>
- Anlin Qu, Jianwei Ni, Shasha Mo. (2021). *aclanthology*. aclanthology:  
<https://aclanthology.org/2021.emnlp-main.237.pdf>
- ANONIMO. (08 de 2019). *stats*. stats: <https://stats.stackexchange.com/questions/421935/what-exactly-are-keys-queries-and-values-in-attention-mechanisms/424127#424127>
- ANONIMO. (27 de 07 de 2020). *redhat*. redhat: <https://www.redhat.com/es/topics/cloud-native-apps/what-is-service-oriented-architecture>
- ANONIMO. (s.f.). *huggingface*. huggingface: <https://huggingface.co/>
- arcgis. (s.f.). *pro.arcgis*. pro.arcgis: <https://pro.arcgis.com/es/pro-app/latest/tool-reference/geoai/how-entity-recognition-works.htm>
- Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser. (2017). *arxiv*. arxiv: <https://arxiv.org/abs/1706.03762>
- Bengio, Y. (2012). Practical Recommendations for Gradient-Based Training of Deep. *Practical Recommendations for Gradient-Based Training of Deep*, 33.
- CAÑETE, J., CHAPERON, G., FUENTES, R., HO, J.-H., KANG, H., & PEREZ, J. (2020). SPANISH PRE-TRAINED BERT MODEL. *SPANISH PRE-TRAINED BERT MODEL*.
- cloud.ibm. (16 de 08 de 2022). *cloud.ibm*. cloud.ibm: <https://cloud.ibm.com/apidocs/natural-language-understanding?code=python#introduction>
- CONTRERAS URGILES, W., MALDONADO ORTEGA, J., & LEON JAPA, R. (03 de 2019). *proquest*. proquest:  
<https://www.proquest.com/openview/ec87568bf6e3b960dfcb9a62ba539c18/1?pq-origsite=gscholar&cbl=4365294>
- DEVLIN, J., MING-WEI, C., LEE, K., & TOUTANOVA, K. (2018). BERT: Pre-training of Deep Bidirectional Transformers for. *Google AI Language*.
- EISENSTEIN, J. (2019). *INTRODUCTION TO NARURAL LANGUAGE PROCESSING*. LIBRARY OF CONGRESS CATALOGING IN PUBLICATION DATA.
- Escuela Politécnica Superior. Universidad de Sevilla. (23 de 02 de 2019). *sciencedirect*. sciencedirect:  
<https://www.sciencedirect.com/science/article/pii/S0031320319300950?via%3Dihub>



- Gernot A. Fink, R. J. (2022). *springer*. springer: <https://www.springer.com/journal/10032/updates>
- Goldberg, Y. (04 de 2017). *morganclaypool*. morganclaypool: <https://www.2.com/doi/abs/10.2200/S00762ED1V01Y201703HLT037>
- GOOGLE. (s.f.). *research.google*. research.google: <https://research.google.com/colaboratory/faq.html>
- IBM. (s.f.). *ibm*. ibm: <https://www.ibm.com/cloud/architecture/articles/ibmaot-redhat-openshift/04-resiliency-application-architecture-01-always-on-pattern/>
- IGI Global. (2018). *books.google*. books.google: [https://books.google.com.co/books?id=AO0\\_DwAAQBAJ&pg=PA213&lpg=PA213&dq=always+on+software+concept&source=bl&ots=ZuS1R\\_wPXr&sig=ACfU3U0nY2v6FUppqRlw50HleZSy9KXMsca&hl=es&sa=X&ved=2ahUKewiQg4m9sd33AhWysDEKHeviCH8Q6AF6BAgaEAM#v=onepage&q&f=true](https://books.google.com.co/books?id=AO0_DwAAQBAJ&pg=PA213&lpg=PA213&dq=always+on+software+concept&source=bl&ots=ZuS1R_wPXr&sig=ACfU3U0nY2v6FUppqRlw50HleZSy9KXMsca&hl=es&sa=X&ved=2ahUKewiQg4m9sd33AhWysDEKHeviCH8Q6AF6BAgaEAM#v=onepage&q&f=true)
- LI, P., ZHONG, P., MAO, K., WANG, D., YANG, X., LIU, Y., . . . SEE, S. (s.f.). ACT: an Attentive Convolutional Transformer for Efficient Text Classification. *ACT: an Attentive Convolutional Transformer for Efficient Text Classification*. <https://doi.org/https://www.aaai.org/AAAI21Papers/AAAI-1396.LiP.pdf>
- Lin, J.-W. (2017). Artificial Neural Network Related to Biological . *Neuron Network: A Review*, 8.
- microsoft. (2022). *github*. github: <https://github.com/microsoft>
- microsoft. (2022). *learn.microsoft*. learn.microsoft: <https://learn.microsoft.com/en-us/ai/>
- microsoft. (2022). *What is Named Entity Recognition (NER) in Azure Cognitive Service for Language?* <https://doi.org/https://learn.microsoft.com/en-us/azure/cognitive-services/language-service/named-entity-recognition/overview>
- Millstein, F. (2018). *Natural Language Processing With Python: Natural Language Processing Using NLTK*.
- openai. (2015-2022). *openai*. openai: <https://openai.com/blog/better-language-models/>
- ORACLE. (2022). *ORACLE*. ORACLE: <https://www.oracle.com/co/big-data/what-is-big-data/>
- OvHcloud. (2022). *pandas.pydata*. pandas.pydata.: <https://pandas.pydata.org/docs/reference/api/pandas.DataFrame.html>
- SANH, V., DEBUT, L., CHAUMOND, J., & WOLF, T. (2019). DistilBERT, a distilled version of BERT: smaller., *a distilled version of BERT: smaller,, 1, 5*.
- tensorflow. (2022). *tensorflow*. tensorflow: [https://www.tensorflow.org/tensorboard/tensorboard\\_projector\\_plugin](https://www.tensorflow.org/tensorboard/tensorboard_projector_plugin)
- Tomas Mikolov, Kai Chen, Greg Corrado, Jeffrey Dean. (2013). *arxiv*. arxiv: <https://arxiv.org/pdf/1301.3781.pdf>
- tortoisemedia. (2021). *tortoisemedia/intelligence/*. tortoisemedia: tortoisemedia

- Variš, D., & Bojar, O. (2021). Sequence Length is a Domain. *Sequence Length is a Domain*, 12.
- VASILIEV, Y. (2020). *Natural Language Processing with Python and spaCy: A Practical Introduction*.  
[https://books.google.es/books?hl=es&lr=&id=IVv6DwAAQBAJ&oi=fnd&pg=PR15&dq=spacy+nlp&ots=6XLLvYOOEH&sig=Nvtop7bFE\\_lfpqaVhFBGd5fjxaA#v=onepage&q=spacy%20nlp&f=false](https://books.google.es/books?hl=es&lr=&id=IVv6DwAAQBAJ&oi=fnd&pg=PR15&dq=spacy+nlp&ots=6XLLvYOOEH&sig=Nvtop7bFE_lfpqaVhFBGd5fjxaA#v=onepage&q=spacy%20nlp&f=false)
- Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A., . . . Polosukhin, I. (2017). Attention is All You Need. *31st Conference on Neural Information Processing Systems*.
- YINHAN, L., OTT, M., GOYAL, N., DU, J., JOSHI, M., CHEN, D., . . . STOYANOV, V. (2019). RoBERTa: A Robustly Optimized BERT Pretraining Approach. *A Robustly Optimized BERT Pretraining Approach*, 13.