**UNIVERSIDAD MAYOR REAL Y PONTIFICIA DE SAN FRANCISCO XAVIER DE CHUQUISACA**

FACULTAD DE CIENCIAS Y TECNOLOGÍA

**CARRERA DE INGENIERÍA EN CIENCIAS DE LA COMPUTACIÓN**

PERFIL DE PROYECTO DE GRADO



**“MODELO GENERADOR DE ESTILOS CSS EN BASE A ETIQUETAS HTML PARA DESARROLLADORES WEB”**

TUMIRI HUANCA ALEX

Sucre, 26 de mayo del 2023

1. **TÍTULO**

MODELO GENERADOR DE ESTILOS CSS EN BASE A ETIQUETAS HTML PARA DESARROLLADORES WEB

1. **INTRODUCCIÓN Y/O ANTECEDENTES**

En la actualidad el lenguaje de diseño y programación CSS tiene muchas ventajas, características, librerías, API’s y es utilizado en todos los navegadores y plataformas web del mundo, siendo así un estándar en el desarrollo de interfaces de usuario y en el desarrollo de aplicaciones web, el problema no reside en que la tecnología CSS sea una tecnología ineficiente para el desarrollo de software, o que tenga problemas de compatibilidad, o una ruta de aprendizaje compleja, todo lo contrario.

Los desarrolladores de software sobre todo especializados en la implementación de diseños e interfaces gráficas durante mucho tiempo han intentado hacer su trabajo más eficiente y productivo en esta exhaustiva búsqueda de eficiencia y productividad los mismos programadores he ingenieros de software, han desarrollado diversas técnicas y herramientas las cuales facilitan su trabajo a la hora de crear software.

Existen muchas librerías y herramientas de CSS que pueden ayudarnos a generar plantillas y diseños web, estas herramientas aumentan la productividad y eficiencia siendo algunas de estas Bootstrap, Materialize CSS, Tailwind CSS, Semantic UI, etc. Sin embargo también aumentan la curva de aprendizaje y el tiempo para aprender las diferentes herramientas que constantemente se van actualizando día a día, sin mencionar que el proceso de desarrollo de estilos CSS no solo depende, que el desarrollador de software conozca muy bien la tecnología y los conceptos de la misma si no que tenga como vulgarmente se dice “un muy buen gusto”, el cual es importante para tener éxito con el desarrollo de la Interfaz de Usuario (UI) y Experiencia de usuario (UX), mientras que uno hace referencia a la experiencia y sensación del usuario, el otro está dirigido hacia un lado más racional de la navegación, también se deben considerar otros desafíos mayores como el buen diseño.

Según el sitio web BuiltWith, que rastrea el uso de tecnologías web en todo el mundo, Bootstrap es la librería de CSS más utilizada en la actualidad, con un 16% de participación de mercado. Le sigue Materialize CSS con un 0,9%, Foundation con un 0,8%, Tailwind CSS con un 0,6%, Semantic UI con un 0,4%, y Bulma con un 0,2%. (BuiltWith, 2019)

Bootstrap es la librería de CSS más utilizada debido a su amplia comunidad de usuarios, su diseño y funcionalidad responsivos, su conjunto completo de componentes y estilos predefinidos, y su capacidad de personalización. (Bootstrap, 2023)

Estas características hacen que sea una herramienta muy útil para los desarrolladores que buscan crear diseños y aplicaciones web modernas y adaptativas.

Bootstrap está constituido por una serie de archivos CSS y JavaScript responsables de asignar características específicas a los elementos de la página. (Author, 2020)

Las herramientas anteriormente mencionadas también brindan un diseño estándar el cual es fácilmente reconocible siendo este el caso de Bootstrap y no es de muy buen gusto por lo cual no agrega valor al diseño de la interfaz.

1. **SITUACIÓN PROBLEMÁTICA**

* Existe poca productividad por parte de los ingenieros de software a la hora de desarrollar interfaces de usuario, diseño y estilos con el lenguaje de diseño gráfico CSS. Debido a la complejidad de la implementación de un diseño web efectivo y la gran cantidad de líneas de código que esto puede requerir, muchos desarrolladores pueden enfrentarse a un proceso de desarrollo lento y tedioso este proceso consiste en escribir código repetitivo constantemente.
* Estas librerías, frameworks y APIs anteriormente mencionadas comúnmente carecen de flexibilidad a la hora de desarrollar ofreciendo muchas veces estándares de diseños con colores predeterminados que no son de buen gusto y que para lograr el resultado deseado por el desarrollador se debe escribir más código e incluso tener más conocimientos sobre la librería que se esté utilizando, muchas veces el trabajo solo se incrementa tanto en líneas de código como en tiempo de desarrollo, reduciendo la productividad y aumentando la complejidad del desarrollo y obviamente aumentando también el tiempo de desarrollo, es cierto que el uso excesivo de estas herramientas puede llevar a una falta de flexibilidad y creatividad en el diseño y la implementación de una aplicación o sitio web. Por lo tanto, es importante que los desarrolladores encuentren el equilibrio adecuado entre el uso de estas herramientas y la implementación de soluciones personalizadas y creativas.
* En algunas situaciones, el desarrollador de software puede tener la capacidad y conocimiento técnico para desarrollar interfaces de muy buena calidad con CSS, pero resulta que carece del buen gusto y habilidades de diseño necesarias para crear una interfaz de usuario atractiva y funcional. El buen gusto y la habilidad de diseño son esenciales para crear una experiencia de usuario efectiva y agradable, y esto a menudo requiere de una combinación de habilidades técnicas y creativas. Si un desarrollador de software carece de estas habilidades, puede resultar en una interfaz de usuario poco atractiva, difícil de usar y menos efectiva.
* Por lo general las diferentes herramientas para CSS que existen en la actualidad son muy diversas y muchas veces se obtiene con estas el mismo resultado, también tienen una curva de aprendizaje compleja sin mencionar que se debe conocer muy bien los conceptos y sintaxis del lenguaje CSS, por esta razón no son alternativas factibles para desarrolladores sin experiencia con CSS o que se estén iniciando en el desarrollo de interfaces de usuario. Existen varias herramientas de CSS que pueden tener una curva de aprendizaje larga e ineficiente para algunos desarrolladores, dependiendo del nivel de experiencia y habilidades técnicas. Algunas de estas herramientas son:
* Sass y Less: Sass (Syntactically Awesome Style Sheets) y Less (Leaner CSS) son preprocesadores de CSS que permiten a los desarrolladores escribir CSS de manera más eficiente y organizada. Sin embargo, su sintaxis y estructura pueden ser más complejas que el CSS convencional, lo que puede requerir tiempo y esfuerzo para aprender.
* CSS Grid y Flexbox: CSS Grid y Flexbox son herramientas de diseño de diseño web que permiten a los desarrolladores crear diseños de página más complejos y flexibles. Sin embargo, la curva de aprendizaje para estas herramientas puede ser larga y requiere un conocimiento detallado de las propiedades de CSS y cómo se aplican.
* Animaciones CSS: Las animaciones CSS permiten a los desarrolladores agregar efectos y animaciones a las páginas web, pero pueden requerir un conocimiento detallado de las propiedades de animación y transición de CSS.
* CSS preprocessors: Además de Sass y Less, existen otros preprocesadores de CSS como Stylus y PostCSS. Cada uno de estos preprocesadores tiene su propia sintaxis y estructura, lo que puede requerir un tiempo adicional para aprender.

Cada herramienta de CSS también tiene sus propios pros y contras, y la elección de la herramienta adecuada dependerá de las necesidades y objetivos específicos del proyecto. A continuación, se presentan algunos de los pros y contras de las herramientas de CSS más populares:

* Bootstrap es una de las herramientas de CSS más populares y ampliamente utilizadas, lo que significa que hay una gran comunidad de desarrolladores y recursos disponibles en línea. Bootstrap es fácil de usar y tiene una curva de aprendizaje corta, lo que lo hace ideal para proyectos de desarrollo web rápidos. Además, Bootstrap ofrece una gran cantidad de componentes, como botones, formularios y carruseles, que se pueden implementar fácilmente en cualquier proyecto. Debido a su popularidad, Bootstrap a menudo se utiliza para crear sitios web que se ven muy similares. Además, Bootstrap tiene una gran cantidad de clases de estilo, lo que puede aumentar el tamaño de los archivos CSS y ralentizar el rendimiento de la página.
* Por otro lado, herramientas como Tailwind CSS, Semantic UI son altamente personalizables y modulares, lo que permite a los desarrolladores crear diseños únicos y específicos para sus proyectos son muy fáciles de usar, pero pueden ser menos conocidas que otras herramientas de CSS, lo que puede dificultar la búsqueda de recursos y apoyo en línea. Además, al igual que con otras herramientas de CSS, aumentar la cantidad de clases de estilo en el código, lo que puede dificultar la lectura y el mantenimiento.

El desarrollo de software con la tecnología CSS como con otras tecnologías tiende a ser repetitivo, poco intuitivo, ineficiente y en su mayoría reduce la productividad del ingeniero de software a causa de las malas prácticas, desconocimiento de la tecnología CSS y otros factores que hacen del ingeniero de software sea improductivo a la hora de desarrollar software de buena calidad.

1. **PROBLEMA CENTRAL**

El uso de CSS en diseño web es complejo y requiere de muchas líneas de código, lo que disminuye la productividad y las librerías y frameworks de CSS estandarizadas limitan la flexibilidad y creatividad, también la falta de habilidades de diseño en los desarrolladores puede generar interfaces ineficaces y la complejidad de aprendizaje de las herramientas de CSS es un obstáculo para los principiantes en el desarrollo de interfaces de usuario esto resulta en un problema crítico en el desarrollo de interfaces.

1. **ABORDAJE DE SOLUCIÓN**

El desarrollo de un modelo de inteligencia artificial generará estilos CSS predefinidos y con características específicas, reconociendo el código HTML introducido por el desarrollador de software en una aplicación web intuitiva y fácil de usar que tendrá como motor de generación de código el modelo anteriormente mencionado, para reducir de manera notable el tiempo de desarrollo, aumentando la productividad y eficiencia del ingeniero de software, evitando que el ingeniero en cuestión escriba código repetitivo reduciendo así el tiempo de desarrollo con el cual el desarrollador de software podrá gestionar de manera más productiva su tiempo para optimizar y desarrollar un código de buena calidad y por consiguiente un mejor producto.

Se recopilará un gran volumen de datos de estilos CSS para el preprocesamiento de las diferentes características de la sintaxis del lenguaje CSS y clasificación de datos obtenidos para crear dos diferentes sets de datos que serán utilizados para el entrenamiento y pruebas de nuestro modelo generador de código CSS.

Como alternativa tecnológica para el desarrollo del modelo generador de código CSS se considera, luego de un análisis profundo e investigación sobre el estado actual del desarrollo de modelos de inteligencia artificial, es difícil ignorar el avance que se realizó en los últimos años en el área del Procesamiento del lenguaje natural (NLP) es una tecnología que brinda a las computadoras la capacidad interpretar, manipular, y comprender el lenguaje humano esta tecnología será de vital utilidad para reconocer, clasificar y extraer texto para que el modelo pueda aprender la sintaxis del lenguaje de programación CSS.

La arquitectura Transformer de redes neuronales será utilizada para el entrenamiento y optimización del modelo, “[…]brinda la innovadora técnica del procesamiento en paralelo de la secuencia de parámetros introducidos a diferencia de las redes neuronales recurrentes donde lo parámetros se procesan de forma serial, a continuación, un gráfico de la arquitectura Transformer”. (Sotaquirá, 2020)

Los módulos de codificación y decodificación, también se utiliza las técnicas de embedding y los bloques atencionales toda esta arquitectura es muy eficiente e innovadora en el desarrollo de modelos de procesamiento de lenguaje natural y generación de texto específicamente, optimizando y ajustando la arquitectura Transformer a nuestro caso en específico obtendremos como resultado un eficiente y potente modelo de generación de código CSS. (Sotaquirá, 2020)

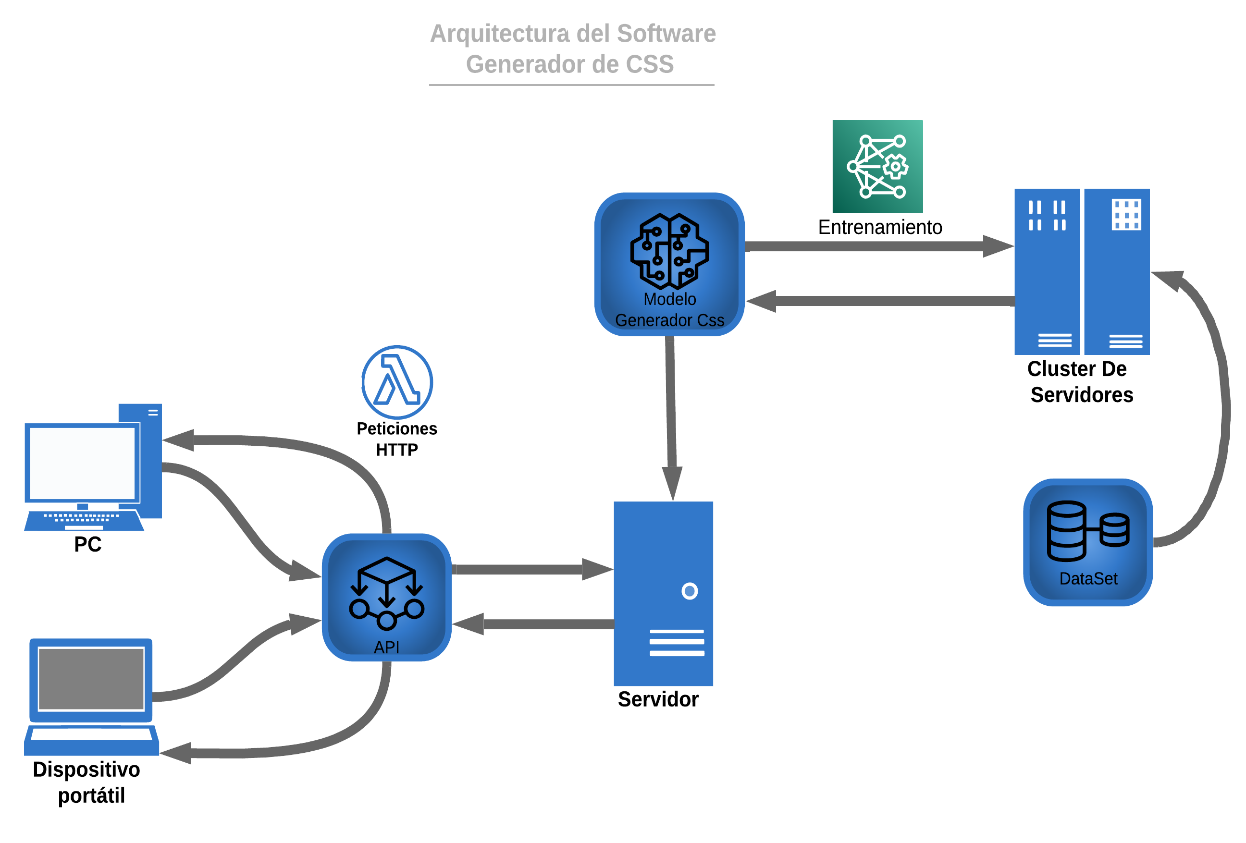


Figura 1: Arquitectura del Software de Generación CSS

Fuente: Elaboración Propia

1. **OBJETIVO GENERAL**

Desarrollar un modelo de inteligencia artificial para la generación de código CSS predefinido y con buen diseño, que facilite el trabajo de los desarrolladores de software, reduciendo así el tiempo de desarrollo al generar diseños predefinidos y personalizables.

1. **OBJETIVOS ESPECÍFICOS**

* Desarrollar una aplicación web para generar código CSS y brinde una previsualización en tiempo real de vistas generadas en base a etiquetas HTML.
* Mejorar la eficiencia del código generado considerando la reducción de la complejidad ciclomática la reducción del código repetitivo y líneas de código.
* Mejorar la eficiencia y la productividad en el código generado respecto a la programación tradicional.

1. **JUSTIFICACIÓN**
   1. **Justificación Tecnológica**

Las herramientas que se utilizan en la actualidad no son suficientes para generar código Css intuitivo y con buen diseño dado que son generadores de código estándar los cuales no solo se concentran en una tecnología en particular si no en diferentes tecnologías, esto hace que estos modelos generadores de código sean más generalistas que especialistas, muchos de los generadores actuales son muy eficientes sin embargo no brindan una interfaz dedicada para el desarrollo de software e implementación de código y está claro que muchas de las herramientas en la actualidad generan código pero no generan otras alternativas en cuestión de vistas y ejemplos de cómo se visualizara la implementación de código Css una vez integrado a los proyectos de los desarrolladores de software y aunque en la actualidad día a día se van desarrollando plugins y diferentes aplicaciones para el uso de estos generadores de código. Esto nos brinda muchas alternativas y oportunidades para el desarrollo de modelos de Inteligencia artificial generadores de texto aplicando NLP (Procesamiento del Lenguaje Natural), tanto en el software como en la generación de código CSS.

La arquitectura Transformers de redes neuronales que se empleara para la optimización y entrenamiento de nuestro modelo Sistema de generación de código CSS mediante la entrada de código HTML.

1. **PLANIFICACIÓN Y CRONOGRAMA**

Para la planificación del proyecto se utilizará la metodología ágil que es un conjunto de técnicas aplicadas en ciclos de trabajo cortos, con el objetivo de que el proceso de entrega de un proyecto sea más eficiente, también se utilizara la herramienta Kanban que es un método de gestión del flujo de trabajo que ayuda a las organizaciones a gestionar y mejorar los sistemas de trabajo.

Como cronograma se estima que el tiempo requerido para el desarrollo del proyecto será de 20 semanas, también se utilizará el Diagrama de Gantt para organizar y ejecutar el cronograma de manera eficiente a continuación mostrado.

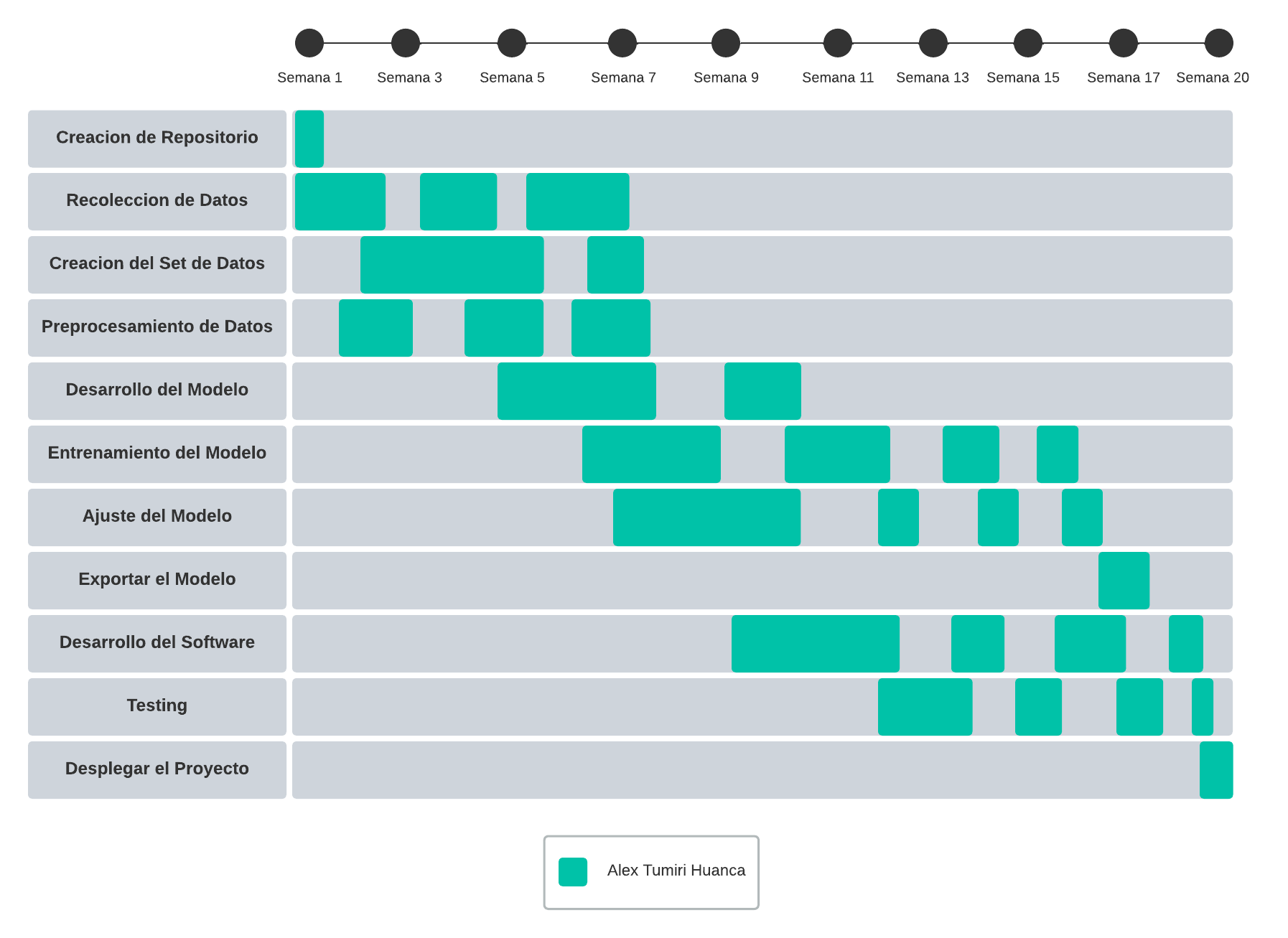


Figura 2: Diagrama de Gantt

Fuente: Elaboración Propia

1. **REFERENCIAS BIBLIOGRAFÍCAS**

**Author, Guest. 2020.** rockcontent. *rockcontent.* [Online] © 2013-2021 Rock Content, Abril 12, 2020. [Cited: Abril 24, 2023.] https://rockcontent.com/es/blog/bootstrap/.

**Bootstrap. 2023.** getbootstrap. *getbootstrap.* [Online] Bootstrap, Enero 15, 2023. [Cited: Abril 24, 2023.] https://getbootstrap.com/.

**BuiltWith. 2019.** BuiltWith. *BuiltWith.* [Online] BuiltWith® Pty Ltd, Octubre 24, 2019. [Cited: Abril 23, 2023.] https://builtwith.com.

**datascientest. 2023.** datascientest.com. *datascientest.* [Online] DataScientest, 8 10, 2023. [Cited: 6 6, 2023.] https://datascientest.com/es/inteligencia-artificial-definicion.

**Sotaquirá, Miguel. 2020.** codificandobits. *codificandobits.* [Online] © 2023 Codificando Bits, Junio 30, 2020. [Cited: Abril 24, 2023.] https://www.codificandobits.com/blog/redes-transformer/#la-red-transformer.

**ANEXOS**

**ANEXO I: MARCO CONTEXTUAL**

**Podemos hablar de la inteligencia artificial como esta en la actualidad con las nuevas tendencias que han revolucionado en algunos campos como el desarrollo de software con el movimiento no code y como en la actualidad hay muchas herramientas de generación de inteligencia artificiales que nos ayudan a generar código, pequeño repaso de copilot, chat gpt , etc**

**También se puede hablar de los Transformers y como se utiliza esta tecnología en la actualidad, y como es**

**Para entender donde se desenvuelve el problema como se aplica la tecnología ia, tranformers y etc en el mundo de la programación como se aplican los modelos cual es la situación actual, como los programadores utilizan la IA o no la utilizan o la subutilizan EN SI TODO LO QUE SE HABLO EN SITUACION PROBLEMÁTICA Y LOS ANTECEDENTES PERO EN DETALLE (todo lo que es teoría no entra al marco contextual)**

**Podemos agregar un árbol de problemas**

**Diagrama causa y efecto**

* **Inteligencia Artificial**

En la actualidad, la inteligencia artificial (IA) ha experimentado avances significativos en diversos campos, incluido el desarrollo de software. Las nuevas tendencias, como el movimiento "no code" y las herramientas de generación de IA, están revolucionando la forma en que se crea y se trabaja con código.

El movimiento "no code" se refiere a la creación de aplicaciones y software sin la necesidad de escribir código tradicional.

Esto se logra a través de plataformas y herramientas que permiten a los usuarios utilizar interfaces visuales e interacciones intuitivas para crear aplicaciones, automatizaciones y flujos de trabajo complejos. Estas herramientas se basan en la IA y el aprendizaje automático para comprender y traducir las intenciones del usuario en código subyacente.

Este movimiento ha democratizado el desarrollo de software al hacerlo accesible a personas sin conocimientos de programación profundos.

en la actualidad hay muchas herramientas de inteligencia artificial de generación de texto y código algunas de estas son OpenAI Codex y GPT-3, DeepCode, Bard, ChatGPT que en la actualidad se encuentra en su cuarta versión y ha revolucionado la manera en que entendemos la inteligencia artificial y sus aplicaciones en nuestro diario vivir, incluso ha causado preocupación en la comunidad científica por el avance que la inteligencia ha tenido aproximadamente en tan solo cinco años.

También existen herramientas de autocompletado de código Kite, Tabnine siendo Github Copilot la herramienta más utilizada para el autocompletado de código se basa en la tecnología de inteligencia artificial y aprendizaje automático, específicamente en el modelo de IA GPT-3 de OpenAI, para ayudar a los desarrolladores a escribir código de manera más eficiente y productiva.

* **Transformers**

Los Transformers son una arquitectura de redes neuronales que se ha vuelto muy popular y ampliamente utilizada en la actualidad, especialmente en tareas relacionadas con el procesamiento del lenguaje natural (NLP, por sus siglas en inglés). En la actualidad los Transformers se utiliza en Modelos de lenguaje preentrenados como BERT (Bidirectional Encoder Representations from Transformers) y GPT (Generative Pre-trained Transformer), se entrenan con grandes cantidades de texto para aprender representaciones del lenguaje.

Chatbots y asistentes virtuales. Los Transformers se utilizan para desarrollar chatbots y asistentes virtuales que pueden mantener conversaciones coherentes y responder preguntas de manera inteligente.

Otros usos muy frecuentes de los Transformers son Traducción Automática, Resumen Automático, Generación de Texto y Código.

* **Lenguajes CSS**

El lenguaje CSS (Cascading Style Sheets) es un lenguaje de hojas de estilo utilizado para definir la apariencia y el formato de un documento HTML. El entorno en el que se desarrolla y se utiliza CSS son los siguientes:

* **Desarrollo web:** CSS es fundamental en el desarrollo web moderno. Se utiliza para controlar la presentación visual de los sitios web, incluyendo el diseño, los colores, las fuentes, los efectos visuales y la disposición de los elementos en la página que también incluye el conocimiento de las mejores prácticas de diseño web, los estándares de la industria y las tendencias actuales en términos de diseño y usabilidad.
* **Compatibilidad con navegadores:** CSS se ejecuta en el lado del cliente, lo que significa que su comportamiento y rendimiento pueden variar según el navegador utilizado por los usuarios de CSS incluye la comprensión de las diferencias de compatibilidad entre los diferentes navegadores y la adopción de enfoques que permitan una experiencia consistente en la mayoría de los navegadores.
* **Selección y especificidad:** CSS utiliza selectores para aplicar estilos a los elementos HTML. Se debe tener conocimiento de los diferentes tipos de selectores, como los selectores de etiqueta, los selectores de clase, los selectores de ID y los selectores de atributo. También es importante comprender la especificidad de los selectores, que determina qué estilos se aplican cuando hay conflictos.
* **Reactividad y diseño adaptable:** En la actualidad, los sitios web deben adaptarse a diferentes dispositivos y tamaños de pantalla. CSS proporciona herramientas para crear diseños receptivos y adaptables, utilizando técnicas como media queries, flexbox y grid. Se debe tener comprensión de cómo crear diseños fluidos y receptivos que se ajusten a diferentes dispositivos y tamaños de pantalla.
* **Preprocesadores y herramientas de construcción:** Para facilitar el desarrollo y la gestión del código CSS, se utilizan preprocesadores como Sass o Less, que permiten utilizar características avanzadas como variables, mixins y funciones. Además, existen herramientas de construcción y automatización, como Gulp o Webpack, que ayudan a optimizar y mejorar el rendimiento del código CSS. Estas herramientas y prácticas sirven para mejorar la eficiencia y la calidad del código CSS.
* **Accesibilidad:** CSS desempeña un papel importante en la accesibilidad web al permitir la creación de diseños y estilos que sean legibles y utilizables para personas con discapacidades visuales o cognitivas. las pautas de accesibilidad y las mejores prácticas son importantes para garantizar que los estilos aplicados no obstaculicen la experiencia de los usuarios con discapacidades.
* **Tendencias y evolución:** El seguimiento de las tendencias y la evolución del lenguaje. CSS está en constante desarrollo, con nuevas especificaciones y características que se introducen regularmente. Es importante mantenerse actualizado con las últimas tendencias y características de CSS para aprovechar al máximo sus capacidades y mantenerse al día con las mejores prácticas en el desarrollo web.
  + **Diseños con CSS:** Los diseños con CSS abarca los principios de diseño, la estructura HTML, los layouts y disposiciones, los estilos y efectos visuales, la compatibilidad con navegadores, los frameworks y bibliotecas CSS, así como las tendencias y estilos actuales. Es fundamental utilizar CSS de manera efectiva y crear diseños web atractivos y funcionales.
  + **Herramientas y Frameworks de CSS:** Las herramientas y frameworks de CSS mejora la eficiencia en el desarrollo, el diseño responsive, la reutilización de componentes, la compatibilidad con navegadores, la optimización y el rendimiento, la documentación y comunidad, y la integración con otros frameworks y tecnologías. Comprender este marco contextual es fundamental para aprovechar al máximo estas herramientas y frameworks y mejorar la productividad y calidad en el desarrollo de proyectos web.
* **Modelo Generador de código**

Un modelo de inteligencia artificial generador de código es un sistema diseñado para generar automáticamente código fuente de programación a partir de una descripción o especificación dada. Estos modelos utilizan técnicas de aprendizaje automático, en particular el aprendizaje profundo, para aprender patrones y reglas de programación a partir de grandes conjuntos de datos de código fuente existente.

El objetivo principal de estos modelos es automatizar y agilizar el proceso de desarrollo de software, especialmente en tareas repetitivas o rutinarias que pueden ser propensas a errores humanos. Al utilizar un generador de código, los desarrolladores pueden acelerar el desarrollo de aplicaciones y sistemas, permitiéndoles centrarse en tareas de mayor nivel, como el diseño arquitectónico o la resolución de problemas complejos.

Los modelos de generación de código pueden abordar diferentes niveles de abstracción en la generación de código. Algunos modelos pueden generar código a nivel de funciones o módulos, mientras que otros pueden generar código completo para una aplicación o incluso para múltiples componentes interconectados. Estos modelos suelen entrenarse en grandes repositorios de código fuente, donde aprenden las estructuras sintácticas, las convenciones de nomenclatura y los patrones comunes que se encuentran en el código existente.

A partir de esta información, el modelo puede inferir y generar código nuevo que siga las pautas aprendidas. Sin embargo, es importante tener en cuenta que los modelos de generación de código no son una solución perfecta y no reemplazan completamente el trabajo de los desarrolladores humanos. Aunque pueden ser útiles para generar partes de código repetitivas o rutinarias, aún requieren la supervisión y validación humana para garantizar la calidad, la eficiencia y la seguridad del código generado.

* + **Generación de textos**

La generación de textos con inteligencia artificial abarca el entorno en el que se utilizan técnicas y modelos de IA para generar automáticamente texto con cierto nivel de coherencia y comprensión. Algunos aspectos importantes de la generación de textos con inteligencia artificial son los siguientes:

* + - Procesamiento del Lenguaje Natural (NLP): La generación de texto con inteligencia artificial se basa en técnicas y algoritmos de procesamiento del lenguaje natural. Esto implica comprender conceptos y técnicas como el análisis gramatical, el reconocimiento de entidades, la desambiguación semántica, la generación de lenguaje natural y la comprensión del contexto lingüístico.
    - Modelos de Lenguaje: Los modelos de lenguaje, como los modelos de aprendizaje automático basados en redes neuronales, son fundamentales en la generación de texto con inteligencia artificial. Estos modelos aprenden de grandes cantidades de datos de texto para capturar patrones y estructuras lingüísticas. La arquitectura y el funcionamiento de estos modelos, como los modelos de lenguaje están basados en Transformers.
    - Datos de entrenamiento: Los modelos de generación de texto se entrenan con grandes conjuntos de datos de texto, como libros, artículos, noticias o corpus específicos. Se debe recopilar, limpiar y preparar los datos de entrenamiento para garantizar resultados de calidad en la generación de texto.
    - Generación creativa: La generación de texto con inteligencia artificial no se limita a la reproducción de patrones existentes, sino que también puede ser creativa y generar texto original. También se pueden aplicar técnicas como la generación condicionada, la generación estocástica y el equilibrio entre la coherencia y la diversidad en la generación de texto.
    - Evaluación y refinamiento: La inteligencia artificial requiere una evaluación cuidadosa y un refinamiento iterativo. Estos implican desarrollar métodos para evaluar la calidad del texto generado, como la coherencia, la fluidez, la relevancia y la ausencia de sesgos. Además, implica realizar ajustes y mejoras en los modelos y en los datos de entrenamiento para lograr resultados más precisos y adecuados.
    - Aplicaciones y casos de uso: La generación de texto tiene una amplia gama de aplicaciones y casos de uso, que van desde la generación automática de contenido, la asistencia en la escritura, el resumen de textos, la traducción automática, hasta la interacción con sistemas de chatbots y asistentes virtuales. Queda demostrado que se puede aplicar la generación de texto en diferentes dominios y escenarios.
    - Consideraciones éticas y de responsabilidad: La inteligencia artificial plantea desafíos éticos y de responsabilidad, como la propagación de información falsa o sesgada. Esto implica considerar y abordar estos aspectos, aplicando principios éticos, transparencia en el uso de los modelos y sistemas, y mitigación de sesgos y discriminación en la generación de texto.
  + **Generación de código**

La Generación de código con inteligencia artificial es el entorno en el que se utilizan técnicas y modelos de IA para generar automáticamente código de programación. Algunos aspectos importantes de la Generación de código con inteligencia artificial son los siguientes:

* + - Lenguajes de programación: La Generación de código con inteligencia artificial puede abarcar diferentes lenguajes de programación, como CSS, Python, JavaScript, Java, C++, entre otros. Para esto el modelo debe comprender los diferentes lenguajes de programación y las mejores prácticas asociadas a cada uno de ellos.
    - Modelos de aprendizaje automático: La Generación de código se basa en modelos de aprendizaje automático, como redes neuronales y algoritmos de aprendizaje profundo. Estos modelos aprenden de ejemplos de código existentes para generar código nuevo.
    - Datos de entrenamiento: Los modelos de Generación de código se entrenan utilizando conjuntos de datos de código fuente existente. Estos conjuntos de datos pueden incluir repositorios de código, documentación técnica y ejemplos de código. Se debe recopilar, limpiar y preparar los datos de entrenamiento para obtener resultados precisos y de calidad.
    - Sintaxis y estructura del código: La Generación de código con inteligencia artificial implica comprender la sintaxis y la estructura del código en el lenguaje de programación específico y también tener conocimientos sólidos sobre los estándares y las convenciones de codificación, así como las mejores prácticas de programación.
    - Optimización y eficiencia del código: El Modelo puede enfocarse en la generación de código eficiente y optimizado. Esto implica tener en cuenta aspectos como la legibilidad del código, la optimización de rendimiento, la gestión de recursos y la aplicación de buenas prácticas de programación.
    - Integración con herramientas y entornos de desarrollo: El generador de código puede integrarse con herramientas y entornos de desarrollo existentes, como IDEs (entornos de desarrollo integrados), sistemas de control de versiones y herramientas de depuración.
    - Validación y pruebas: Para generar código de calidad se debe validar y probar el código generado para garantizar su calidad y funcionamiento correcto, para lograr esto se debe desarrollar técnicas y metodologías de validación y prueba específicas para el código generado automáticamente.

**ANEXO II: MARCO TEÓRICO**

* **Inteligencia Artificial**

La IA es el estudio de los agentes que reciben percepciones del entorno y realizan acciones. (datascientest, 2023)

A continuación, se presentan algunas de las principales formas en que funciona la IA en la actualidad:

* Aprendizaje automático (Machine Learning): El aprendizaje automático es una rama de la IA que se basa en algoritmos y modelos para permitir a las máquinas aprender y mejorar a partir de datos. Los algoritmos de aprendizaje automático pueden identificar patrones y realizar predicciones o tomar decisiones basadas en conjuntos de datos históricos. Se utilizan técnicas como el aprendizaje supervisado, no supervisado y por refuerzo para entrenar modelos y permitirles aprender de manera autónoma.
* Redes Neuronales Artificiales: Las redes neuronales artificiales son modelos inspirados en el funcionamiento del cerebro humano. Estas redes están compuestas por nodos interconectados llamados neuronas artificiales, que procesan y transmiten información. Las redes neuronales son ampliamente utilizadas en tareas como el reconocimiento de imágenes y voz, la traducción automática y el procesamiento del lenguaje natural.
* Procesamiento del Lenguaje Natural (NLP): El NLP es un campo de la IA que se centra en la interacción entre las computadoras y el lenguaje humano. Los sistemas de NLP permiten a las máquinas comprender, interpretar y generar texto de manera similar a como lo haría un ser humano. Esto se aplica en tareas como la traducción automática, la generación de texto, la clasificación de sentimientos y el procesamiento de voz a texto.
* Visión por Computadora: La visión por computadora se ocupa del análisis, procesamiento e interpretación de imágenes y videos por parte de las máquinas. Utiliza algoritmos de IA para reconocer objetos, realizar seguimiento de movimientos, detectar rostros, analizar imágenes médicas, entre otras aplicaciones. Esto se aplica en campos como la conducción autónoma, la vigilancia de seguridad, la realidad aumentada, entre otros.
* Robótica y Automatización: La IA se utiliza ampliamente en robots y sistemas automatizados para permitirles realizar tareas complejas y adaptarse a diferentes situaciones. Los robots equipados con IA pueden realizar tareas de ensamblaje, navegación, reconocimiento de objetos y colaboración con humanos. La automatización de procesos empresariales también se beneficia de la IA al utilizar algoritmos de aprendizaje automático para automatizar tareas repetitivas y mejorar la eficiencia.
* Sistemas de Recomendación: Los sistemas de recomendación utilizan algoritmos de IA para analizar y comprender los patrones de comportamiento de los usuarios y proporcionar recomendaciones personalizadas. Estos sistemas se aplican en plataformas de comercio electrónico, servicios de streaming, redes sociales y otras aplicaciones en línea para ofrecer sugerencias de productos, música, películas, contenido y conexiones sociales relevantes para cada usuario.
* Asistentes Virtuales y Chatbots: Los asistentes virtuales y chatbots utilizan IA para interactuar con los usuarios y brindar respuestas y servicios. Estos sistemas pueden entender y responder preguntas, realizar tareas específicas, brindar información y asistencia en tiempo real. Ejemplos conocidos son Siri de Apple, Google Assistant, Alexa de Amazon y chatbots de atención al cliente.
  + **Redes Neuronales**

Las redes neuronales son un modelo computacional inspirado en el sistema nervioso biológico. Estas redes están compuestas por unidades de procesamiento interconectadas, llamadas neuronas artificiales o nodos, que trabajan en conjunto para resolver problemas y realizar tareas de aprendizaje automático.

Componentes básicos de una red neuronal:

* + - Neuronas artificiales: Las neuronas artificiales son los bloques fundamentales de una red neuronal. Cada neurona recibe una o varias entradas, realiza un cálculo y produce una salida. El cálculo generalmente involucra una combinación lineal de las entradas ponderadas por los pesos sinápticos, seguido de la aplicación de una función de activación no lineal.
    - Pesos sinápticos: Los pesos sinápticos son parámetros ajustables asociados a las conexiones entre las neuronas. Estos pesos determinan la influencia relativa de las diferentes entradas en la salida de una neurona. Durante el entrenamiento de la red neuronal, los pesos sinápticos se ajustan para minimizar una función de pérdida o error.
    - Funciones de activación: Las funciones de activación introducen no linealidad en la red neuronal, lo que les permite modelar relaciones complejas en los datos. Algunas funciones de activación comunes incluyen la función sigmoide, la función ReLU (Rectified Linear Unit) y la función tangente hiperbólica.
    - Capas: Las neuronas se organizan en capas dentro de una red neuronal. Las capas típicas en una red neuronal incluyen una capa de entrada, una o varias capas ocultas y una capa de salida. Las capas ocultas son responsables de extraer características y representaciones más abstractas de los datos de entrada.
    - Conexiones: Las neuronas dentro de una red neuronal están conectadas mediante conexiones ponderadas. Estas conexiones transmiten las salidas de las neuronas anteriores a las entradas de las neuronas siguientes. Dependiendo de la arquitectura de la red neuronal, las conexiones pueden ser feedforward (unidireccionales) o recurrentes (bidireccionales).
  + **Modelos de Inteligencia Artificial**

Los modelos de Inteligencia Artificial (IA) son algoritmos y estructuras computacionales diseñados para simular el proceso de pensamiento y toma de decisiones humanas. Estos modelos se basan en diversas técnicas y enfoques que permiten a las máquinas aprender, comprender, razonar y tomar decisiones de manera similar a los seres humanos.

Algunos de los principales modelos de Inteligencia Artificial más utilizados en la actualidad son:

* + - Redes Neuronales Artificiales (RNA): Están inspiradas en la estructura y funcionamiento del cerebro humano. Consisten en un conjunto interconectado de nodos (neuronas) que procesan la información y la transmiten a través de conexiones ponderadas. Las RNA son ampliamente utilizadas en áreas como el reconocimiento de imágenes, procesamiento del lenguaje natural y reconocimiento de voz.
    - Aprendizaje Automático (Machine Learning): Es una rama de la IA que se enfoca en el desarrollo de algoritmos y modelos que permiten a las máquinas aprender patrones y tomar decisiones basadas en datos, sin ser programadas explícitamente. Dentro del aprendizaje automático, existen diferentes enfoques, como el aprendizaje supervisado, no supervisado y por refuerzo.
    - Aprendizaje Profundo (Deep Learning): Es una subcategoría del aprendizaje automático que utiliza redes neuronales artificiales de múltiples capas para extraer características y aprender representaciones complejas de los datos. El aprendizaje profundo ha revolucionado áreas como el reconocimiento de imágenes, el procesamiento del lenguaje natural y la traducción automática.
    - Sistemas Expertos: Son sistemas de software diseñados para emular el conocimiento y la experiencia de un experto humano en un dominio específico. Utilizan reglas y heurísticas para tomar decisiones y proporcionar recomendaciones. Los sistemas expertos se aplican en campos como la medicina, la ingeniería y el soporte técnico.
    - Procesamiento del Lenguaje Natural (NLP): Se refiere a la capacidad de las máquinas para comprender, interpretar y generar lenguaje humano de manera natural. El NLP se utiliza en aplicaciones como el reconocimiento de voz, la traducción automática, la generación de texto y los chatbots.
    - Aprendizaje por Refuerzo (Reinforcement Learning): Es un enfoque del aprendizaje automático en el que un agente aprende a través de la interacción con un entorno. El agente toma decisiones y recibe recompensas o penalizaciones según su desempeño. El objetivo es que el agente aprenda a tomar las acciones que maximicen las recompensas a largo plazo.
    - Redes Generativas Adversariales (GAN): Son modelos compuestos por dos redes neuronales, un generador y un discriminador, que compiten entre sí. El generador crea muestras sintéticas que intentan engañar al discriminador, mientras que este último busca distinguir entre las muestras reales y falsas. Las GAN se utilizan para generar imágenes, música y texto sintético de alta calidad.
  + **Procesamiento del Lenguaje Natural (NLP)**

El Procesamiento del Lenguaje Natural es una rama de la Inteligencia Artificial que se ocupa de la interacción entre las computadoras y el lenguaje humano. El objetivo principal del NLP es permitir que las máquinas comprendan, interpreten, generen y respondan al lenguaje humano de manera natural y significativa.

A continuación, se presentan los elementos clave dentro del NLP:

* + - Lingüística computacional: Se refiere al estudio y la aplicación de principios lingüísticos en la construcción de sistemas de procesamiento del lenguaje natural. La lingüística computacional proporciona los fundamentos teóricos para entender la estructura y la gramática del lenguaje humano, incluyendo la sintaxis, la semántica y la pragmática.
    - Representación del lenguaje: El NLP se basa en diferentes representaciones del lenguaje humano para permitir su procesamiento por parte de las máquinas. Estas representaciones pueden incluir modelos basados en reglas, gramáticas formales, estructuras de datos como árboles sintácticos y grafos semánticos, así como modelos basados en vectores y distribuciones estadísticas de palabras (word embeddings).
    - Análisis morfológico y sintáctico: El procesamiento del lenguaje natural implica el análisis de la estructura y la gramática de las oraciones. Esto implica identificar y etiquetar las partes del discurso (sustantivos, verbos, adjetivos, etc.), analizar las relaciones sintácticas entre las palabras y construir árboles sintácticos que representen la estructura gramatical de una oración.
    - Semántica y comprensión del significado: El NLP busca comprender el significado de las oraciones y textos. Esto implica la aplicación de técnicas y modelos para extraer información semántica, inferir relaciones entre conceptos, resolver la anáfora y la elipsis, y realizar tareas de comprensión y representación del significado.
    - Modelos de aprendizaje automático: El NLP se beneficia ampliamente del aprendizaje automático, especialmente del aprendizaje supervisado y no supervisado. Los modelos de aprendizaje automático, como las redes neuronales, los clasificadores estadísticos y los modelos de lenguaje, se utilizan para tareas como el reconocimiento de entidades, la clasificación de texto, la traducción automática y la generación de lenguaje natural.
    - Recursos lingüísticos: El NLP requiere de recursos lingüísticos, como corpus anotados, léxicos, bases de conocimiento y ontologías, que sirven como datos de entrenamiento y referencia para los modelos de procesamiento del lenguaje. Estos recursos proporcionan información léxica, semántica y gramatical para mejorar la precisión y el rendimiento de los sistemas de NLP.
    - Evaluación y métricas: Para medir la eficacia de los sistemas de procesamiento del lenguaje natural, se utilizan métricas de evaluación específicas. Estas métricas incluyen la precisión, la cobertura, el F1-score, la coherencia del discurso y la fluidez del lenguaje generado.
* **La Red Transformer**

La Red Transformer descrita en el artículo de 2017 Attention is all you need, desarrollado por investigadores de Google, y nacieron inicialmente como una alternativa al problema de la traducción de texto de un idioma a otro. En estas redes la totalidad de la secuencia de entrada es procesada en paralelo por la red, a diferencia de las Redes Recurrentes en donde se procesan uno a uno (es decir de forma serial) los elementos de la secuencia.

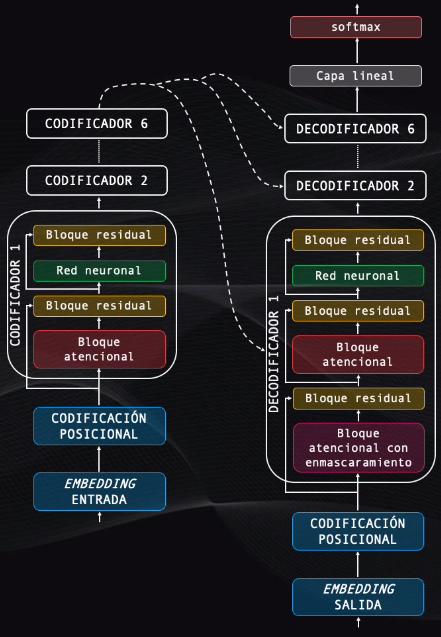


Figura 3: Diagrama de bloques general de una Red Transformer

Fuente: (Sotaquirá, 2020)

Esta secuencia es inicialmente convertida en una representación numérica usando un embedding Después se añade una codificación de posición y los vectores resultantes ingresan a la etapa de codificación, que se encarga de extraer la información más relevante de la secuencia en su idioma original. La salida de esta etapa se conecta al decodificador, que toma esta información para generar secuencialmente el texto traducido al segundo idioma. Aunque tiene muchos elementos realmente es sencillo entender cómo funciona. Veamos entonces en detalle cada módulo de esta red.

* + **El embedding de entrada**

Primero está el bloque embedding, que es simplemente un algoritmo que convierte el texto en una serie de vectores, o tokens, es decir en una representación numérica que puede ser “comprendida” por la red.

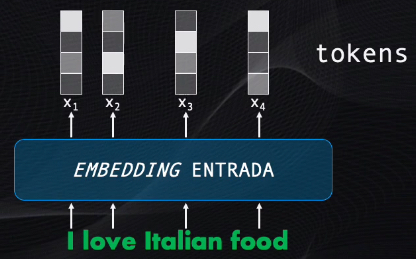


Figura 4: El embedding de entrada de la Red Transformer

Fuente: (Sotaquirá, 2020)

* + **El codificador de posición**

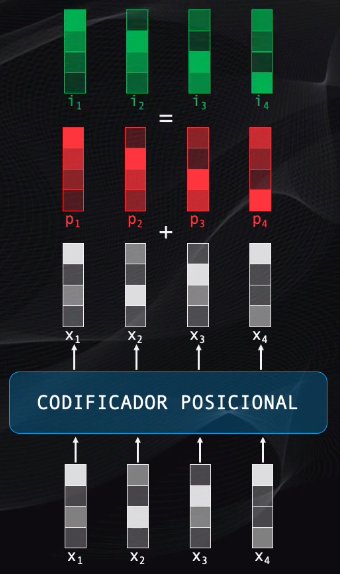


Figura 5: El codificador de posición de la Red Transformer

Fuente: (Sotaquirá, 2020)

Como la secuencia se procesa en paralelo es necesario indicarle a la red el orden en el que se encuentran las palabras dentro del texto. Esto se logra con el codificador de posición. Este codificador genera una serie de vectores que se sumarán a los tokens, y que indican la posición relativa de cada token dentro de la secuencia. Para esto se usan funciones senoidales para las posiciones pares, y cosenoidales para las impares, con lo que cada vector generado tendrá un patrón numérico único con la información de la posición.

* + **Codificación y el bloque atencional**

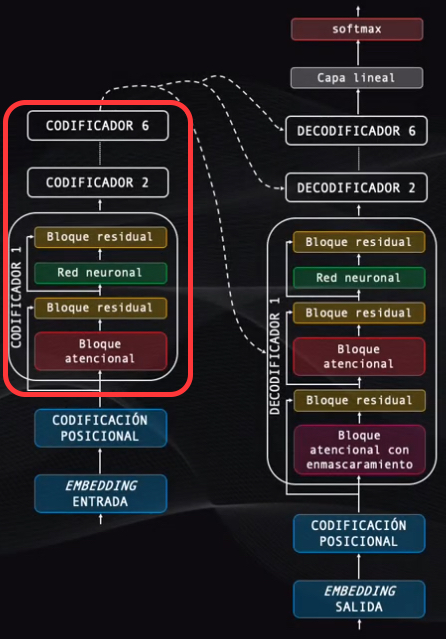


Figura 6: El bloque de codificación de la Red Transformer

Fuente: (Sotaquirá, 2020)

Ahora viene el bloque de codificación, que contiene seis codificadores, todos con una estructura idéntica. Analicemos en detalle uno de estos codificadores. Cada codificador tiene cuatro elementos: un bloque atencional, un bloque de conexión residual, una red neuronal y otro bloque de conexión residual:

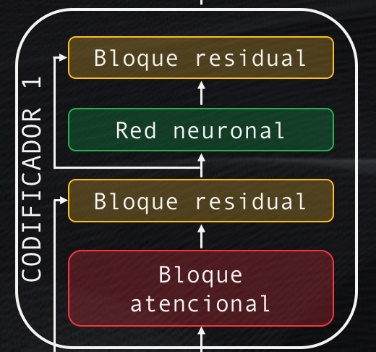


Figura 7: Detalle del primer codificador en la Red Transformer

Fuente: (Sotaquirá, 2020)

Veamos en detalle el bloque atencional, que es tal vez el más importante de toda la red, pues se encarga de analizar la totalidad de la secuencia de entrada (recordemos que la red la procesa de manera simultánea) y de encontrar relaciones entre varias palabras de esta secuencia. Por ejemplo, si el texto de entrada es “I love Italian food”, podemos ver que hay al menos dos posibles asociaciones entre palabras: el verbo “love” y el sujeto (“I”) y el sustantivo “food” asociado al adjetivo “Italian”. Pero además entre estas dos frases (I Love e Italian Food) también hay una asociación:



Figura 8: La relación entre cada palabra de la frase 'I love Italian food'

Fuente: (Sotaquirá, 2020)

Así, lo que hace el bloque atencional es expresar numéricamente las relaciones que existen a diferentes niveles dentro de la secuencia, y luego codifica cada una de ellas con esta información del contexto, indicando así cuáles son los elementos del texto a los que se deben prestar más atención al momento de hacer la traducción. Esta es precisamente la manera como las redes transformer “comprenden” este contexto para codificar adecuadamente cada palabra. Para lograr esto en primer lugar los tokens se llevan simultáneamente a tres pequeñas redes neuronales, entrenadas para calcular los vectores “query”, “key” y “value”. Estos vectores son simplemente tres representaciones alternativas de los tokens originales:

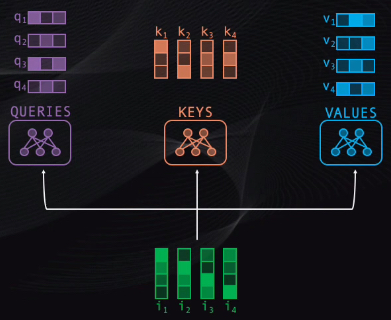


Figura 9: Obtención de los 'queries', 'keys' y 'values'

Fuente: (Sotaquirá, 2020)

Después de esto se toma el query de cada token y se compara con cada uno de los keys existentes. Esta comparación es simplemente una multiplicación de vectores, y con esto se obtendrá un puntaje que mide el grado de asociación entre pares de palabras.

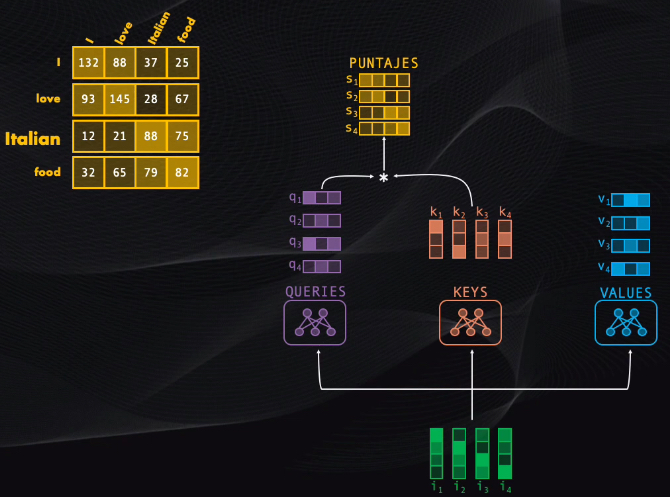


Figura 10: Comparación de 'queries' y 'keys' y obtención de los puntajes Fuente: (Sotaquirá, 2020)

Así, para el caso de la frase que queremos traducir, si analizamos la palabra “Italian” los puntajes obtenidos indican que al codificar este token se le debería prestar más atención a la propia palabra “Italian” seguida por la palabra “food”, y se debería enfocar menos en las palabras “love” y “I”, que tienen los menores puntajes. La idea es ahora usar estos puntajes para ponderar cada uno de los vectores values, indicando así la importancia de cada palabra al momento de la codificación de los tokens. Para poder hacer esto se deben escalar los puntajes, dividiéndolos primero entre el tamaño de cada vector, y luego llevándolos a una función softmax. Esta función permite simplemente representar cada puntaje como una probabilidad entre cero y uno:

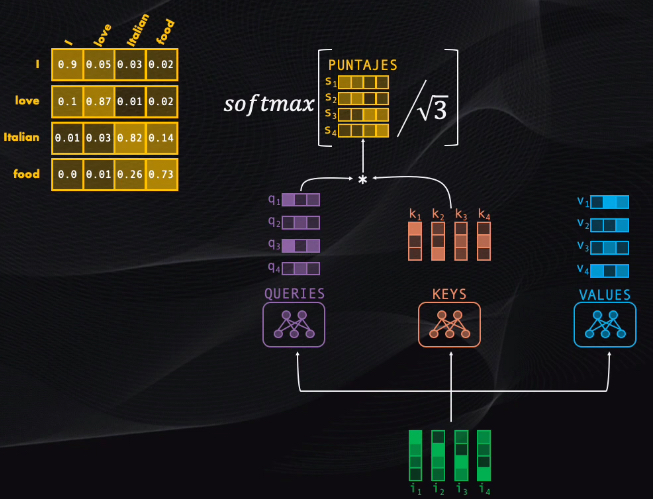


Figura 11: Aplicación de la función 'softmax' a la matriz de puntajes

Fuente: (Sotaquirá, 2020)

Un valor cercano a uno indica que la red debe prestarle más atención a ese token en particular, y un valor cercano a 0 que la palabra no es muy relevante. Finalmente, se debe condensar toda esta información resultante de la comparación en un solo vector por cada token. Así que tomamos la matriz de puntajes que acabamos de obtener y la multiplicamos por la matriz de values: el resultado serán cuatro nuevos tokens, que contendrán la codificación de la información de contexto más relevante para cada palabra de la secuencia:

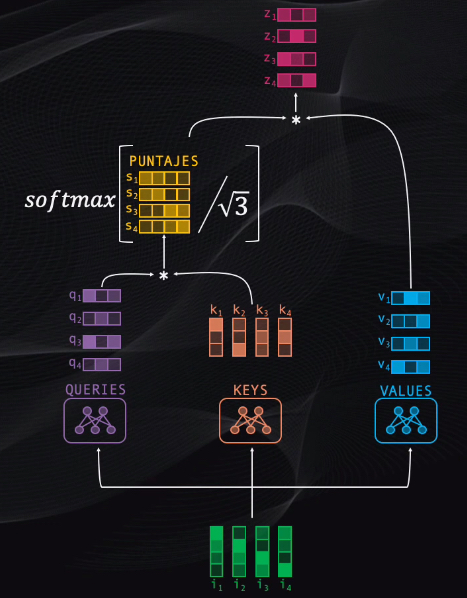


Figura 12: La salida del bloque atencional

Fuente: (Sotaquirá, 2020)

Así que, en resumen, el bloque atencional toma los tokens iniciales y codifica en los tokens resultantes los elementos de la secuencia a los que se debe dar más relevancia. Sin embargo recordemos que en nuestra frase original encontramos, además de asociaciones entre palabras, asociaciones entre frases: para traducir la porción “Italian food” se necesita prestar atención a “I love”. Así que un solo bloque atencional no es suficiente. Al usar múltiples bloques atencionales es posible detectar y codificar asociaciones entre palabras y grupos de palabras a diferentes niveles. Las salidas de estos bloques se combinan en una última red neuronal que condensa toda la información resultante en un único vector para cada token de entrada:

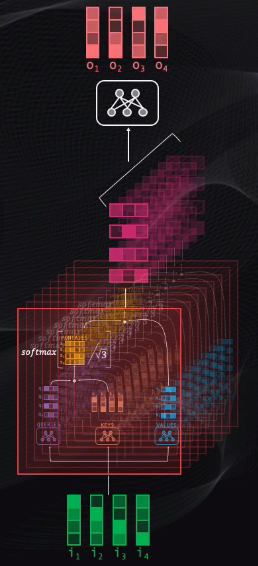


Figura 13: Múltiples bloques atencionales permiten detectar asociaciones entre palabras y grupos de palabras a diferentes niveles

Fuente: (Sotaquirá, 2020)

* + **Codificación y el bloque residual**

A este bloque se llevan tanto la entrada como la salida del bloque atencional, y esto se hace pues la red es muy profunda y si tan solo se enviara la salida la información progresivamente se degradaría y esto dificultaría el entrenamiento y desempeño de la red. Esta etapa toma los dos datos, los suma y luego los normaliza para que tengan la escala adecuada requerida por el siguiente bloque:

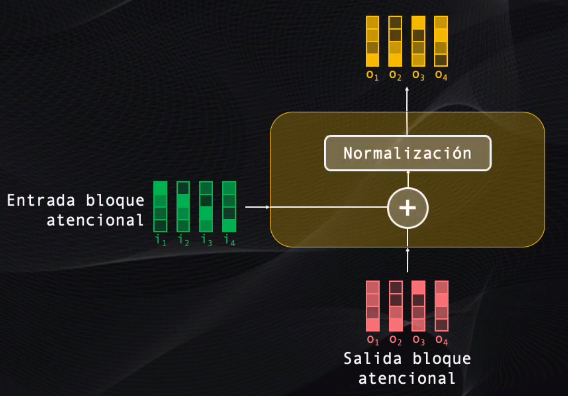


Figura 14: Los elementos del bloque residual

Fuente: (Sotaquirá, 2020)

* + **Codificación: red neuronal y otro bloque residual**

Después de esto tenemos una red neuronal seguida por un bloque residual:

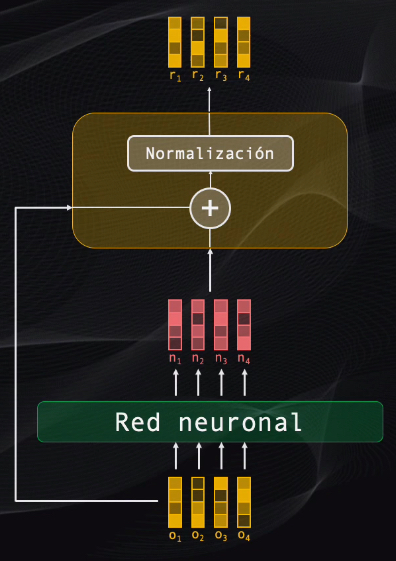


Figura 15: La etapa de salida del codificador: una red neuronal seguida de un bloque residual

Fuente: (Sotaquirá, 2020)

La red neuronal procesa en paralelo todos los vectores de la secuencia, tomando la información atencional de las capas anteriores y consolidándola en una única representación. La entrada y la salida de esta red neuronal son luego llevadas a un bloque residual que tiene exactamente las mismas características del bloque anterior: una suma seguida por una normalización de los datos.

* + **Codificación: resultado final**

Este bloque toma los tokens de entrada, los procesa en paralelo y entrega a la salida una representación que contiene información atencional sobre las diferentes relaciones entre palabras o grupos de palabras de la secuencia, importantes al momento de la traducción:

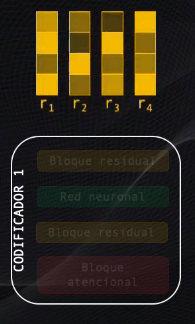


Figura 16: Salida resultante del primer codificador

Fuente: (Sotaquirá, 2020)

Y este proceso se repite para los codificadores restantes, que son idénticos en estructura al codificador que acabamos de analizar.

* + **Decodificación**

Ahora nos enfocamos el segundo bloque importante de la red transformer, que se encarga de hacer la traducción:

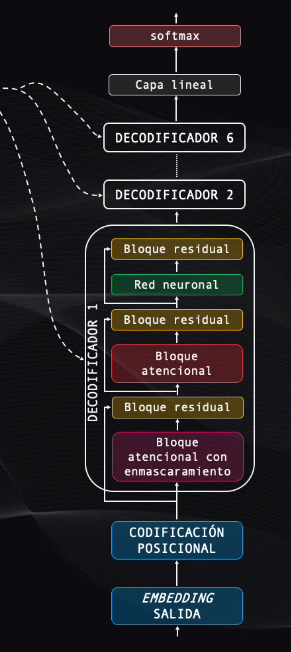


Figura 17: Elementos del decodificador

Fuente: (Sotaquirá, 2020)

En primer lugar, tenemos los bloques de embedding de salida y un codificador posicional, que cumplen exactamente la misma función de los bloques que vimos en la etapa de codificación. Luego viene el decodificador, que es muy similar al bloque de codificación: en total cuenta con 6 decodificadores, cada uno de ellos conectado al codificador, lo que permite conocer la información atencional de la entrada, en el idioma original, para poder realizar la traducción. Cada decodificador es similar a los bloques de codificación que vimos anteriormente: cuenta con bloques atencionales, residuales y redes neuronales que tienen la misma estructura de los codificadores. Sin embargo, tienen un bloque atencional de enmascaramiento y un bloque residual adicionales. Luego viene una capa lineal que, junto con la capa softmax, permite generar una a una las palabras de la secuencia de salida. Veamos entonces cómo funciona paso a paso la decodificación.

* + **Decodificación: bloque atencional con enmascaramiento**

La traducción comienza con la palabra clave “inicio”, la cual es codificada con el embedding y posicionalmente. Al ingresar al primer decodificador es procesada por el bloque atencional de enmascaramiento. Este bloque es prácticamente idéntico al bloque atencional visto anteriormente: codifica la relación entre diferentes elementos de la secuencia de salida, usando los queries, keys y values vistos anteriormente en la Figura 12.

Pero con una diferencia importante: como se está generando cada palabra de manera secuencial, una a una, el decodificador debe prestar atención únicamente a la palabra generada actualmente y a las anteriores, no a las futuras. Por ejemplo, si en la secuencia traducida nos ubicamos en la palabra “la”, el decodificador debería tener acceso a esta palabra y a “amo”, pero no a palabras que aparecerán posteriormente en la secuencia (“comida” e “italiana”):



Figura 18: Al momento del enmascaramiento no se prestará atención a las palabras futuras en la secuencia: 'comida' e 'italiana'

Fuente: (Sotaquirá, 2020)

Así que para evitar esto se agrega un bloque que enmascara, es decir que simplemente hace cero, las palabras a las que durante la decodificación no se debe prestar atención:



Figura 19: El resultado del enmascaramiento

Fuente: (Sotaquirá, 2020)

Al igual que con el codificador, en este caso también se emplean múltiples bloques atencionales para detectar relaciones a diferentes niveles.

* + **Decodificación: el bloque atencional**

Todos los bloques residuales, así como la red neuronal de este decodificador funcionan de forma idéntica a como ocurría en los codificadores. Así que nos enfocaremos ahora en el bloque atencional que en este caso tiene la misma estructura, pero un funcionamiento ligeramente diferente al del codificador. Este bloque enfoca su atención tanto en la secuencia original como en la de salida y para ello toma la salida del codificador y las lleva a las redes “queries” y “keys”, mientras que el nodo “values” usa como entrada el dato proveniente del bloque residual anterior:

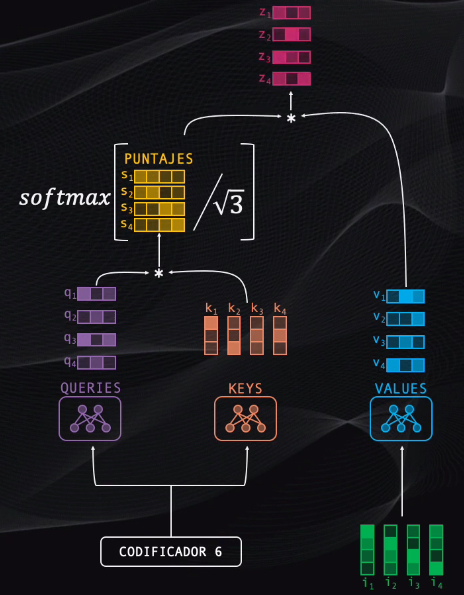


Figura 20: El bloque atencional del decodificador toma como entrada la información generada por el codificador de la etapa anterior Fuente: (Sotaquirá, 2020)

Es de esta manera como el codificador le indica al decodificador a qué elementos debe prestar más atención al momento de generar la secuencia de salida. De nuevo, se usan múltiples bloques atencionales de manera simultánea para codificar asociaciones a diferentes niveles.

* + **Decodificación: múltiples decodificadores y etapas de salida**

Este bloque se replica un total de seis veces, y al final genera un vector con cantidades numéricas, y lo único que falta es convertirlo, en una palabra:

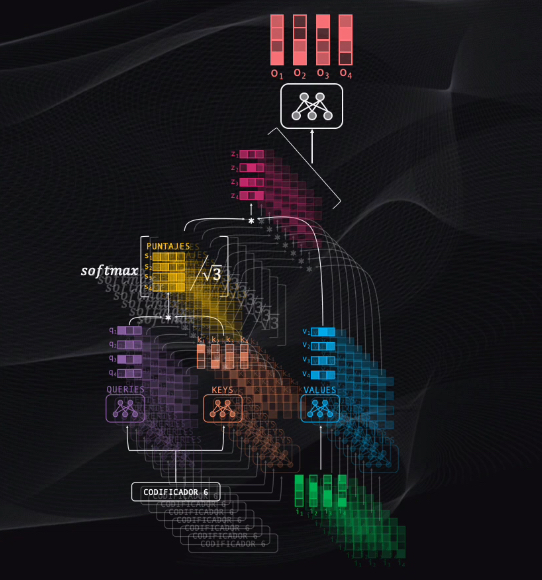


Figura 21: En la decodificación también se requieren múltiples decodificadores para encontrar relaciones entre palabras y grupos de palabras a diferentes niveles

Fuente: (Sotaquirá, 2020)

Para eso se usa en primer lugar la capa lineal, que es simplemente una red neuronal que toma el vector producido por el decodificador y lo transforma en un vector mucho más grande. Por ejemplo, si el traductor aprende 10000 palabras (es decir el tamaño del vocabulario), entonces el vector de salida de la capa lineal tendrá precisamente 10000 elementos. La capa softmax toma cada elemento de este vector y lo convierte en una probabilidad, todas con valores positivos entre 0 y 1. La posición con la probabilidad más alta será seleccionada y la palabra asociada con dicha posición será precisamente la salida del modelo en ese instante de tiempo:

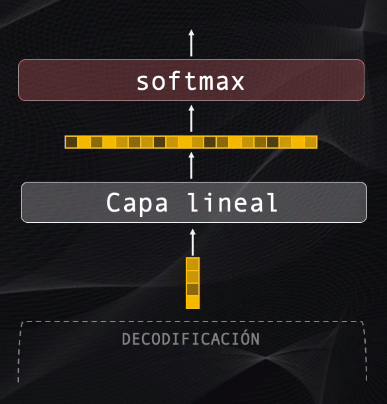


Figura 22: Capas de salida del bloque de decodificación

Fuente: (Sotaquirá, 2020)

Y el proceso se repite hasta generar la totalidad de la secuencia de salida.

* **Modelos Generadores de Texto**

Los Modelos Generadores de Texto son una clase de modelos de Inteligencia Artificial que se utilizan para generar texto automáticamente. Estos modelos son capaces de generar secuencias de palabras coherentes y relevantes que se asemejan al lenguaje humano.

Los Modelos Generadores de Texto incluyen los siguientes elementos clave:

* + **Aprendizaje Automático:** Los Modelos Generadores de Texto se basan en técnicas de aprendizaje automático, especialmente en el aprendizaje profundo. Utilizan redes neuronales recurrentes o transformadores para aprender patrones y estructuras del lenguaje a partir de grandes conjuntos de datos de texto.
  + **Modelo de Lenguaje:** Los Modelos Generadores de Texto se construyen sobre la base de los modelos de lenguaje, que son modelos estadísticos o de aprendizaje automático que asignan probabilidades a secuencias de palabras. Estos modelos capturan la estructura y las regularidades del lenguaje, lo que permite al modelo generador producir secuencias de palabras coherentes.
  + **Codificación de Entrada:** En muchos casos, los Modelos Generadores de Texto utilizan una codificación de entrada que convierte las palabras en vectores numéricos. Esta codificación permite que el modelo trabaje con representaciones numéricas del lenguaje y facilite el procesamiento por parte de las redes neuronales.
  + **Decodificación:** Durante la generación de texto, el modelo utiliza técnicas de decodificación para seleccionar y generar palabras de manera secuencial. Estas técnicas pueden incluir métodos de muestreo estocástico, como el muestreo softmax o el muestreo de diversidad, que permiten al modelo seleccionar palabras basándose en sus probabilidades asignadas.
  + **Entrenamiento Supervisado:** Los Modelos Generadores de Texto a menudo se entrenan utilizando enfoques de aprendizaje supervisado, donde se les proporciona un conjunto de datos de texto de entrenamiento que contiene ejemplos de entrada y las secuencias de texto objetivo que se espera que el modelo genere. Estos ejemplos se utilizan para aprender los patrones y las relaciones en los datos y ajustar los parámetros del modelo.
  + **Transferencia de Estilo:** Algunos Modelos Generadores de Texto tienen la capacidad de generar texto en un estilo específico, como imitar el estilo de un autor famoso o generar texto poético. Estos modelos pueden ser entrenados utilizando técnicas de transferencia de estilo, donde se les proporciona un conjunto de datos de entrenamiento que contiene ejemplos del estilo deseado.
  + **Evaluación:** La evaluación de los Modelos Generadores de Texto es un desafío importante. Se utilizan métricas y técnicas específicas para evaluar la calidad y la coherencia del texto generado, como la evaluación humana, métricas automáticas como el BLEU (Bilingual Evaluation Understudy) y el ROUGE (Recall-Oriented Understudy for Gisting Evaluation), y la comparación con modelos de referencia o textos de alta calidad.
  + **Modelos Generadores de código:** Los Modelos Generadores de código son modelos de Inteligencia Artificial diseñados para generar código fuente automáticamente. Estos modelos pueden generar segmentos de código, funciones completas o incluso programas completos que sean sintácticamente correctos y cumplan con los requisitos especificados. Estos
* **Desarrollo de Software**

El desarrollo de software se refiere al proceso de crear, diseñar, programar, probar y mantener programas de computadora, aplicaciones móviles o sistemas informáticos. Es un proceso sistemático que involucra diversas etapas y actividades para construir software funcional y de calidad.

El desarrollo de software implica trabajar en conjunto con profesionales de diferentes disciplinas, como programadores, analistas de sistemas, diseñadores de interfaces, testers y gerentes de proyectos, entre otros. Estos profesionales colaboran para convertir los requisitos y necesidades de los usuarios en soluciones de software.

* + **Desarrollo de Software con Modelos de Inteligencia Artificial**

El desarrollo de software con modelos de Inteligencia Artificial (IA) es una disciplina que combina los principios y prácticas del desarrollo de software con el uso de modelos y algoritmos de IA para crear soluciones más inteligentes y automatizadas.

El marco teórico del desarrollo de software con modelos de IA incluye los siguientes elementos clave:

* Integración de modelos de IA: En este enfoque, los modelos de IA se utilizan como componentes dentro del software desarrollado. Estos modelos pueden incluir algoritmos de aprendizaje automático, redes neuronales, algoritmos de procesamiento del lenguaje natural (NLP) u otros modelos de IA específicos. Los modelos de IA se entrenan con datos relevantes para la tarea específica que el software debe abordar.
* Análisis de datos y preparación de datos: Para desarrollar software con modelos de IA, es necesario realizar un análisis exhaustivo de los datos que se utilizarán para entrenar y alimentar los modelos de IA. Esto implica la identificación de los conjuntos de datos apropiados, la limpieza y el preprocesamiento de los datos, así como la selección de las características relevantes para la tarea en cuestión.
* Entrenamiento de modelos de IA: Los modelos de IA utilizados en el desarrollo de software deben ser entrenados con los datos relevantes para aprender patrones y relaciones. Esto implica la elección del algoritmo de entrenamiento adecuado, la definición de la arquitectura del modelo y la optimización de los parámetros para maximizar el rendimiento del modelo.
* Integración de modelos de IA en el flujo de trabajo: Los modelos de IA se incorporan en el flujo de trabajo del software desarrollado para realizar tareas específicas. Esto puede incluir la clasificación de datos, el procesamiento del lenguaje natural, la generación de contenido, la detección de anomalías, la toma de decisiones automatizada, entre otras aplicaciones. Los modelos de IA pueden ser utilizados como módulos independientes o integrados en sistemas más complejos.
* Validación y evaluación: Al igual que en el desarrollo de software convencional, la validación y evaluación son cruciales en el desarrollo de software con modelos de IA. Se utilizan métricas específicas para evaluar el rendimiento de los modelos de IA, como la precisión, la sensibilidad, la especificidad, el F1-score, entre otros. Además, es importante realizar pruebas rigurosas para asegurarse de que el software funcione correctamente y cumpla con los requisitos establecidos.
* Actualización y mejora continua: Los modelos de IA en el desarrollo de software suelen ser iterativos y requieren actualizaciones y mejoras continuas. Esto puede implicar la recopilación y el análisis de nuevos datos para reentrenar los modelos de IA, la adaptación a cambios en los requisitos y la optimización de los algoritmos para mejorar el rendimiento.
* Aspectos éticos y legales: El desarrollo de software con modelos de IA plantea consideraciones éticas y legales. Es importante asegurarse de que los datos utilizados sean éticamente recopilados y utilizados de manera adecuada. Además, se deben considerar aspectos legales, como la privacidad de los datos y el cumplimiento de las regulaciones aplicables.