**UNIVERSIDAD CATÓLICA BOLIVIANA**

**“SAN PABLO”**

FACULTAD DE POSTGRADO

**DEPARTAMENTO DE INGENIERÍA DE SISTEMAS**



**PERFIL PROYECTO**

“DESARROLLO DE UN CHATBOT MEDIANTE EL PROCESAMIENTO DE LENGUAJE NATURAL Y APRENDIZAJE PROFUNDO PARA LA NIVELACIÓN DE ESTUDIANTES PRE UNIVERSITARIOS”

**PRESENTADO POR:**

José Estensoro (josee906@gmail.com)

Roger Patón (oviroger@gmail.com)

**Docente:**

Jhesser Ghuzman

**LA PAZ – BOLIVIA**

**2023**

# Introducción

La creciente cantidad de datos ha ocasionado, al mismo tiempo, un crecimiento y una evolución en los algoritmos de inteligencia artificial. La inteligencia artificial trata de imitar la inteligencia humana. Se la utiliza en una gran variedad de aplicaciones. Está en constante evolución y se espera que tenga un impacto significativo en la sociedad y la economía, pero también plantea importantes cuestiones éticas y de privacidad. Una de las aplicaciones más utilizadas de las inteligencias artificiales son los ChatBots.

Los chatbots son programas de inteligencia artificial diseñados para imitar conversaciones humanas y se utilizan en aplicaciones de mensajería, páginas web y móviles. Utilizan bases de datos de respuestas predefinidas y procesamiento de lenguaje natural para comprender y responder a los usuarios. Algunos chatbots utilizan aprendizaje profundo para mejorar su comprensión del lenguaje humano y proporcionar respuestas más precisas. Los chatbots están evolucionando para mejorar la atención al cliente, pero todavía hay desafíos en cuanto a la coherencia y la privacidad de las respuestas.

# Antecedentes

Los proyectos de Bots con el uso de NLP (natural lenguage processing) estuvieron evolucionando de gran manera en los últimos años, existen varios proyectos que realizan esta tarea. En los últimos meses se ha estado discutiendo la aparición del Chat-GPT.

Chat-GPT es un proyecto de la empresa OpenAI. Es un modelo de Deep Learning, más específicamente es una red neuronal transformer basada en el modelo Transformador pre entrenado generativo (OpenAI, 2022). Chat-GPT tiene las siguientes características:

* Comprensión del lenguaje natural
* Generación de texto
* Aprendizaje automático.

El BERT (Representación de Codificador Bidireccional de Transformadores) es una técnica basada en redes neuronales para el pre-entrenamiento del procesamiento del lenguaje natural (PLN) desarrollada por Google (Google, 2023).

El BERT fue creado y publicado en 2018 por Jacob Devlin y sus compañeros en Google. Google está aprovechando el BERT para comprender mejor las búsquedas de los usuarios (Google, 2023).

Quizlet: Es una plataforma de estudio en línea que ofrece juegos interactivos y flashcards para ayudar a los estudiantes a aprender y memorizar información. También tiene un chatbot de preguntas y respuestas que los estudiantes pueden usar para hacer preguntas sobre cualquier tema.()

Existen varias plataformas parecidas a Quizlet como:

* Brainly
* Socratic
* IBM Watson Assistant

# Identificación y formulación del problema de investigación

# Planteamiento del problema

En la actualidad, es común observar que muchos estudiantes que ingresan a la universidad presentan dificultades en su nivel de conocimientos en comparación con otros estudiantes. Esto se debe, en gran parte, a las deficiencias en la enseñanza de ciertos temas en algunos colegios, lo que resulta en una falta de preparación académica adecuada para enfrentar los retos de la educación superior.

Estas deficiencias en la enseñanza pueden ser el resultado de varios factores, incluyendo la falta de recursos y materiales didácticos adecuados, la falta de capacitación de los docentes y la ausencia de una política educativa que promueva la excelencia académica. Estas dificultades pueden ser especialmente graves en áreas como las matemáticas y las ciencias, que son fundamentales para muchas carreras universitarias.

# Formulación del problema

¿Se puede desarrollar un bot utilizando aprendizaje profundo, “natural language processing” y otros algoritmos y métodos de Machine Learning para que los jóvenes que están entrando a la universidad puedan nivelarse y aprendan de una manera más fácil y divertida?

# Objetivos

# Objetivo general

Desarrollar un Bot de preguntas y respuestas con “natural language processing” y aprendizaje profundo para que los y las jóvenes que están ingresando a las universidades puedan nivelarse y aprender ciertos temas de una manera fácil y divertida.

# Objetivos específicos

* Buscar y descargar un dataset de preguntas y respuestas para que el bot pueda ser entrenado de se dataset.
* Realizar la limpieza de los datos del dataset para que no existan outliers y no se distorsione el entrenamiento.
* Desarrollar un algoritmo de IA con NLP para que pueda ser usado como el algoritmo de IA.
* Validar los datos y el entrenamiento del bot para que se compruebe su correcto funcionamiento.

# Alcances y límites

# Alcances

* Los datos a utilizar solo serán datos de prueba.
* Crear una herramienta única e innovadora.
* Se utilizará un dataset público.

# Límites

* El Bot solo llegará a prototipado no se realizará la implementación
* El Bot podría tener dificultades en entender las preguntas más complejas.
* El Bot tendrá recursos a nivel de “Hardware” limitados.
* No se desarrollará ninguna técnica de NLP nueva se utilizara técnicas ya conocidas.

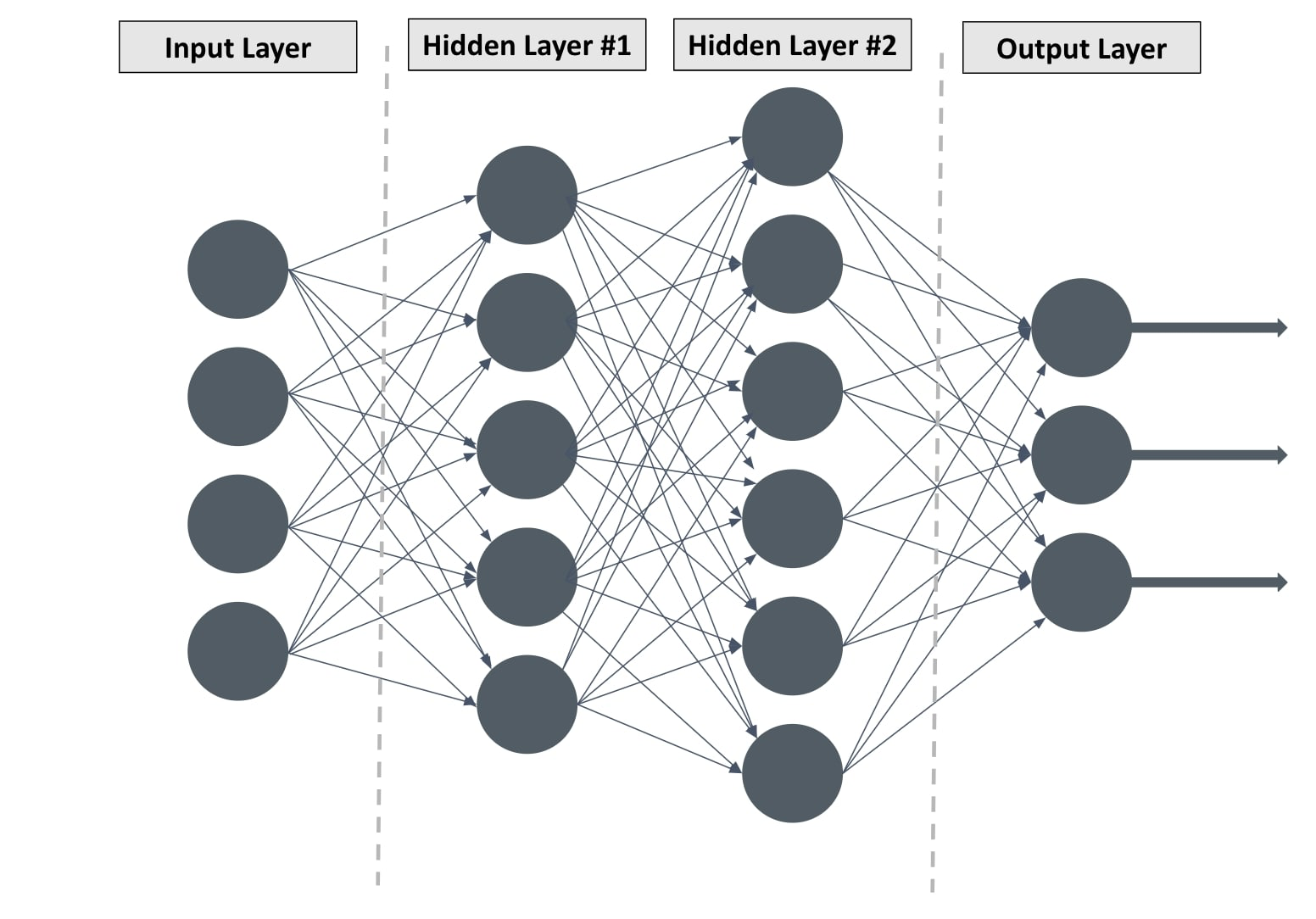
# Propuesta solución

# solución 1

# Arquitectura del Modelo

El modelo que se ha definido para el proyecto es una red neuronal de tipo feedforward, esta red neuronal es de un tipo común de arquitectura utilizada para la clasificación y la regresión.

La entrada se alimenta a través de cinco capas ocultas completamente conectadas, cada una con 100 neuronas y una función de activación ReLU (Rectified Linear Unit). Cada capa oculta toma como entrada la salida de la capa anterior y produce una nueva salida que se convierte en la entrada de la siguiente capa oculta. La función de activación ReLU se utiliza para introducir no linealidad en la red y permitir que la red aprenda funciones más complejas. Las capas ocultas se denominan "totalmente conectadas" porque cada neurona en una capa se conecta con todas las neuronas en la siguiente capa.



La última capa utilizada en la red es una capa de salida completamente conectada con una función de activación softmax. La función softmax se utiliza para convertir las salidas de las neuronas en la capa de salida en una distribución de probabilidad sobre las posibles clases de salida.

Finalmente, el modelo se define utilizando la clase DNN de TFLearn, que toma la arquitectura de la red como argumento y proporciona métodos para entrenar la red y hacer predicciones. El modelo se entrena utilizando un algoritmo de optimización de descenso de gradiente estocástico (SGD) y una función de pérdida de entropía cruzada categórica.

# El dataset

El dataset utilizado para este proyecto, llamado WikiQA, es un conjunto de datos de acceso público disponible en la plataforma Kaggle. Este conjunto de datos está diseñado para ser utilizado en la tarea de recuperación de información

El corpus de WikiQA es un nuevo conjunto de pares de preguntas y oraciones disponible públicamente, recopilados y anotados para la investigación sobre la respuesta a preguntas de dominio abierto. Con el fin de reflejar la verdadera necesidad de información de los usuarios generales, utilizamos los registros de consultas de Bing como fuente de preguntas. Cada pregunta está vinculada a una página de Wikipedia que potencialmente tiene la respuesta. Debido a que la sección de resumen de una página de Wikipedia proporciona la información básica y, por lo general, más importante sobre el tema, utilizamos frases en esta sección como respuestas del candidato. Con la ayuda del crowdsourcing, incluimos 3.047 preguntas y 29.258 oraciones en el conjunto de datos, donde 1.473 oraciones fueron etiquetadas como oraciones de respuesta a sus preguntas correspondientes.

# pre procesamiento de datos

Primero, se inicializan cuatro listas vacías para preparar los datos en estas listas. Luego, se itera sobre cada fila del dataframe importado y se tokeniza. Las palabras se agregan a una de las listas vacías. la lista tokenizada se agrega a otra lista. La etiqueta del documento correspondiente se agrega a una 3 lista.

A continuación, se procesan las palabras en la lista. Se convierten en minúsculas y se aplica un algoritmo de stemming utilizando. Las palabras procesadas se agregan a la 4 lista y se eliminan los signos de interrogación.

Se preparan los datos de entrenamiento y salida para el modelo de aprendizaje automático.

Se itera sobre cada tokenizado. Para cada tokenizado, se crea una bolsa de palabras (o "bag-of-words") que indica qué palabras de las lista 1 aparecen en el tokenizado. Si una palabra aparece en el tokenizado, se agrega 1 a la bolsa, de lo contrario, se agrega 0.

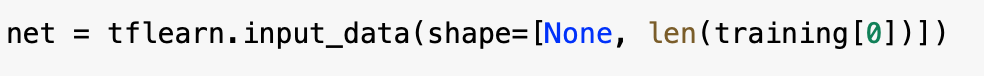
Se crea una fila de salida correspondiente a la bolsa de palabras actual. La fila de salida es una lista de ceros de longitud igual a la cantidad de etiquetas, excepto que se establece el valor en 1 para la etiqueta correspondiente a la pregunta actual. La fila de salida se agrega a la salida .

Finalmente, el código produce dos listas: training y output, que se pueden usar para entrenar un modelo de aprendizaje automático para clasificar preguntas en función de su título de documento correspondiente.

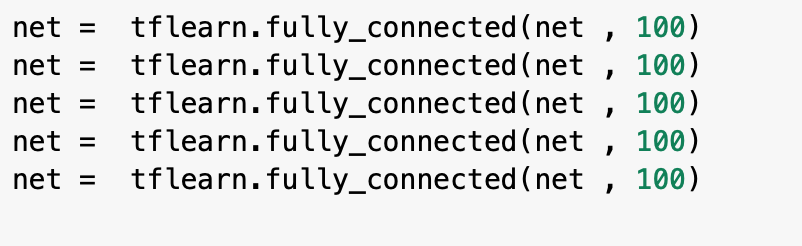
# El modelo

Se utiliza la biblioteca de aprendizaje profundo "TFLearn" para definir y entrenar una red neuronal para un problema de clasificación.

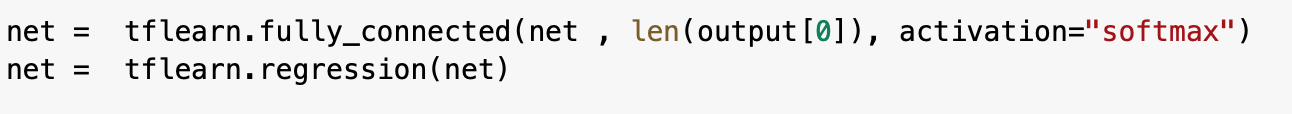
La primera línea de código define la entrada de datos de la red, especificando el tamaño de la entrada. El parámetro "None" indica que el tamaño de lote de entrada puede variar.



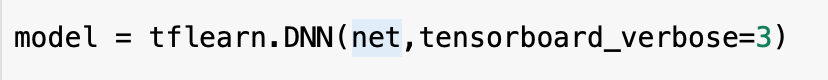
Las siguientes líneas definen capas completamente conectadas (fully-connected) en la red, donde cada neurona de una capa está conectada a todas las neuronas de la capa anterior. La función "fully\_connected" de TFLearn se utiliza para definir estas capas. En este caso, se definen cinco capas ocultas de 100 neuronas cada una.



La última capa utiliza una función de activación softmax, que es adecuada para la clasificación multiclase, y tiene un número de neuronas igual al número de clases en el conjunto de datos de salida. La función "regression" se utiliza para definir el tipo de salida de la red como una regresión.



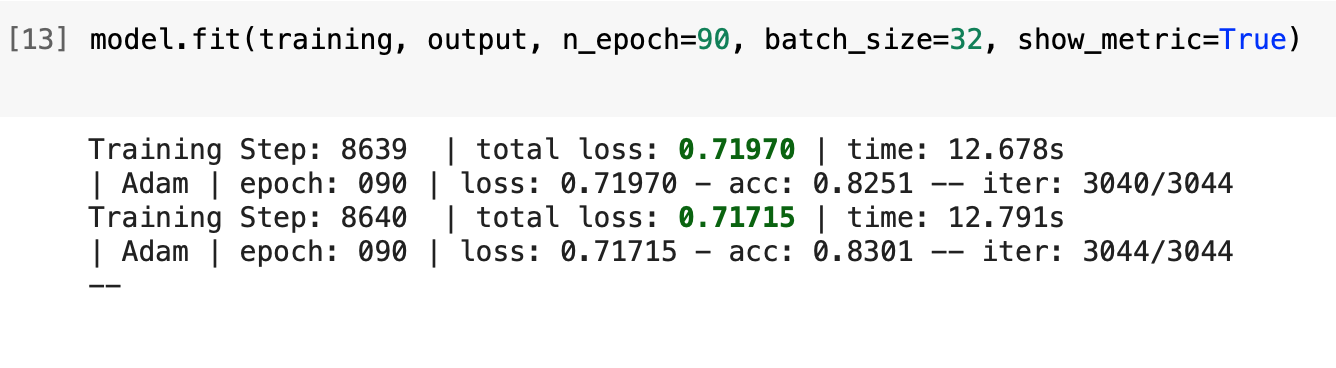
Finalmente, se crea una instancia del modelo de red neuronal utilizando la clase "DNN" de TFLearn, y se utiliza para entrenar la red. En este código no se incluye el proceso de entrenamiento, por lo que se omite.



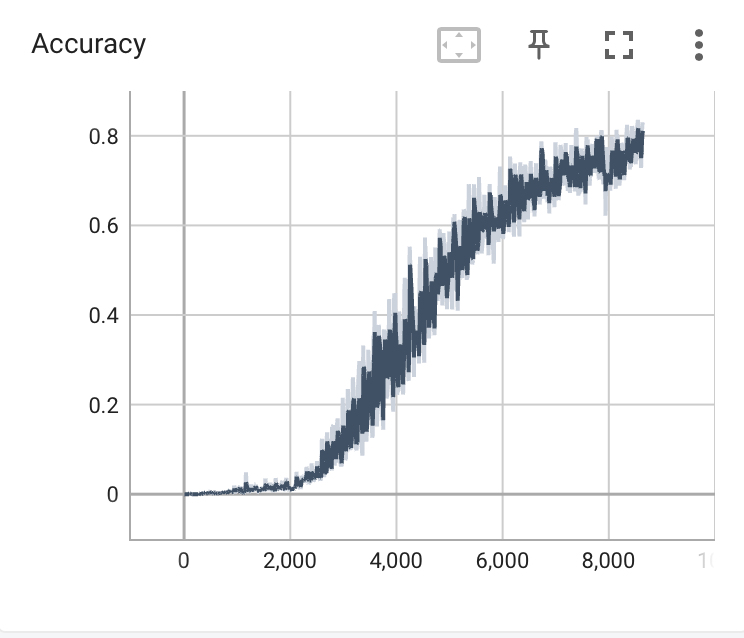
# Entrenamiento

En el entrenamiento, se realizaron 8640 pasos de entrenamiento y en la última época (epoch 090), la pérdida total del modelo fue de 0.71970 y la precisión alcanzada en el conjunto de entrenamiento fue del 82.51%. Esto significa que el modelo ha aprendido a predecir las etiquetas de salida con una precisión del 82.51% en el conjunto de datos de entrenamiento.

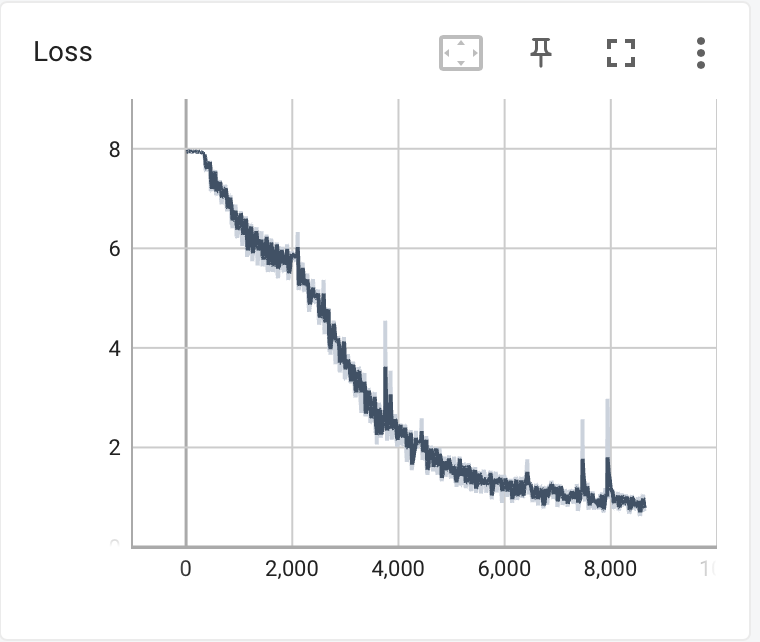
El argumento “show\_metric=True” se utiliza para mostrar la precisión de la red en cada paso de entrenamiento en la consola, lo que puede ser útil para monitorear el progreso del entrenamiento.



En la siguiente grafica podemos apreciar cómo evoluciona el “Accuracy” mientras el modelo va entrenando

****

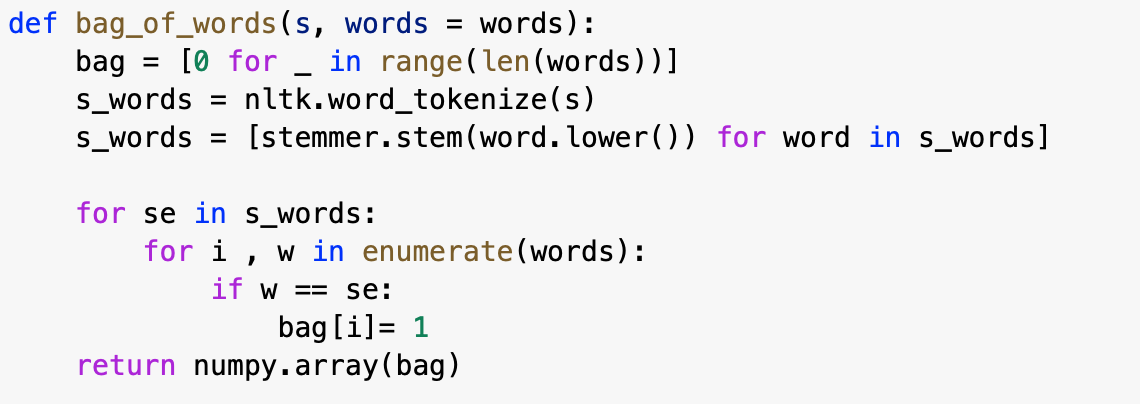
Y finalmente en la siguiente grafica podemos apreciar la disminución de la perdida mientras el modelo se va entrenando.

****

# pruebas del modelo

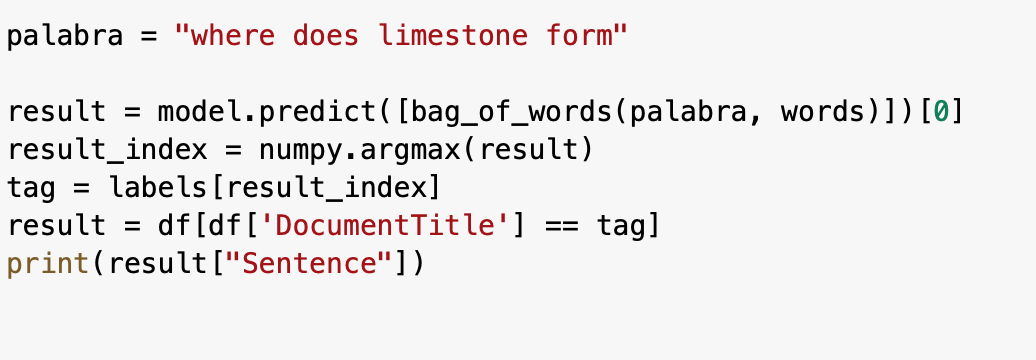
Esta sección de código utiliza la función bag\_of\_words() para convertir una oración en una bolsa de palabras, que se utiliza como entrada para el modelo entrenado.

Primero, la función “bag\_of\_words()” tokeniza la oración utilizando “nltk.word\_tokenize()”, lo que significa que la oración se divide en palabras individuales. Luego, se realiza una normalización de cada palabra utilizando stemmer.stem() y se verifica si cada palabra está presente en el conjunto de palabras words utilizadas en el entrenamiento del modelo. Si la palabra está presente, se establece la posición correspondiente en la bolsa de palabras como 1, de lo contrario se mantiene como 0.



A continuación, la función model.predict() utiliza la bolsa de palabras como entrada y devuelve una lista de valores de salida para cada clase. La función numpy.argmax() se utiliza para obtener el índice de la clase con el valor de salida más alto, que se utiliza para buscar el tag (etiqueta) correspondiente en la lista labels.

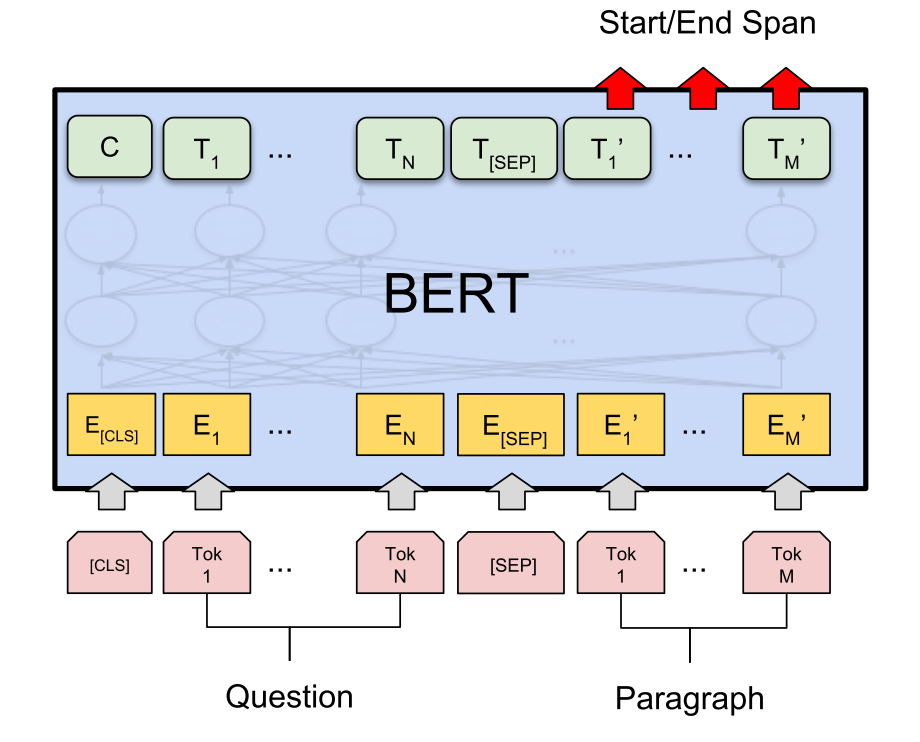
Finalmente, se busca en un DataFrame la fila que corresponde a la etiqueta identificada y se imprime una oración del DataFrame que se ajuste a la etiqueta. Esto se hace para proporcionar una respuesta a la pregunta original basada en los datos de entrenamiento utilizados para entrenar el modelo.

****

# Solución propuesta 2

# Arquitectura del Modelo

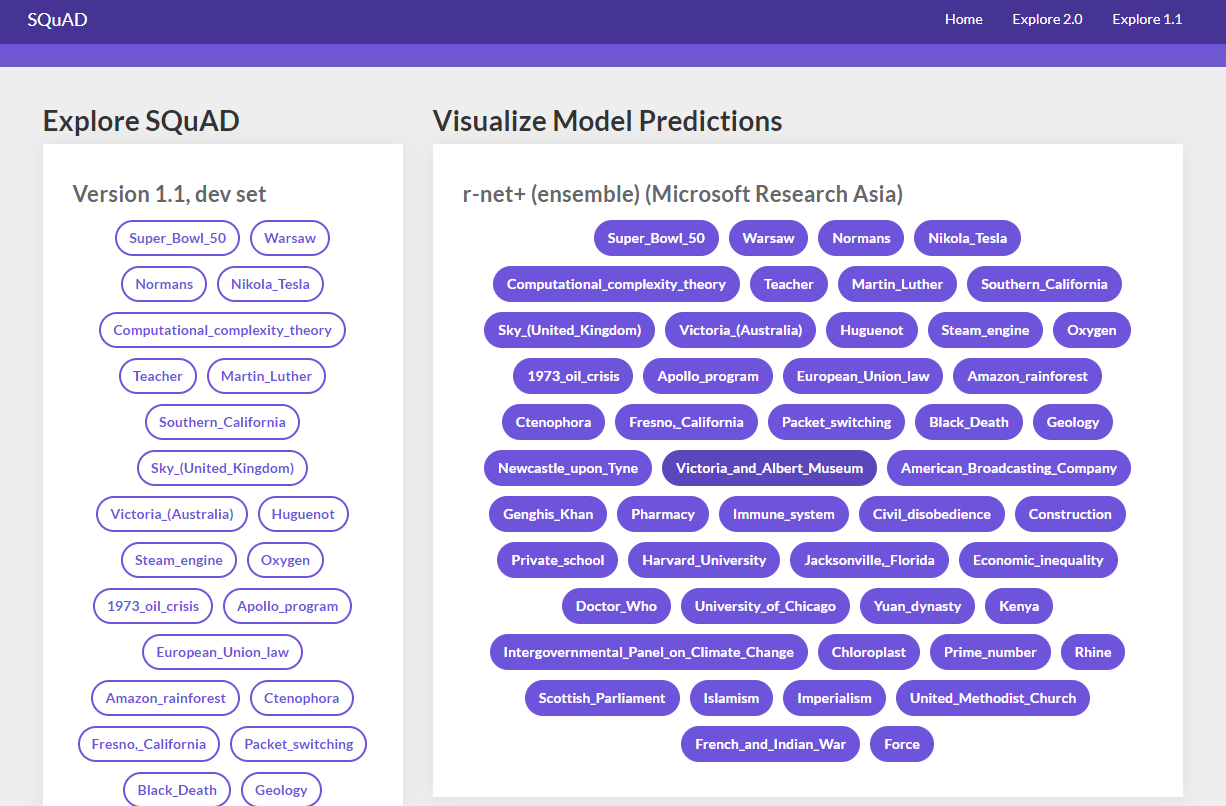
Para esto se tomará un modelo “BERT BASE” pre-entrenado, y se agregará algunas capas adicionales, además se afinará para que aprenda a encontrar la respuesta dados un contexto y una pregunta como entrada. Debido a la complejidad del modelo, y la imposibilidad de entrenarlo con los recursos proporcionados por Google Colab, usaremos la librería Tokenizer y el modelo BERT pre-entrenado mencionado anteriormente.

**Figura 1**  
*Arquitectura del modelo propuesto  
*  
*Cómo se usa BERT para resolver tareas de preguntas y respuestas. (Fuente de la imagen: Devlin et al., 2018)*

# El set de datos

Usaremos el set SQuAD v1.1 (Stanford Question Answering Dataset), que contiene más de 100.000 pares de preguntas-respuestas. Cada ejemplo de entrenamiento contiene las siguientes variables:

* **context**: el texto completo (contexto)
* **question**: una pregunta acerca del contenido del contexto
* **answer\_text**: el texto dentro del contexto que responde la pregunta
* **start\_char\_idx**: el caracter dentro del contexto en donde inicia la respuesta

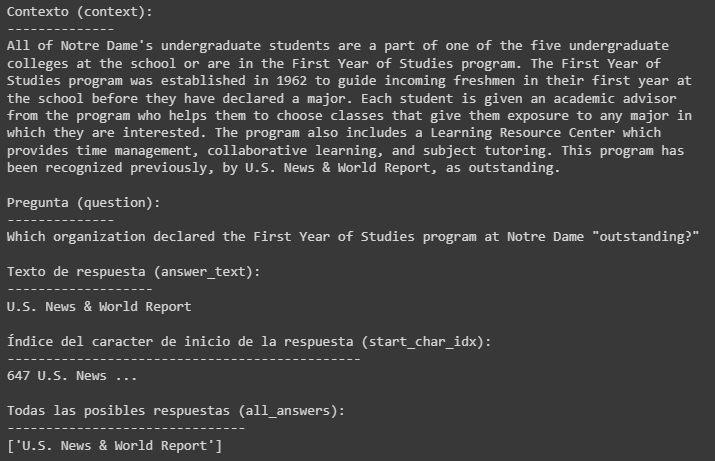
**Figura 2**  
 Set de datos SQuAD **** *Fuente****:*** [*https://rajpurkar.github.io/SQuAD-explorer/explore/1.1/dev/*](https://rajpurkar.github.io/SQuAD-explorer/explore/1.1/dev/)

* **all\_answers**: todos los posibles textos de respuesta (en caso de que la respuesta aparezca varias veces dentro del contexto)

El set de datos se divide en dos partes: **raw\_train\_dataset** (set de entrenamiento) y **raw\_test\_dataset** (set de prueba).

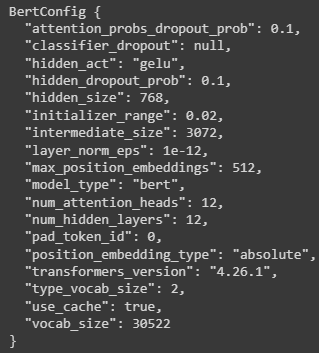
# Data set de entrenamiento

Para entender el dataset de entrenamiento veamos un ejemplo del mismo:

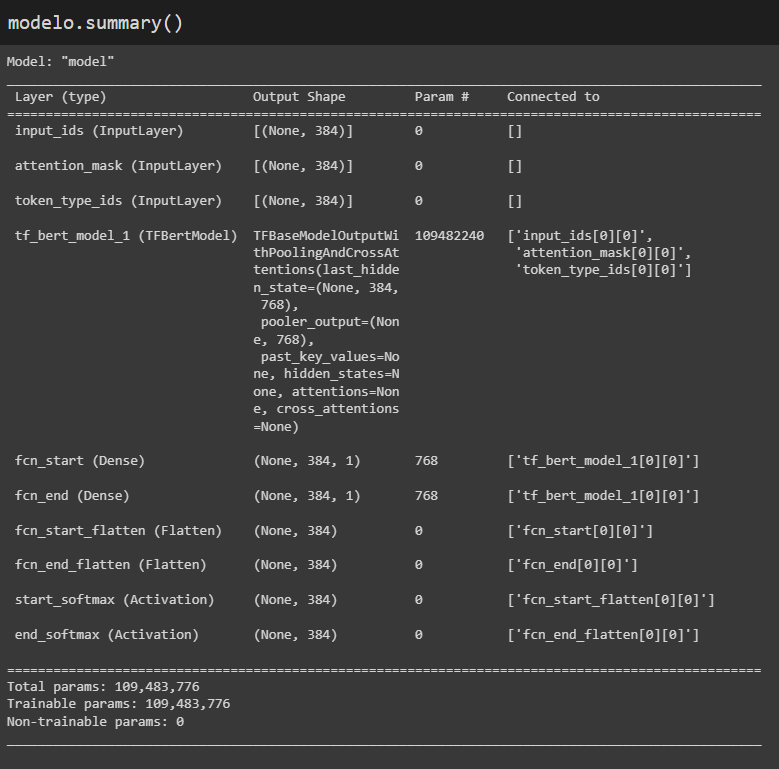


# Creación del modelo

Usando la librería transformers, implementaremos la arquitectura descrita anteriormente:



Luego de incluir las capas de entrada y salidas el modelo final tiene la siguiente estructura y parámetros:

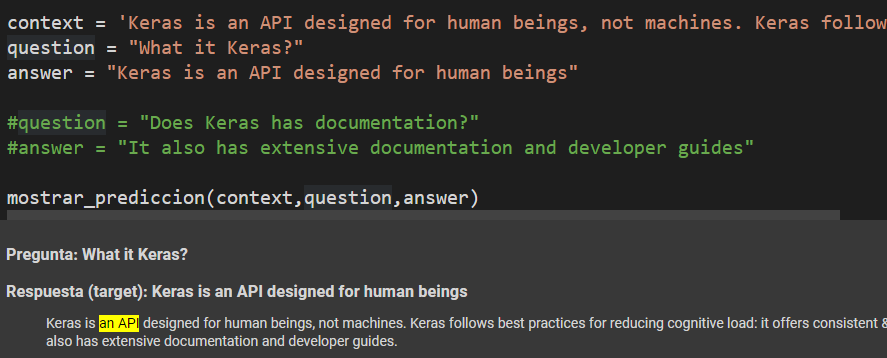


# Entrenamiento del modelo

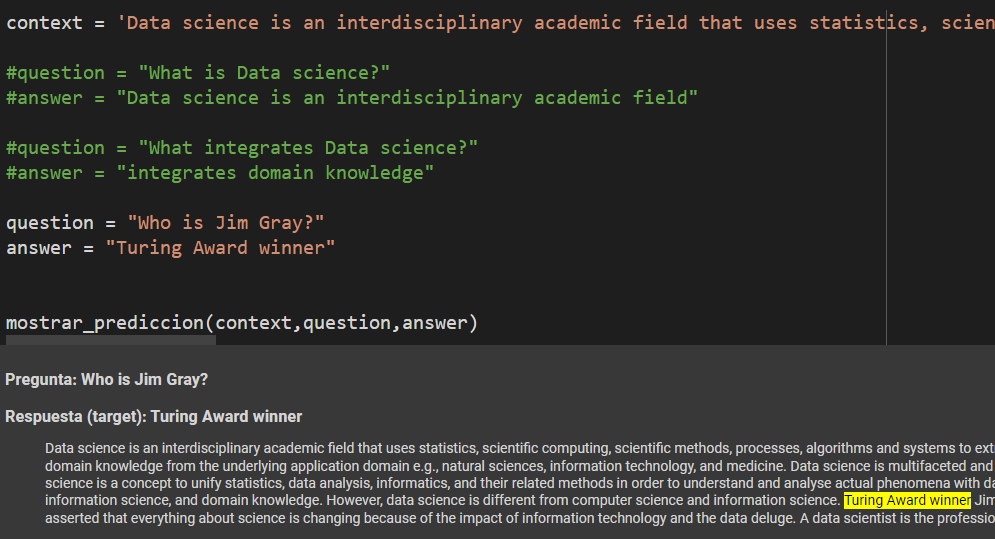
Para el entrenamiento usaremos el método fit que nos proporciona Keras, y nuevamente al tamaño del set de entrenamiento, a la complejidad del modelo (sin embargo, es pre-entrenado), solo se realizarán un total de 3 iteraciones, y se hará uso del entorno de ejecución “TPU” proporcionado por Google Colab. Para cada iteración se requirió alrededor de 10 minutos.

# Pruebas al modelo entrenado

Para probar el modelo utilizaremos algunos contextos diferentes de los datasets de entrenamiento y validación. En el primer caso tomaremos una descripción de Keras y luego realizaremos un par de preguntas para verificar si el modelo logra responderlas.



En un segundo ejemplo utilizaremos un contexto relacionado con Data Science y nuevamente realizaremos algunas preguntas:



# Conclusiones y recomendaciones

# Conclusiones

El Bot de la solucion 1 una vez entrenado puede responder adecudamente a las preguntas realizadas.​

* El bot solo responde a las preguntas obtenidas en el dataset, no puede responder a ninguna pregunta fuera del dataset​
* El bot en la mayoria de las preguntas preguntas puede entender el contexto de la pregunta sin que la pregunta este completa ​

Inicialmente el modelo responde adecuadamente cuando se le realizan pruebas con el dataset de entrenamiento como con el dataset de validación. También tiene una buena respuesta al utilizar textos fuera de los datasets ya mencionados, por lo tanto, se logró utilizar el modelo BERT pre-entrenado y el dataset SQuAD v1.1 (Stanford Question Answering Dataset) para construir un modelo que puede responder de manera genérica respondiendo preguntas en base a otros contextos. Con este modelo se puede construir muy fácilmente un chatbot que llame al modelo y enviándole el contexto para que luego pueda devolver la respuesta a las preguntas de los usuarios.

# Recomendaciones

​

* Se recomienda utilizar un dataset mas grande ​
* Se recomienda realizar variaciones al modelo para un mejor aprendisaje ​
* Se recomienda utilizar la version paga del colab

Es importante mencionar que el modelo puede ser mejorado utilizando un dataset de entrenamiento más grande como lo es SQuAD v2.0 y se pueden utilizar versiones de modelo BERT pre-entrenado que soporte otros idiomas o incluso varios idiomas al mismo tiempo.

# References

Google. (2023). *Comienza a usar el algoritmo integrado BERT | AI Platform Training*. Google Cloud. Retrieved February 17, 2023, from <https://cloud.google.com/ai-platform/training/docs/algorithms/bert-start?hl=es-419>

OpenAI. (2022). *Documentacio*. OpenAI. Retrieved February 17, 2023, from https://openai.com