RETRIEVAL AUGMENTED GENERATION(RAG)

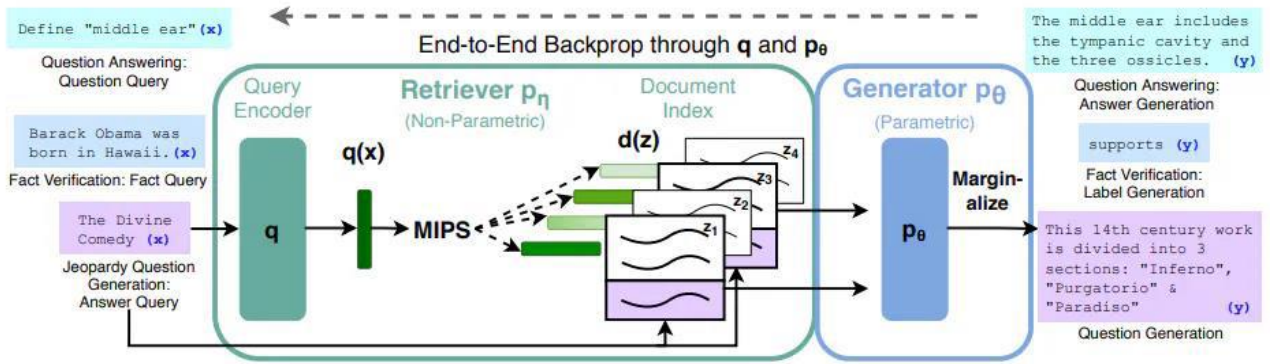


Figure 1: Overview of our approach. We combine a pre-trained retriever (*Query Encoder + Document Index*) with a pre-trained seq2seq model (*Generator*) and fine-tune end-to-end. For query x, we use Maximum Inner Product Search (MIPS) to find the top-K documents z_i. For final prediction y, we treat z as a latent variable and marginalize over seq2seq predictions given different documents.

Índice

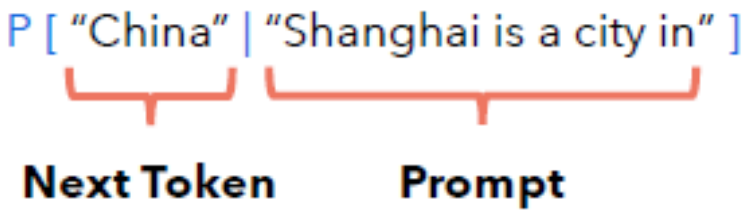
Prerrequisitos:

- Estructura del modelo Transformer y cómo funciona el mecanismo de atención.
- BERT (tarea MLM, token [cls])

Índice

¿Qué es un modelo de lenguaje?

Un modelo de lenguaje es un modelo probabilístico que asigna probabilidades a secuencias de palabras. En la práctica, un modelo de lenguaje nos permite calcular lo siguiente:

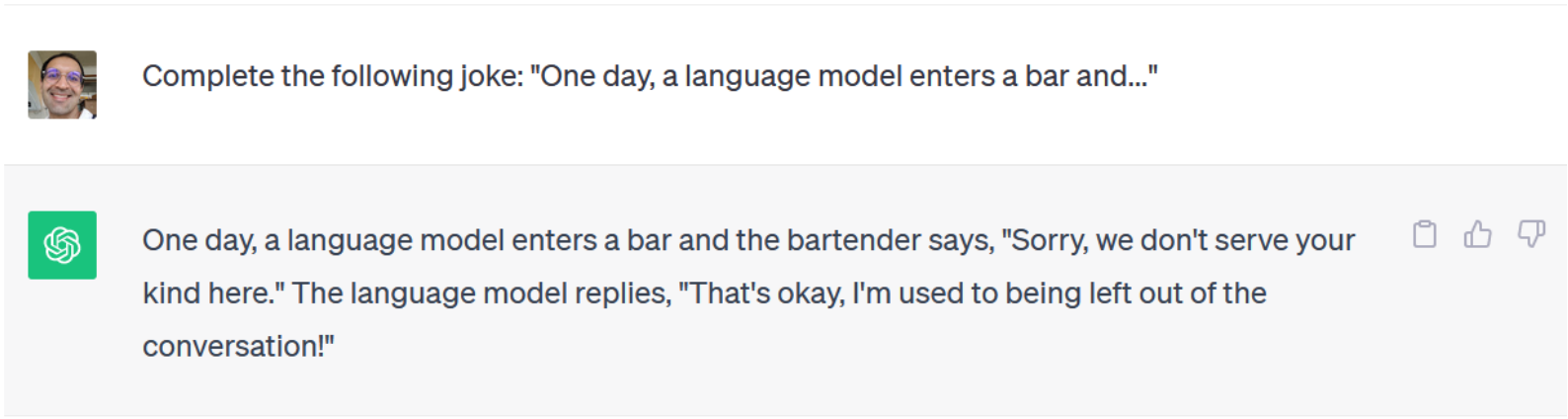


Usualmente entrenamos una red neuronal para predecir estas probabilidades. Una red neuronal entrenada en grandes corpus de texto se conoce como un Modelo de Lenguaje Extenso (LLM, por sus siglas en inglés).

¿Cómo entrenamos e inferimos un modelo de lenguaje?

Entrenamiento
Un modelo de lenguaje se entrena en un corpus de texto, es decir, una gran colección de documentos. A menudo, los modelos de lenguaje se entrenan en toda la Wikipedia y en millones de páginas web. Esto permite que el modelo de lenguaje adquiera la mayor cantidad de conocimiento posible.Usualmente entrenamos una red neuronal basada en transformadores como modelo de lenguaje.

Inferencia
Para inferir un modelo de lenguaje, construimos un contexto (prompt) y dejamos que el modelo de lenguaje genere el resto añadiendo tokens iterativamente.



Eres lo que comes

Un modelo de lenguaje solo puede generar texto e información sobre lo que fue entrenado. Esto significa que, si entrenamos un modelo de lenguaje solo con contenido en inglés, es muy probable que no sea capaz de generar texto en japonés o francés. Para enseñarle nuevos conceptos, necesitamos ajustar finamente (fine-tune) el modelo.

Los contras del ajuste fino (fine-tuning)

- Puede ser costoso.
- El número de parámetros del modelo puede no ser suficiente para capturar todo el conocimiento que queremos enseñarle. Es por eso que LLaMA fue introducido con 7B, 13B y 70B parámetros.
- El ajuste fino no es aditivo. Puede reemplazar el conocimiento existente del modelo con nuevo conocimiento. Por ejemplo, un modelo de lenguaje entrenado en inglés que es (fuertemente) ajustado en japonés puede "olvidar" el inglés.

¡Ingeniería de Prompts al rescate!

Es posible "enseñar" a un modelo de lenguaje cómo realizar una nueva tarea jugando con el prompt. Por ejemplo, utilizando el "prompting de pocos ejemplos" (few-shot prompting). A continuación se muestra un ejemplo:

奥利奥 is a cat that likes to play tricks on his friend Umar by replacing all the names in everything he writes with "meow".

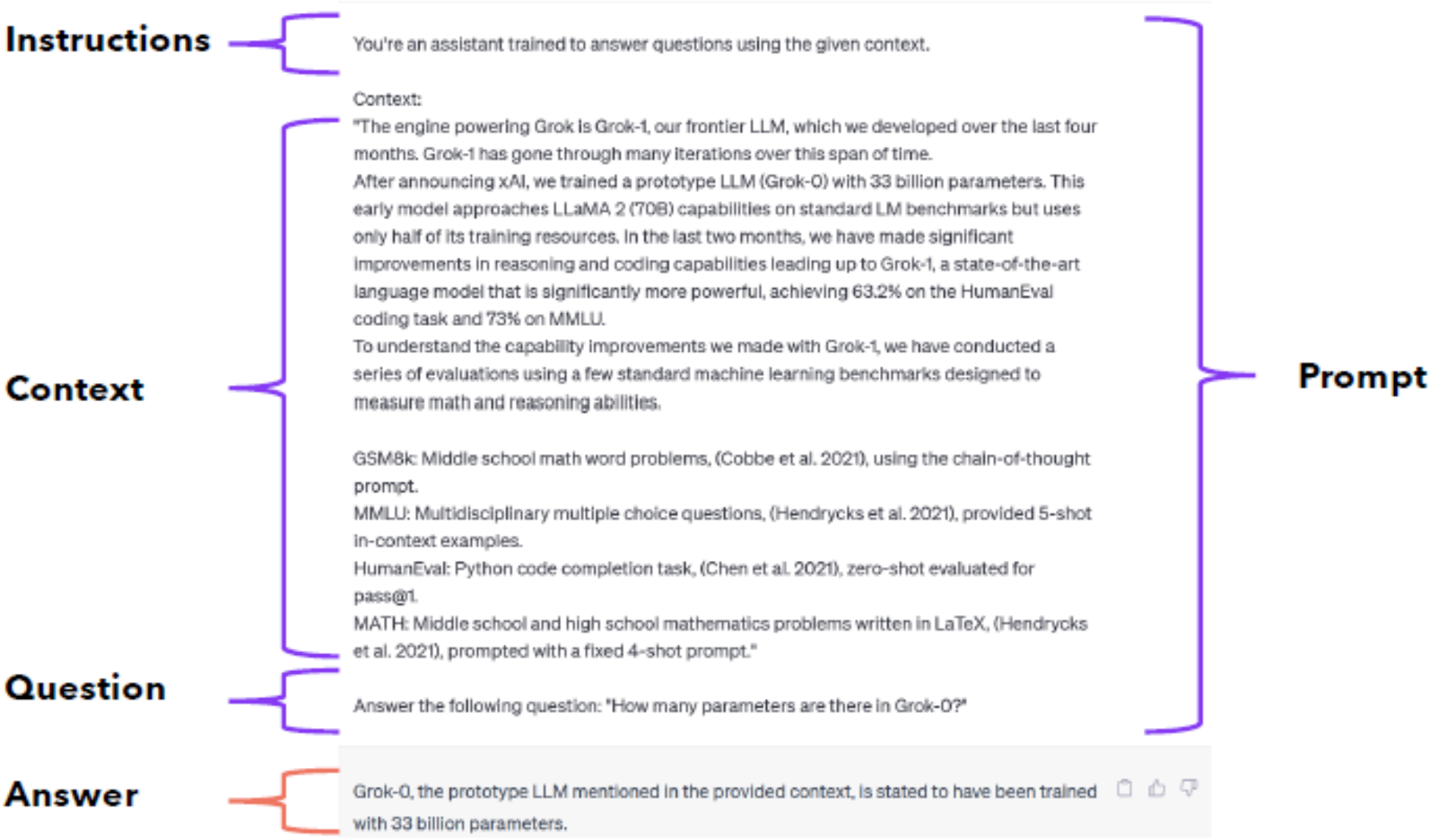
For example:

Umar writes: "Bob runs a YouTube channel."
奥利奥 modifies it to: "Meow runs a YouTube channel."

Umar writes: "Alice likes to play with his friend Bob"
奥利奥 modifies it to:

"Meow likes to play with his friend Meow."

QA con Ingeniería de Prompts



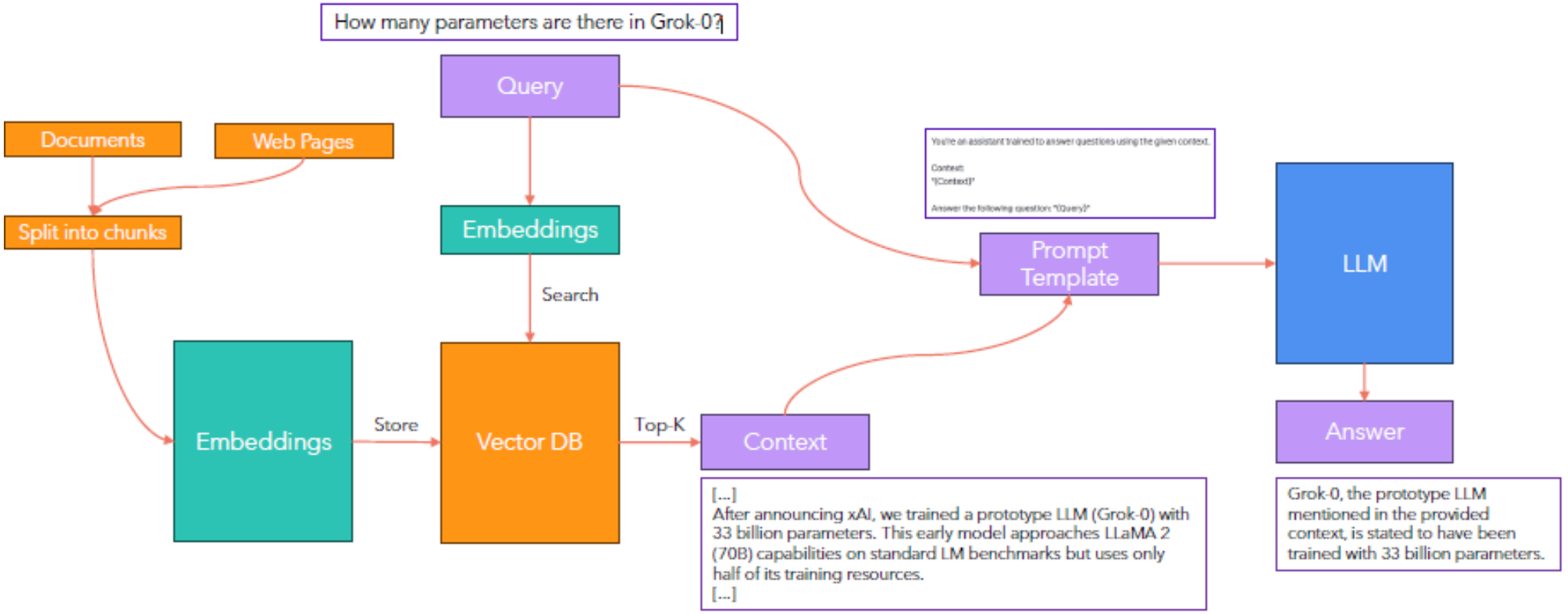
Los pros de ajustar el modelo (fine-tuning)

- Resultados de mayor calidad en comparación con la ingeniería de prompts.
- Tamaño de contexto (tamaño de entrada) más pequeño durante la inferencia, ya que no necesitamos incluir el contexto y las instrucciones.

Índice

- Introducción a los Modelos de Lenguaje a Gran Escala
- Proceso de RAG
- Vectores de Embeddings
    - Sentence BERT
- Base de Datos de Vectores
    - Algoritmos (HNSW)

QA con Generación Aumentada por Recuperación

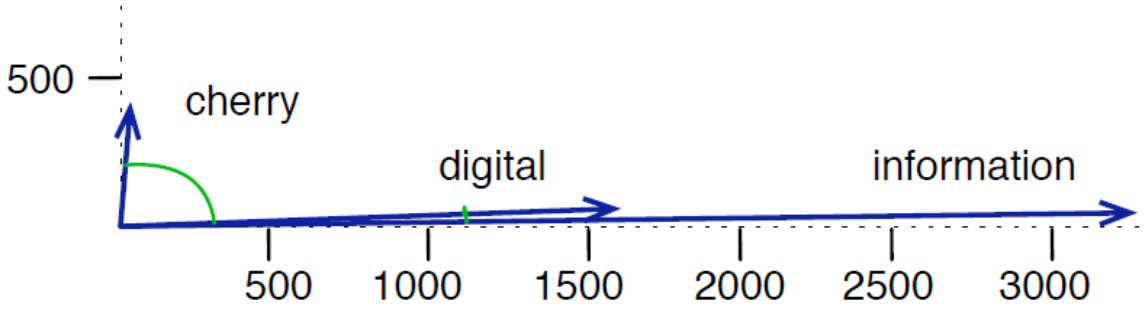


Índice

- Introducción a los Modelos de Lenguaje a Gran Escala
- Proceso de RAG
- Vectores de Embeddings
  - Sentence BERT
- Base de Datos de Vectores
  - Algoritmos (HNSW)

¿Por qué usamos vectores para representar palabras?

- Dado que las palabras **"cherry“, "digital"** e **"information"** (información), si representamos los vectores de embeddings utilizando solo 2 dimensiones (X, Y) y los graficamos, esperamos ver algo como esto: el ángulo entre palabras con significado similar es pequeño, mientras que el ángulo entre palabras con diferente significado es grande. Entonces, los embeddings "capturan" el significado de las palabras que representan al proyectarlas en un espacio de alta dimensionalidad.



Source: Speech and Language Processing 3rd Edition Draft, Dan Jurafsky and James H. Martin

- Comúnmente utilizamos la **similitud coseno**, que se basa en el **producto punto** entre los dos vectores.

Embeddings de palabras: las ideas

- Las palabras que son sinónimos tienden a aparecer en el mismo contexto (rodeadas por las mismas palabras).
- Por ejemplo, la palabra teacher (profesor/a) y professor (catedrático/a) usualmente aparecen rodeadas por palabras como school (escuela), university (universidad), exam (examen), lecture (conferencia), course (curso), etc.
- Lo inverso también puede ser cierto: las palabras que aparecen en el mismo contexto tienden a tener significados similares. Esto se conoce como la **hipótesis distribucional.**
- Esto significa que para capturar el significado de una palabra, también necesitamos tener acceso a su contexto (las palabras que la rodean).
- Es por eso que utilizamos el mecanismo de Self-Attention en el modelo Transformer para capturar información contextual para cada token. El mecanismo de Self-Attention relaciona cada token con todos los demás tokens en la oración.

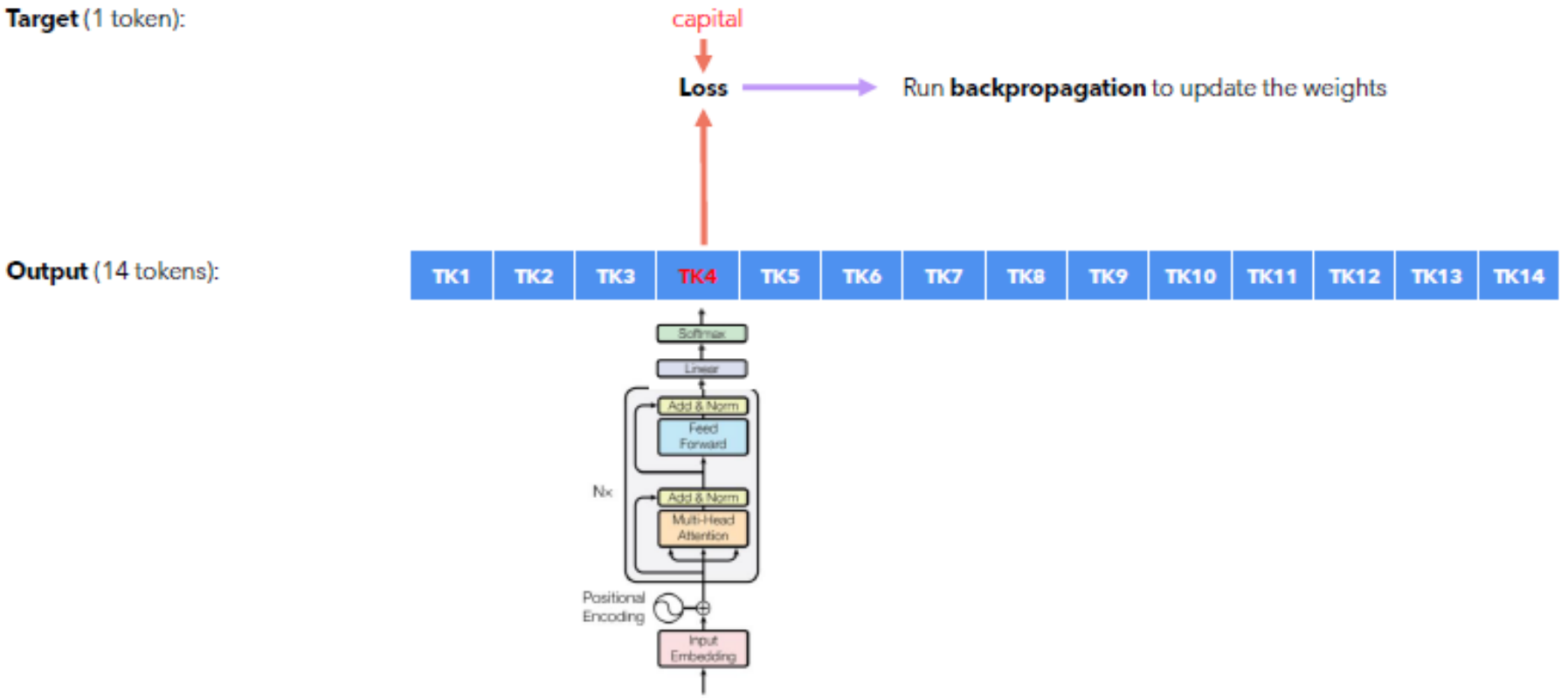
Embeddings de palabras: la tarea Cloze

- Imagina que te doy la siguiente oración:
  Roma es la _____ de Italia, por lo cual alberga muchos edificios gubernamentales.
  ¿Puedes decirme cuál es la palabra que falta?

- ¡Por supuesto! La palabra que falta es "capital", porque al observar el resto de la oración, es la que tiene más sentido.

- Así es como entrenamos a BERT: queremos que el mecanismo de Self-Attention relacione todos los tokens de entrada entre sí, de modo que BERT tenga suficiente información sobre el "contexto" de la palabra que falta para predecirla.

¿Cómo entrenamos los vectores de embeddings en BERT?

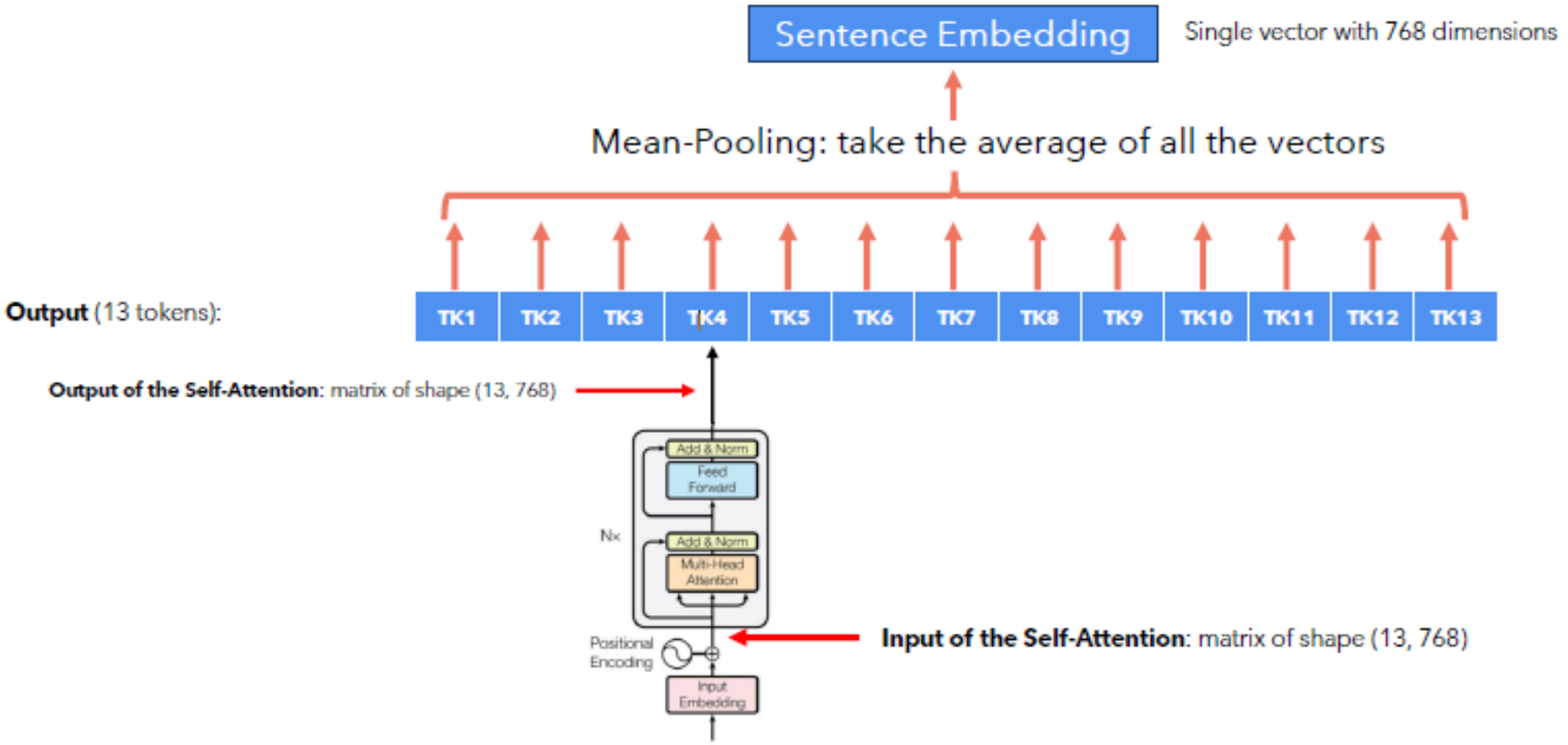


Embeddings de Oraciones

- Podemos utilizar el mecanismo de Self-Attention para capturar el "significado" de una oración completa.

- Podemos usar un modelo BERT preentrenado para producir embeddings de oraciones completas. Veamos cómo.

Embeddings de Oraciones



Embeddings de Oraciones: comparación

- ¿Cómo podemos comparar los embeddings de oraciones para ver si dos oraciones tienen un "significado" similar? Podríamos usar la similitud del coseno, que mide el coseno del ángulo entre los dos vectores. Un ángulo pequeño resulta en una alta puntuación de similitud del coseno.

$$\text{cosine similarity} = S_C(A, B) := \cos(\theta) = \frac{\mathbf{A} \cdot \mathbf{B}}{\|\mathbf{A}\|\|\mathbf{B}\|} = \frac{\sum_{i=1}^{n} A_i B_i}{\sqrt{\sum_{i=1}^{n} A_i^2} \cdot \sqrt{\sum_{i=1}^{n} B_i^2}}$$

- **Pero hay un problema:** nadie le dijo a BERT que los embeddings que produce deberían ser comparables con la similitud del coseno, es decir, que dos oraciones similares deberían estar representadas por vectores que apunten en la misma dirección en el espacio. ¿Cómo podemos enseñar a BERT a producir embeddings que puedan compararse con una función de similitud de nuestra elección?

Índice

- Introducción a los Modelos de Lenguaje a Gran Escala
- Proceso de RAG
- Vectores de Embeddings
    - Sentence BERT
- Base de Datos de Vectores
    - Algoritmos (HNSW)

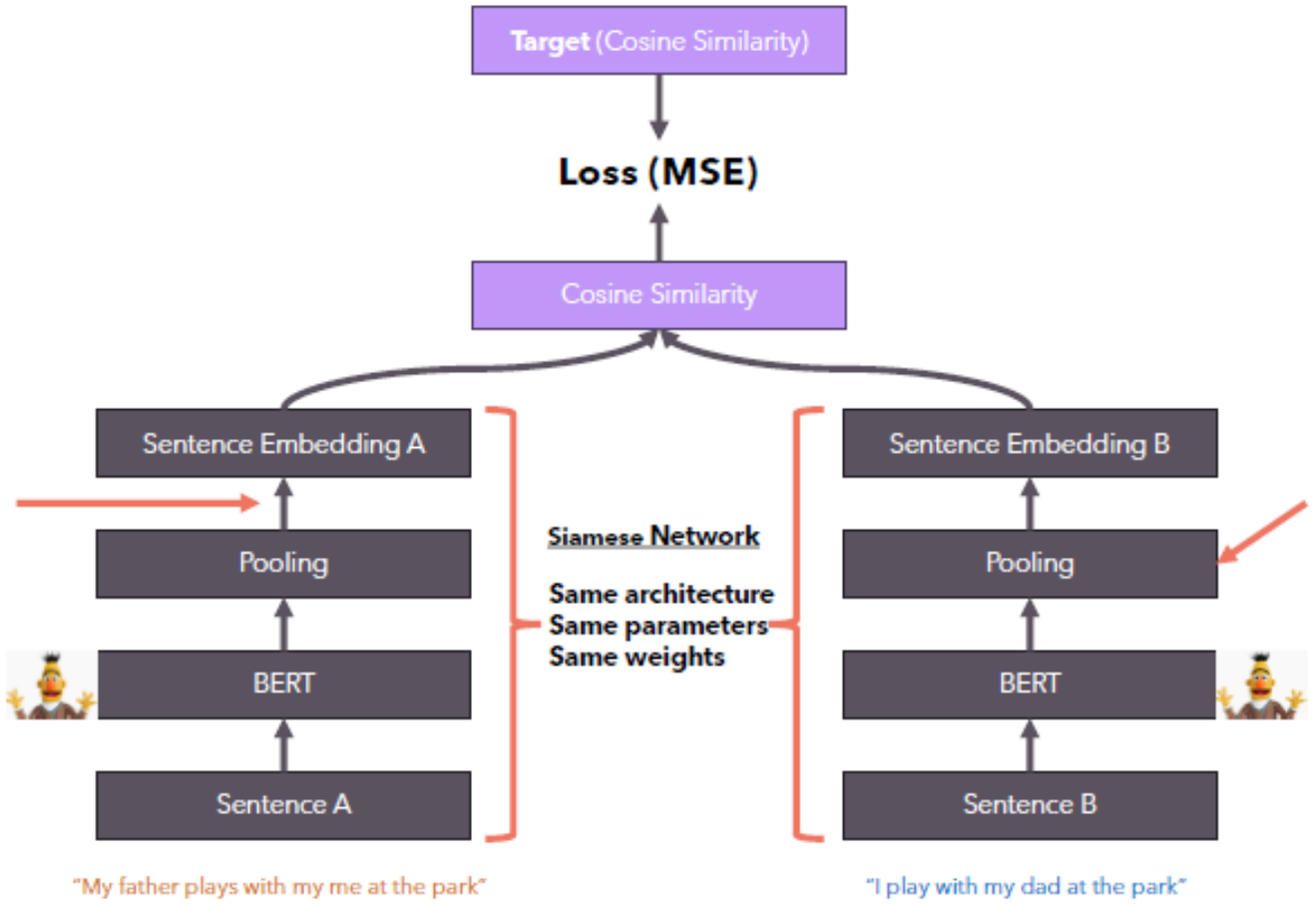
Sentence-BERT: Embeddings de Oraciones utilizando Redes BERT-Siamés

Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks

Nils Reimers and Iryna Gurevych
Ubiquitous Knowledge Processing Lab (UKP-TUDA)
Department of Computer Science, Technische Universität Darmstadt
www.ukp.tu-darmstadt.de

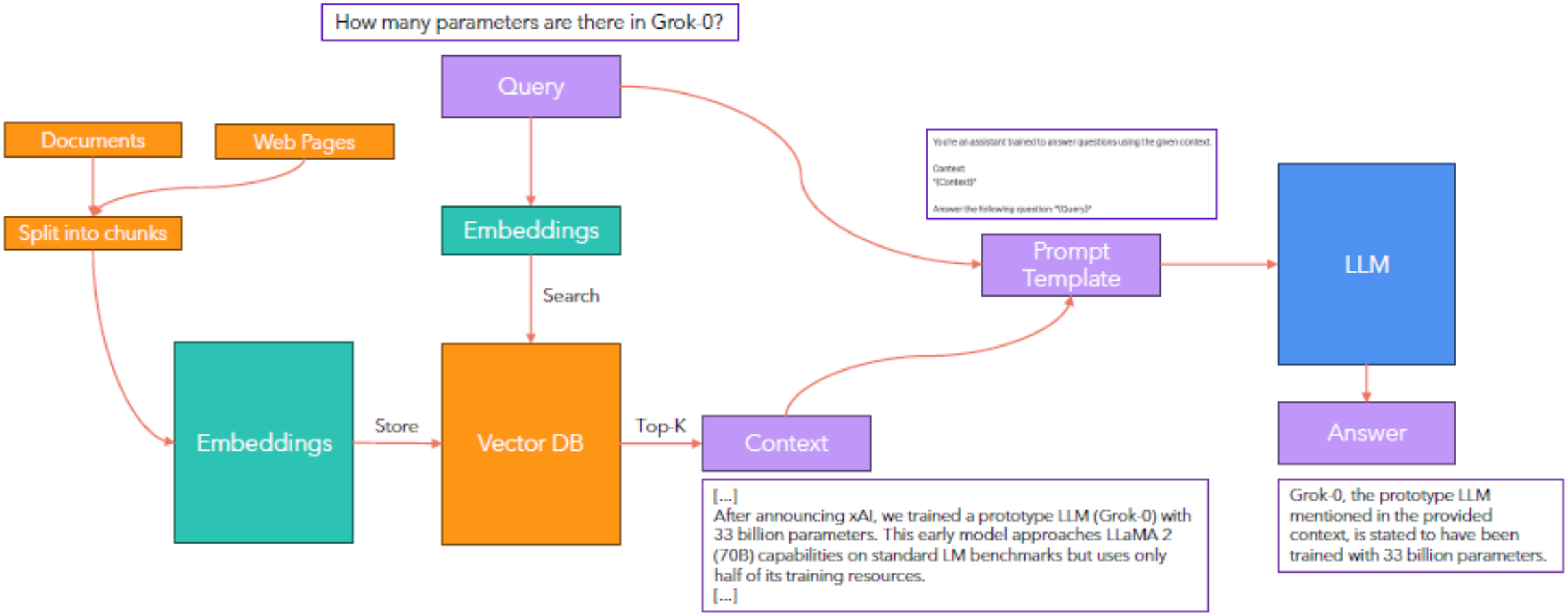
Sentence BERT: arquitectura

Podemos aplicar una capa lineal para reducir el tamaño del vector de embeddings, por ejemplo, de 768 a 512.



Podemos usar mean-pooling, max-pooling o simplemente usar el token [cls] como embedding de oración..

QA con Generación Aumentada por Recuperación



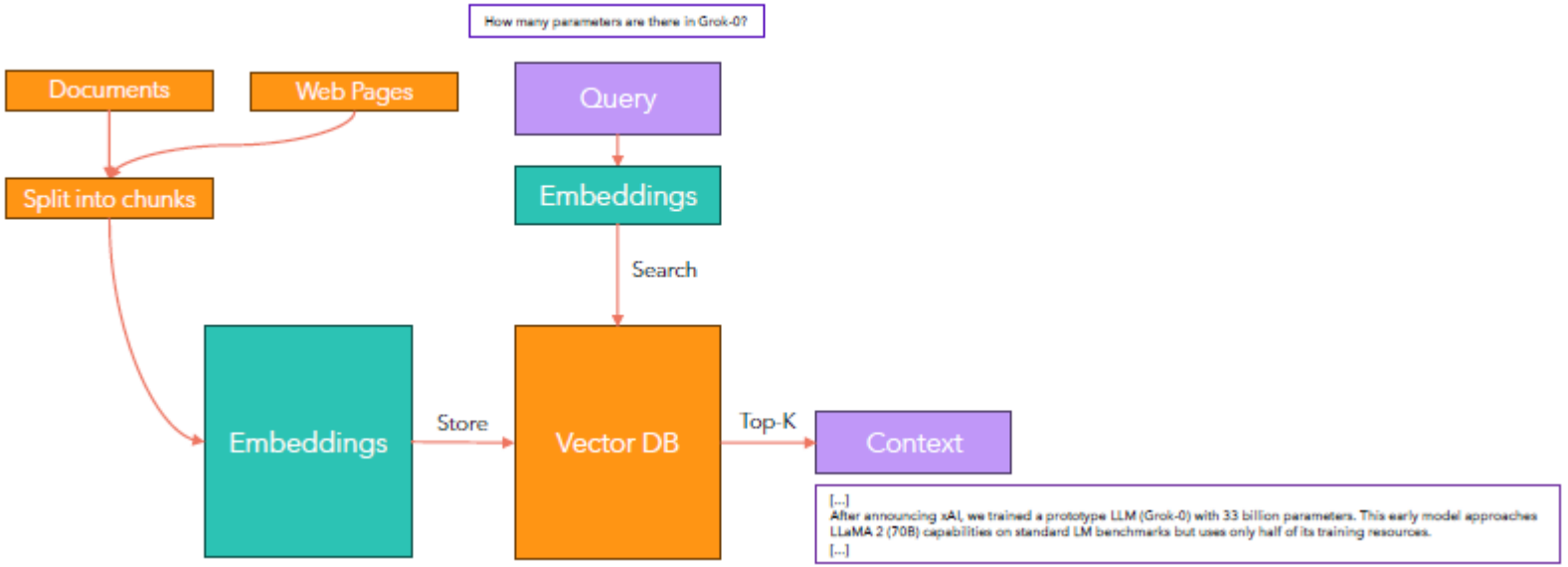
Estrategias para enseñar nuevos conceptos a un LLM

Índice

- Introducción a los Modelos de Lenguaje a Gran Escala
- Proceso de RAG
- Vectores de Embeddings
    - Sentence BERT
- Base de Datos de Vectores
    - Algoritmos (HNSW)

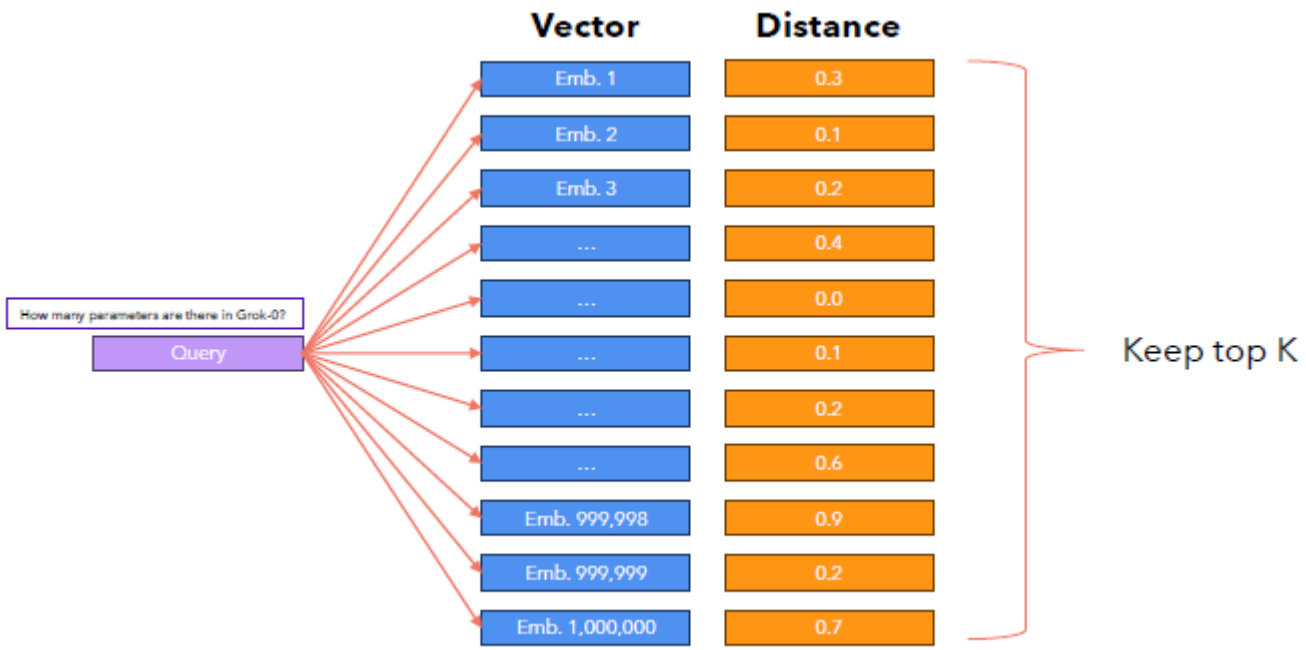
Base de Datos Vectorial: introducción

Una base de datos vectorial almacena vectores de dimensiones fijas (llamados embeddings), lo que nos permite hacer consultas a la base de datos para encontrar todos los embeddings que sean los más cercanos (o más similares) a un vector de consulta utilizando una métrica de distancia, que generalmente es la similitud del coseno, aunque también se puede usar la distancia euclidiana. La base de datos utiliza una variante del algoritmo KNN (K Nearest Neighbor) u otro algoritmo de búsqueda de similitudes. Las bases de datos vectoriales también se utilizan para encontrar canciones similares (por ejemplo, Spotify), imágenes (por ejemplo, Google Imágenes) o productos (por ejemplo, Amazon).



K-NN: un enfoque ingenuo

Imagina que queremos buscar la consulta en nuestra base de datos: una forma sencilla sería comparar la consulta con todos los vectores, ordenarlos por distancia y quedarnos con los top K.



Si hay N vectores de embedding y cada uno tiene D dimensiones, la complejidad computacional está en el orden de O(N*D), ¡demasiado lento!

Índice

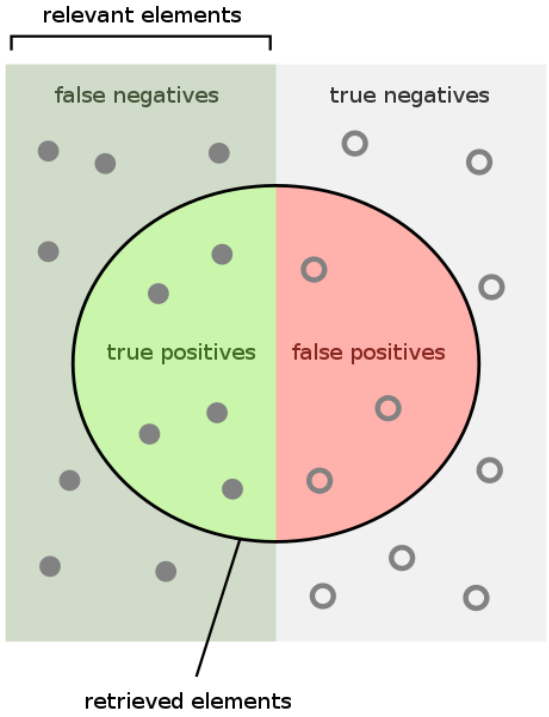
- Introducción a los Modelos de Lenguaje a Gran Escala
- Proceso de RAG
- Vectores de Embeddings
  - Sentence BERT
- Base de Datos de Vectores
  - Algoritmos (HNSW)

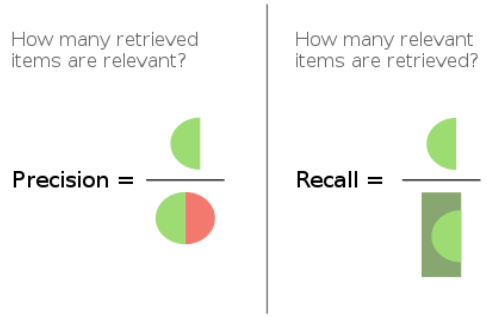
Búsqueda de Similitud: cambiemos precisión por velocidad

El enfoque ingenuo que usamos antes siempre produce resultados precisos, ya que compara la consulta con todos los vectores almacenados. Pero, ¿qué pasaría si redujéramos el número de comparaciones, pero aún así obtuviéramos resultados precisos con alta probabilidad?

La métrica que generalmente nos importa en la Búsqueda de Similitud es el recall (tasa de recuperación).

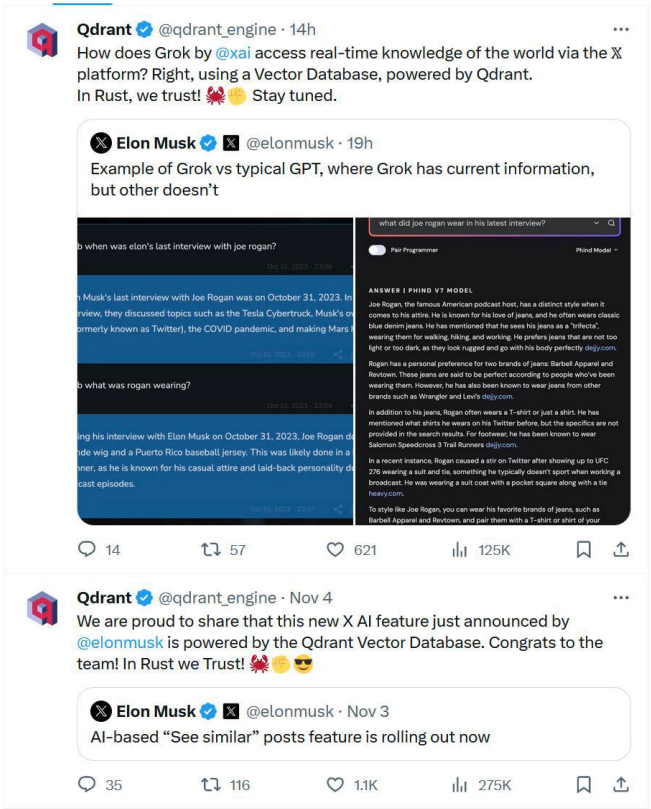
En este video exploraremos un algoritmo para Vecinos Aproximados Más Cercanos, llamado **Mundos Pequeños Navegables Jerárquicos (HNSW).**

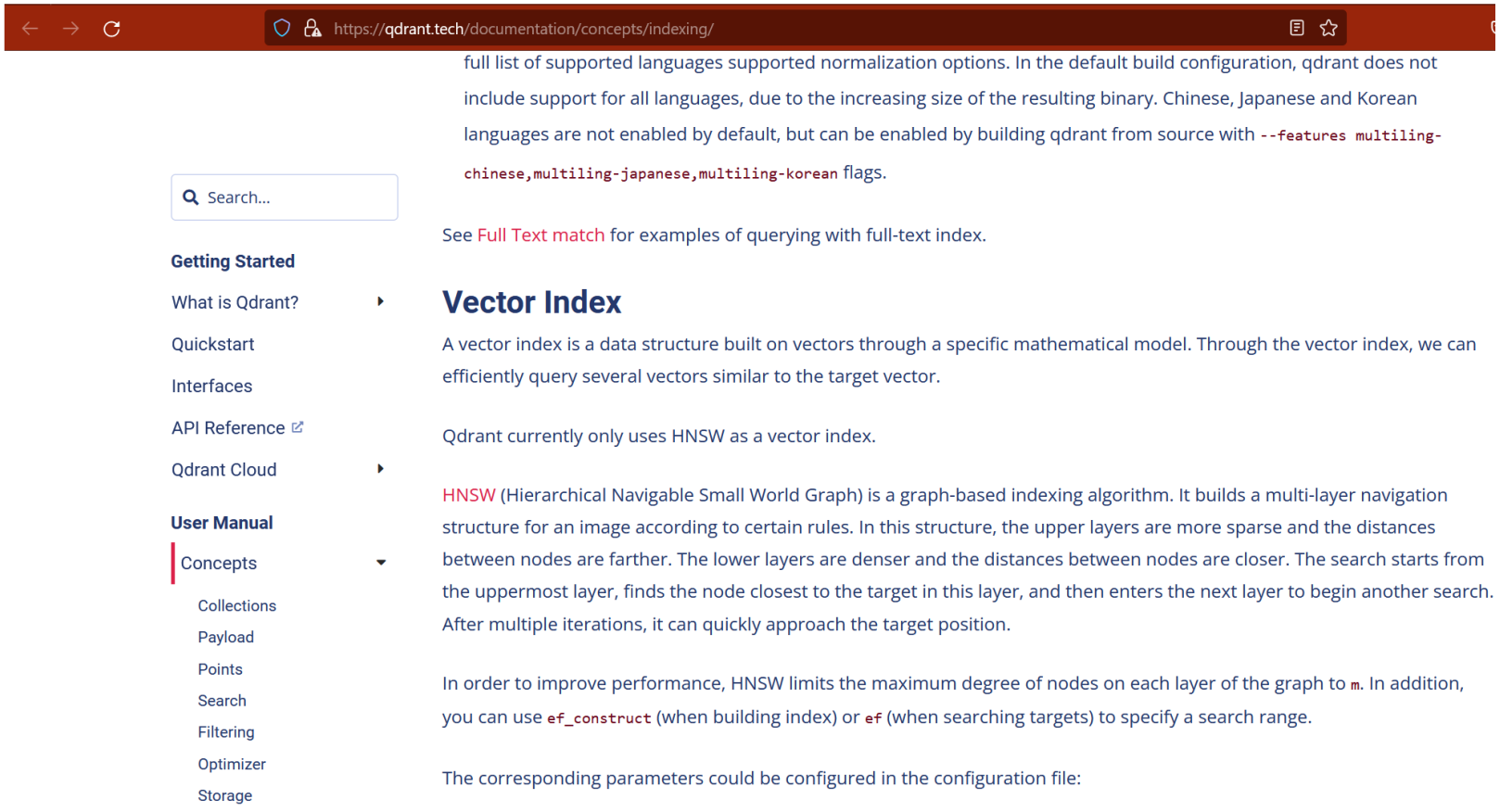




HNSW en el mundo real

Es el mismo algoritmo que potencia Qdrant, la base de datos de vectores de código abierto utilizada por el LLM Grok de Twitter (X), que puede acceder a tweets en tiempo real.





HNSW: idea #1

HNSW es una evolución del algoritmo de **Navigable Small Worlds** para Approximate Nearest Neighbors, que se basa en el concepto de **Six Degrees of Separation** (Seis grados de separación).

El experimento de Milgram tenía como objetivo probar las conexiones sociales entre las personas en los Estados Unidos. Los participantes, que inicialmente estaban ubicados en Nebraska y Kansas, recibieron una carta para entregarla a una persona específica en Boston. Sin embargo, no se les permitió enviar la carta directamente al destinatario. En su lugar, se les indicó que la enviaran a alguien que conocieran por su nombre, a quien creyeran que tendría más posibilidades de conocer a la persona objetivo.

Al final del experimento de Milgram sobre el mundo pequeño, Milgram descubrió que la mayoría de las cartas llegaron al destinatario final en cinco o seis pasos, creando el concepto de que todas las personas en el mundo están conectadas por seis grados de separación.

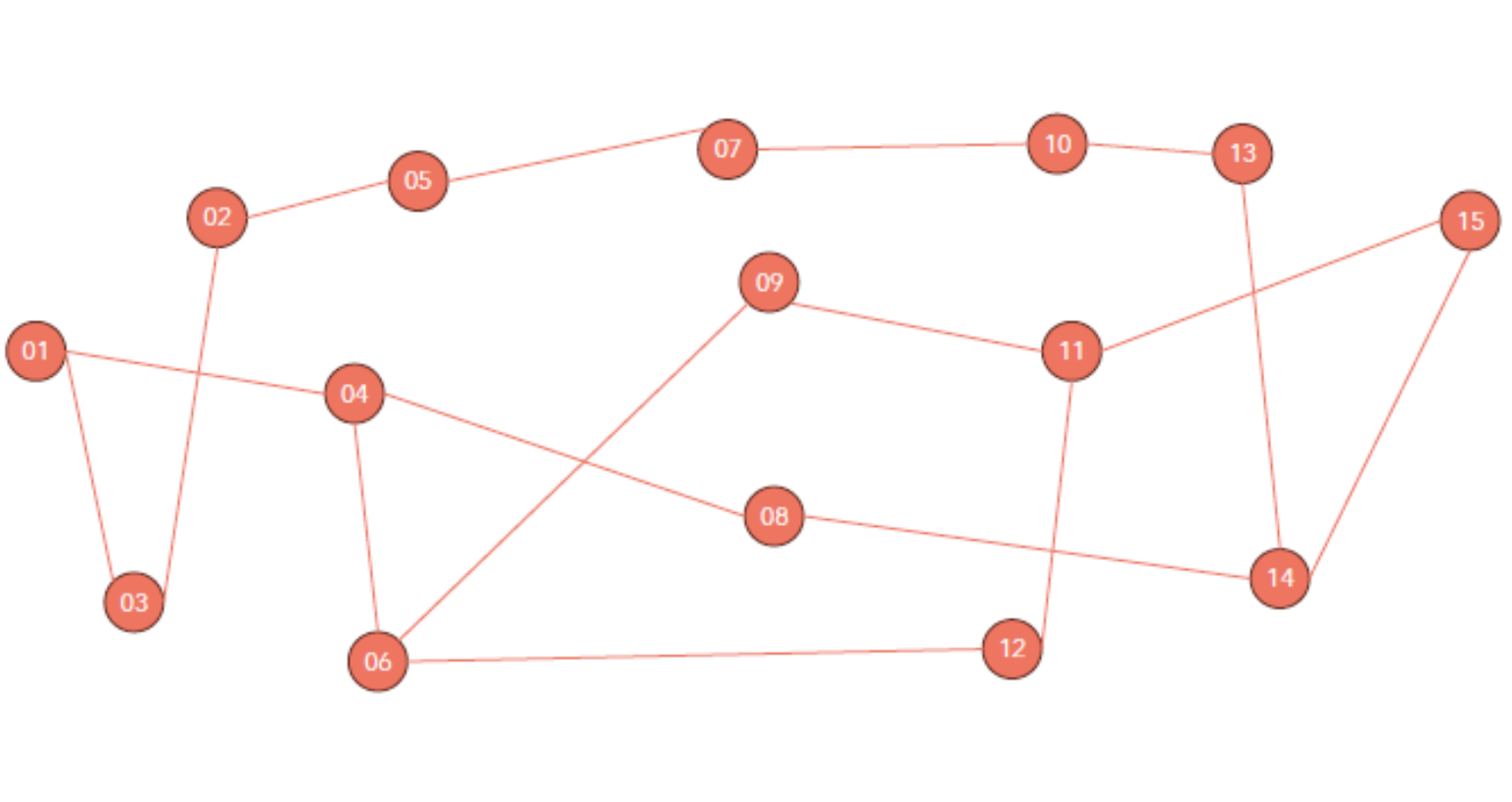
Facebook descubrió en 2016 que sus 1.59 mil millones de usuarios activos estaban conectados en promedio por 3.5 grados de separación. https://research.facebook.com/blog/2016/02/three-and-a-half-degrees-of-separation/

Esto significa que tú y Mark Zuckerberg están separados por solo 3.5 conexiones.

Navigable Small Worlds

El algoritmo NSW construye un gráfico que, al igual que los amigos en Facebook, conecta vectores cercanos entre sí, pero manteniendo pequeño el número total de conexiones. Por ejemplo, cada vector puede estar conectado hasta con 6 otros vectores (para imitar los Seis Grados de Separación).

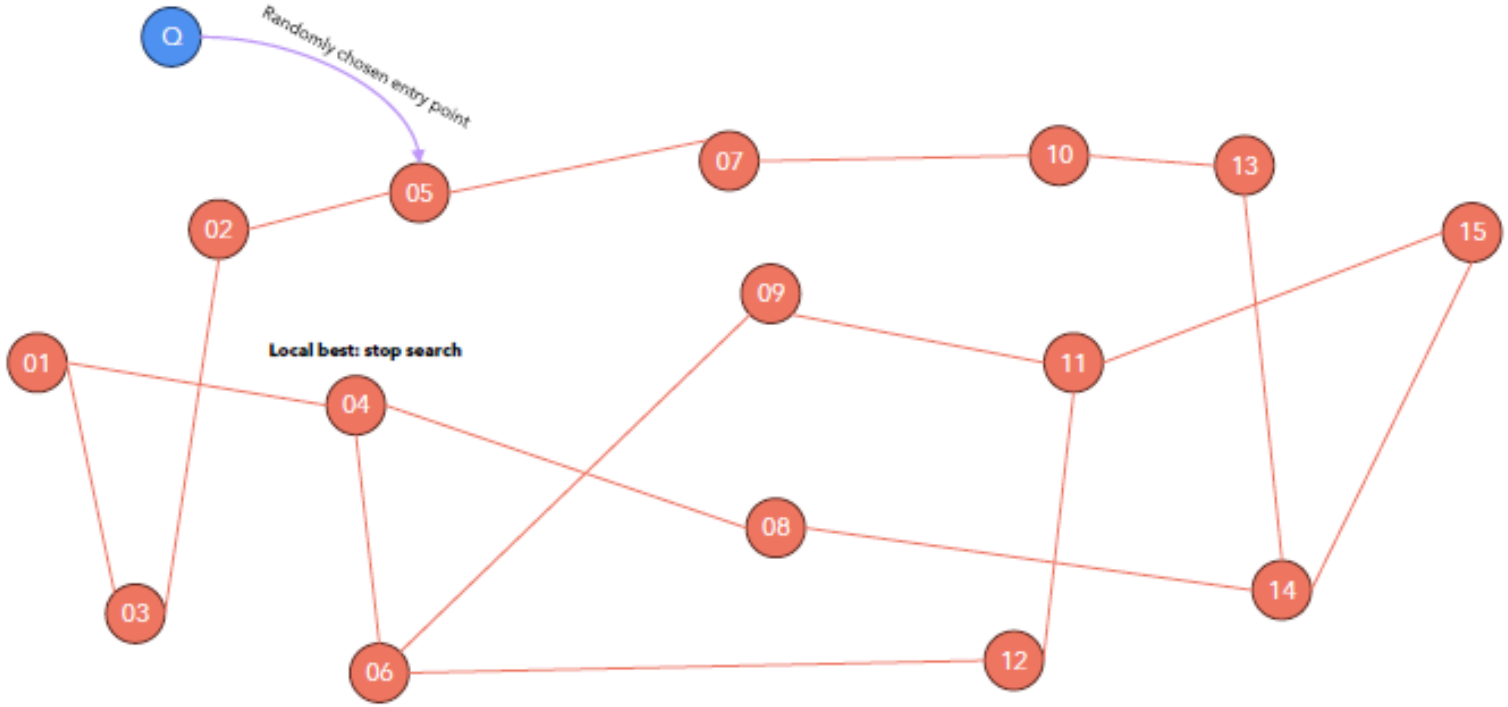
Node	Text
01	[...] The Transformer is a model [...]
02	[...] Diagnose cancer with AI [...]
03	[...] A transformer-based model [...]
04	[...] The Transformer has 6 layers [...]
05	[...] An MRI machine that costs 1$ [...]
06	[...] The dot-product is a [...]
07	[...] Big-Pharma is not so big [...]
08	[...] Cross-Attention is a great [...]
09	[...] To solve an ODE [...]
10	[...] We are aging too fast [...]
11	[...] Open-source models like [...]
12	[...] MathBERT: a new model [...]
13	[...] AI to control aging [...]
14	[...] Attention is all you need [...]
15	[...] LLaMA 2 has 7B params [...]



Navigable Small Worlds: buscando K-NN

Dada la siguiente consulta: "¿Cuántas capas de codificador hay en el modelo Transformer?", ¿Cómo encuentra el algoritmo los K Vecinos Más Cercanos?

Node	Text
01	[...] The Transformer is a model [...]
02	[...] Diagnose cancer with AI [...]
03	[...] A transformer-based model [...]
04	[...] The Transformer has 6 layers [...]
05	[...] An MRI machine that costs 1$ [...]
06	[...] The dot-product is a [...]
07	[...] Big-Pharma is not so big [...]
08	[...] Cross-Attention is a great [...]
09	[...] To solve an ODE [...]
10	[...] We are aging too fast [...]
11	[...] Open-source models like [...]
12	[...] MathBERT: a new model [...]
13	[...] AI to control aging [...]
14	[...] Attention is all you need [...]
15	[...] LLaMA 2 has 7B params [...]



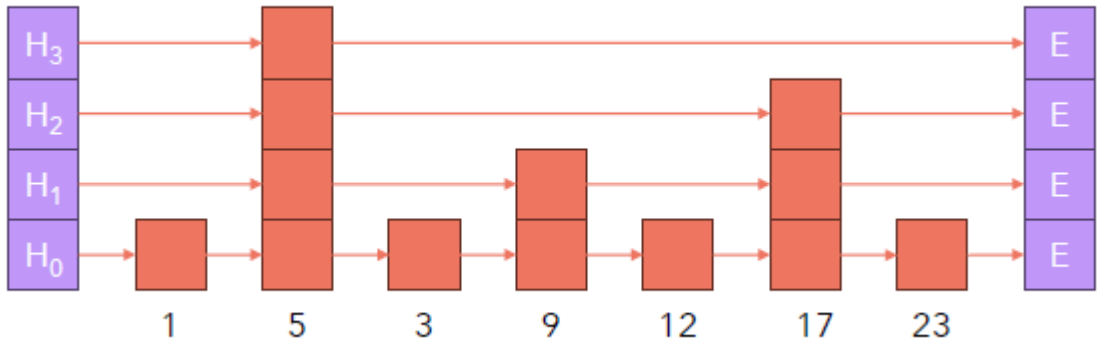
Repetimos la búsqueda con puntos de partida elegidos aleatoriamente y luego mantenemos los top K entre todos los nodos visitados.

Navigable Small Worlds: insertando un nuevo vector

- Podemos insertar un nuevo vector buscando los top K vecinos más cercanos (KNN) con el algoritmo de búsqueda descrito anteriormente y creando un enlace entre el nuevo vector y los resultados top K.

HNSW: idea #2

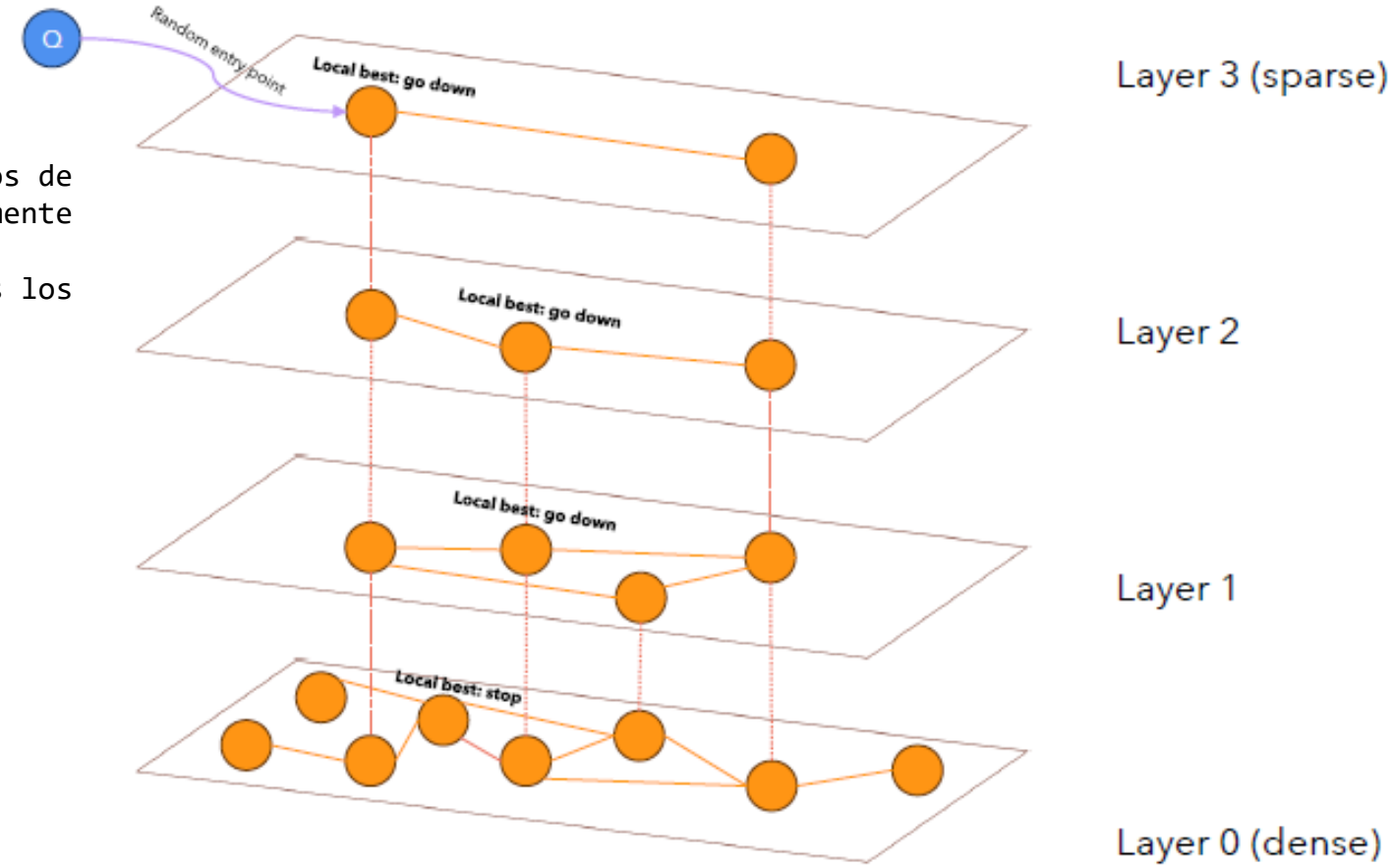
Para pasar de NSW (Mundos Pequeños Navegables) a HNSW (Mundos Pequeños Navegables Jerárquicos), necesitamos introducir el algoritmo detrás de la estructura de datos conocida como Skip-List (lista con saltos).La Skip-List es una estructura de datos que mantiene una lista ordenada y permite la búsqueda e inserción con una complejidad de tiempo promedio de O(log N).Busquemos el número 9.



HNSW: Mundos Pequeños Navegables Jerárquicos

¡Vamos a buscar!

Repetimos la búsqueda con puntos de inicio seleccionados aleatoriamente (en la capa superior) y luego mantenemos el top K entre todos los nodos visitados.



QA con Generación Aumentada por Recuperación