Apuntes Semana 12, Martes 21 de Octubre

Carranza Jiménez Kevin
Instituto Tecnológico de Costa Rica
Correo electrónico: kcarranza@estudiantec.cr

***Resumen*—El siguiente documento presenta el resumen de la clase del día martes 21 de octubre, impartida por el profesor Steven Pacheco Portugüez en el Instituto Tecnológico de Costa Rica. La clase presenta el cronograma restante del curso, un resumen de la clase anterior en el que se repasan los temas de modelos de lenguaje a gran escala (LLM), tokenización, embeddings y la introdución del paradigma de Retrieval-Augmented Generation (RAG) y agentes inteligentes. En la presente clase se profundiza en el tema de LLM, RAGs y agentes introducidos en la clase anterior.**

***Index Terms*—LLM, RAG, Embedding, Agente**

I. Introducción

La sesión inició con una revisión del cronograma restante del curso. Posteriormente, se realizó un repaso de la clase anterior, en la cual se introdujeron conceptos fundamentales sobre los modelos de lenguaje de gran escala (Large Language Models, LLM), el proceso de tokenización y la representación semántica mediante *embeddings*.

A partir de este punto, la clase se centró en dos temas principales: la integración de modelos mediante esquemas de Recuperación Aumentada de Generación (Retrieval-Augmented Generation, RAG) y la noción de agentes inteligentes.

II. Aspectos administrativos de la clase

Se presentó el calendario en el cual se muestrán las próximas actividades y evaluaciones que restan del curso. En la tabla I.

III. RAGs y Agentes Utilizando LLMs

Los esquemas de Recuperación Aumentada de Generación (Retrieval-Augmented Generation, RAG) y los agentes inteligentes representan una evolución significativa en el uso de los modelos de lenguaje de gran escala (LLM). Los RAG combinan la capacidad generativa de los LLM con mecanismos de recuperación de información externa, permitiendo respuestas más precisas y actualizadas basadas en conocimiento relevante [1]. Por su parte, los agentes inteligentes extienden este enfoque al incorporar razonamiento, planificación y toma de decisiones, posibilitando sistemas que no solo generan texto, sino que también actúan de manera autónoma en función de objetivos específicos [2].

III-A. ¿Por qué los LLM son tan utilizados?

Los modelos de lenguaje de gran escala (LLM) se han convertido en la base de numerosos sistemas modernos de inteligencia artificial, impulsando avances significativos en tareas de generación, comprensión y razonamiento sobre texto, código e incluso modalidades más complejas como imágenes y audio. Estos modelos son capaces de representar conocimiento a gran escala mediante el aprendizaje de patrones lingüísticos y semánticos a partir de enormes volúmenes de datos, lo que explica la sorprendente coherencia y versatilidad de sus resultados [3]. Comprender los mecanismos internos que permiten estas representaciones, así como sus limitaciones y potencial de generalización, resulta esencial para el desarrollo de aplicaciones más seguras y efectivas basadas en inteligencia artificial generativa.

La Figura 1 ilustra la arquitectura de un modelo de red neuronal preentrenado diseñado para la clasificación de eventos de colisión. Este modelo recibe como entrada un conjunto de características o *features* que incluyen la velocidad del vehículo, la calidad del terreno, el grado de visión disponible y la experiencia total del conductor. A partir de estos parámetros, la red aprende a identificar patrones que permiten estimar la probabilidad de que ocurra una colisión bajo determinadas

Cuadro I
Cronograma de clases y actividades

Semana	Martes	Jueves
12	Clase Agentes - LLM y Asignación de Tarea 04 Agentes	Clase Quantization
13	Aplicar Quiz 6 y Clase Quantization - Unsupervised	Clase Unsupervised - PCA y Entrega de Proyecto I
14	Evaluación presencial Proyecto I.	Entrega Tarea 04 Agentes y Evaluación presencial PRoyecto I
15	Clase virtual Unsupervised - PCA, Asignación de Proyecto II y Asignación de Tarea 05 Autoencoder - Quantization	Revisión virtual de Tarea 04 Agentes
16	Clase Sesgos de AI	
17	Semana Colchón	Semana Colchón
18	Examen I	Entrega Proyecto II

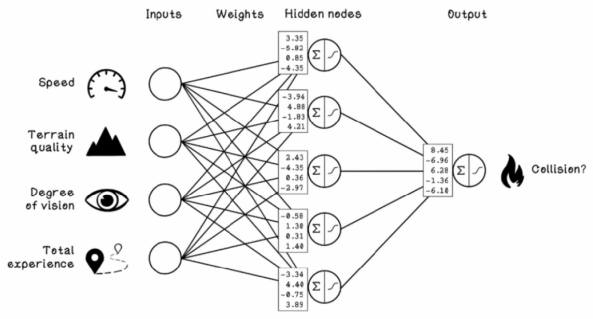

Figura 1. Modelo de red neuronal preentrenado para la clasificación de eventos de colisión.

Cuadro II
EJEMPLO SIMPLIFICADO DE TOKENIZACIÓN

Palabra	Token	ID Numérico
Los	los	105
LLM	llm	2124
aprenden	aprenden	893
patrones	patrones	5749

condiciones. El uso de modelos preentrenados en este contexto facilita una generalización más robusta y una convergencia más rápida durante el proceso de entrenamiento, lo cual resulta ventajoso en escenarios donde los datos etiquetados son limitados [4].

III-B. Tokenización

En el procesamiento del lenguaje natural, cada palabra, signo o símbolo debe transformarse en una representación numérica para que pueda ser comprendida y procesada por los modelos de lenguaje. Este proceso se conoce como *tokenización*, y consiste en dividir el texto en unidades mínimas denominadas *tokens*, que pueden corresponder a palabras, subpalabras o incluso caracteres individuales. A cada token se le asigna un identificador numérico único dentro de un vocabulario previamente definido, lo que permite representar oraciones completas como secuencias de números. Existen diversas estrategias de tokenización, como la basada en subpalabras (*Byte Pair Encoding* o *WordPiece*), que buscan equilibrar la eficiencia del vocabulario con la capacidad del modelo para manejar palabras desconocidas o de diferentes idiomas [5].

La Tabla II muestra un ejemplo simplificado del proceso de tokenización, en el cual cada palabra del texto es descompuesta en su correspondiente token y asociada a un identificador numérico dentro del vocabulario del modelo. Este procedimiento permite representar de forma estructurada los elementos lingüísticos, facilitando que el modelo procese el texto como una secuencia de valores discretos que posteriormente serán transformados en vectores continuos mediante técnicas de *embedding*.

La Tabla III resume algunos de los tipos más comunes de tokenización utilizados en modelos de lenguaje. Cada enfoque difiere en el nivel de granularidad con que divide el texto: desde unidades completas como palabras, hasta fragmentos más pequeños como subpalabras, caracteres o incluso bytes individuales. Esta diversidad de métodos permite adaptar la representación del texto según las necesidades del modelo, equilibrando la complejidad del vocabulario con la capacidad para manejar palabras desconocidas o símbolos especiales.

Cuadro III
TIPOS COMUNES DE TOKENIZACIÓN

Tipo	Ejemplo	Ventaja principal
Palabra	"Los modelos"	Simplicidad
Caracter	"L", .o", "s"	Sin OOV*
Subpalabra	.ªprend-ïendo"	Equilibrio vocabulario/contexto
Byte-level	bytes UTF-8	Soporta cualquier símbolo
Espacio en blanco	"Hola", "mundo"	Rápido y simple

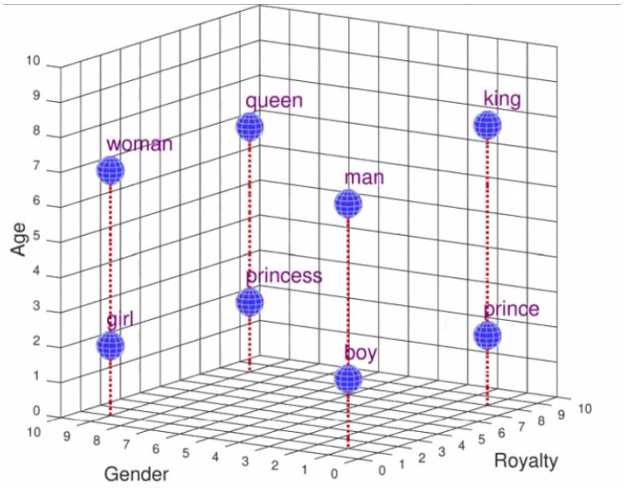

Figura 2. 3D Semantic feature space

III-C. Representación de tokens en un espacio vectorial

Una vez que el texto ha sido tokenizado, cada token se convierte en un número que sirve únicamente como identificador dentro del vocabulario del modelo. Sin embargo, estos valores numéricos carecen de significado semántico por sí mismos, ya que no reflejan las relaciones o similitudes entre las palabras. Para que un modelo pueda comprender el contexto y el significado del lenguaje, es necesario transformar dichos identificadores en representaciones continuas que capturen las propiedades semánticas y sintácticas de las palabras dentro del texto. Este proceso se logra mediante el uso de *embeddings*, los cuales permiten a los modelos de lenguaje aprender representaciones vectoriales que preservan relaciones de significado y proximidad contextual [6].

La Figura 2 representa un espacio vectorial tridimensional en el que las palabras se distribuyen según tres dimensiones semánticas: edad, género y realeza. Cada punto del espacio corresponde a la proyección de una palabra en función de sus características aprendidas por el modelo, lo que permite observar relaciones de similitud y diferencia entre conceptos. Por ejemplo, términos como “rey” y “reina” se ubican próximos entre sí en la dimensión de realeza, pero difieren en la dimensión de género, ilustrando cómo los *embeddings* capturan relaciones semánticas complejas dentro de un espacio continuo [6].

III-D. Similaridad entre vectores

Una vez que las palabras han sido transformadas en vectores dentro de un espacio continuo, es posible cuantificar su grado de similitud midiendo la distancia o el ángulo entre dichos vectores. En este contexto, dos vectores próximos representan palabras con significados semánticamente similares, mientras que aquellos que se encuentran alejados reflejan conceptos distintos o no relacionados. Esta propiedad permite a los modelos de lenguaje capturar relaciones latentes como analogías o asociaciones conceptuales, lo que ha sido fundamental para tareas como la búsqueda semántica, la traducción automática y la inferencia contextual [7].

III-E. *Métricas más comunes*

Las métricas más comunes para calcular similitud entre vectores son:

- Distancia euclidiana:

$$d(a,b) = \sqrt{\sum_i (a_i - b_i)^2} \tag{1}$$

  Mide que tan lejos están los puntos.
- Similitud del coseno

$$sim(a,b) = \frac{a \cdot b}{\|a\|\|b\|} \tag{2}$$

  Mide el ángulo entre vectores: cuanto más pequeño, más similares.

La más usada en modelos de lenguaje es la similitud de coseno, ya que se enfoca en la dirección del vector más que en su magnitud.

IV. Embeddings

Los *embeddings* son representaciones numéricas densas que asignan a cada token —ya sea una palabra, subpalabra o incluso una frase— un vector en un espacio continuo de alta dimensión. Estas representaciones permiten capturar el significado semántico y las relaciones contextuales entre los términos, de modo que palabras con sentidos similares se ubiquen próximas entre sí dentro del espacio vectorial. Además, los modelos modernos son capaces de generar *embeddings* a nivel de frase o enunciado (*sentence embeddings*), los cuales condensan el significado global de un texto. Este tipo de representación posibilita comparar oraciones, ideas o documentos en función de su contenido semántico, en lugar de basarse únicamente en coincidencias literales de palabras [8].

La Figura 3 representa un ejemplo conceptual de embeddings para frases similares en el que se muestran las diferentes fases de forma general y simplificada, pasando des de la palabra, documento u oración, hasta el espacio del embedding.

IV-A. *Capacidad de los modelos de lenguaje*

Gracias a su entrenamiento a gran escala y al uso de arquitecturas basadas en *transformers*, los modelos de lenguaje de gran escala (LLM) han desarrollado un conjunto de capacidades emergentes que trascienden las funciones para las que fueron diseñados explícitamente. Estas habilidades —como el razonamiento contextual, la inferencia lógica o la adaptación a tareas no vistas durante el entrenamiento— no fueron programadas de forma directa, sino que surgen como resultado del aprendizaje de patrones complejos a partir de enormes volúmenes de datos textuales y contextuales. Este fenómeno ha sido objeto de creciente interés, ya que evidencia cómo la escala y la estructura de los modelos pueden dar lugar a comportamientos no lineales y sofisticados en el procesamiento del lenguaje natural [9].

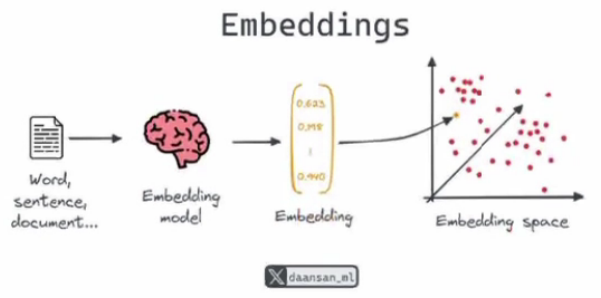

Figura 3. Ejemplo conceptual de embeddings de frases similares

IV-B. *Capacidades de modelos de lenguaje*

- **Comprensión textual**: interpretan el significado de palabras y frases según el entorno en el que aparecen.
- **Generación coherente de texto**: pueden redactar, traducir o resumir información manteniendo estilo y consistencia.
- **Razonamiento y planificación**: resulven problemas, explican pasos y trazan estrategias.
- **Aprendizaje de prompt**: adaptan su comportamiento a partir de ejemplos dados en la misma conversación (in-context learning).
- **Multitarea**: realizan traducción, clasificación, codificación, análisis o dialogo sin requerir reentrenamiento.

IV-C. *Limitación de los modelos de lenguaje*

- **Alucinaciones:** generan respuestas convincentes pero incorrectas o inventadas.
- **Memoria limitada:** no recuerdan interacciones pasadas más allá de su ventana de contexto.
- **Conocimiento estático:** su información proviene de los datos de entrenamiento.
- **Costos computacionales:** requieren grandes recursos para entrenamiento e inferencia.

V. Retrival-Augmented Generation (RAG)

El enfoque de Recuperación Aumentada de Generación (*Retrieval-Augmented Generation*, RAG) combina la potencia generativa de los modelos de lenguaje de gran escala (LLM) con un módulo de recuperación de información externa, conocido como *retriever*. Este componente permite inyectar conocimiento relevante proveniente de bases de datos o colecciones de documentos en el momento de la consulta, ampliando así la capacidad del modelo para generar respuestas más precisas, actualizadas y fundamentadas en evidencia. De esta manera, el sistema integra razonamiento generativo con recuperación informativa, superando las limitaciones de los LLM entrenados únicamente con conocimiento estático [1].

V-A. *Ingesta y Chunking*

El primer paso en la construcción de un sistema de Recuperación Aumentada de Generación (RAG) consiste en preparar los documentos que servirán como fuente de información. Para ello, el texto se segmenta en fragmentos manejables denominados *chunks*, que suelen tener una longitud entre 200 y 500 tokens, con el fin de preservar la coherencia semántica y facilitar la recuperación eficiente de información relevante.

Posteriormente, cada fragmento se transforma en un vector mediante un módulo de *embeddings*, el cual codifica su significado semántico en un espacio de alta dimensión. Esta representación vectorial permite medir similitudes entre consultas y fragmentos de texto, habilitando la búsqueda contextual basada en significado y no en coincidencias literales [10].

V-A1. Chunking Tamaño fijo: El proceso de *chunking* de tamaño fijo consiste en dividir los documentos en segmentos de texto de longitud predefinida, con el objetivo de estandarizar las unidades de información utilizadas en los sistemas de recuperación. Esta técnica permite equilibrar la granularidad del contenido: fragmentos demasiado pequeños pueden perder contexto semántico, mientras que fragmentos demasiado grandes dificultan la búsqueda eficiente y aumentan la ambigüedad en la recuperación. Al mantener un tamaño constante, los *chunks* facilitan la indexación vectorial y mejoran la precisión de los modelos que emplean *embeddings* para comparar consultas y pasajes de texto [11].

V-A2. Chunking recursivo: El *chunking* recursivo es una técnica avanzada utilizada para segmentar texto de manera jerárquica y adaptativa, en lugar de emplear longitudes fijas. Este método divide los documentos siguiendo la estructura lingüística del contenido, como párrafos, oraciones o secciones, y aplica fragmentaciones adicionales cuando un segmento excede un límite de tokens definido. De este modo, se preserva el contexto semántico relevante en cada fragmento, evitando cortes arbitrarios que podrían afectar la coherencia del texto. El enfoque recursivo resulta especialmente útil en tareas de Recuperación Aumentada de Generación (RAG), donde mantener la integridad semántica de los *chunks* mejora significativamente la precisión de la recuperación contextual [12].

V-A3. Chunking similaridad semántica: El *chunking* basado en similitud semántica emplea medidas vectoriales —principalmente la similitud del coseno— para dividir un texto en fragmentos coherentes según su significado, en lugar de hacerlo por longitud o estructura gramatical. En este enfoque, se generan *embeddings* de oraciones o párrafos consecutivos, y se calcula la similitud coseno entre ellos. Cuando la similitud cae por debajo de un umbral predefinido, se considera que el contexto cambia significativamente, estableciendo así un nuevo límite de *chunk*. Este método produce divisiones más naturales desde el punto de vista semántico, preservando la coherencia temática y mejorando la recuperación contextual en sistemas basados en *Retrieval-Augmented Generation (RAG)* [13].

V-B. Indexación

La *indexación vectorial* es un proceso fundamental en los sistemas de recuperación aumentada (*Retrieval-Augmented Generation*, RAG), que permite almacenar y buscar eficientemente representaciones numéricas de documentos o fragmentos de texto en un espacio vectorial de alta dimensión. Su propósito es facilitar la recuperación de información semánticamente similar a una consulta mediante la comparación de vectores utilizando métricas como la similitud coseno o la distancia euclidiana. Entre las soluciones más utilizadas se encuentran **Qdrant**, un motor de búsqueda vectorial de código abierto optimizado para búsquedas por similitud y filtrado híbrido; **Pinecone**, un servicio en la nube que ofrece indexación vectorial escalable y mantenimiento automático de índices; y **FAISS** (*Facebook AI Similarity Search*), una biblioteca desarrollada por Meta que permite búsquedas eficientes en grandes volúmenes de vectores mediante técnicas de cuantización y optimización de memoria. Estas herramientas son esenciales para el funcionamiento de sistemas RAG modernos, al permitir una recuperación rápida y precisa del contexto más relevante [14]–[16].

V-C. Consulta o Recuperación

Una vez construida la base vectorial, el siguiente paso consiste en realizar la recuperación semántica de información. Dada una consulta o pregunta formulada por el usuario, esta se transforma en un *embedding* que captura su significado en un espacio de alta dimensión. Posteriormente, se calcula la similitud —comúnmente mediante la métrica del coseno— entre este vector de consulta y todos los *embeddings* previamente indexados. Finalmente, el sistema devuelve los *top-k* fragmentos más cercanos, es decir, aquellos cuya representación vectorial es más similar a la de la consulta. Este proceso permite realizar búsquedas basadas en el significado semántico del texto, en lugar de depender de coincidencias literales o palabras exactas, lo que mejora significativamente la precisión contextual en aplicaciones basadas en *Retrieval-Augmented Generation (RAG)* [1].

V-D. Augmentación y Generación (Inyección de contexto)

El paso final en un sistema *Retrieval-Augmented Generation (RAG)* consiste en integrar la información recuperada dentro del *prompt* que se enviará al modelo de lenguaje. Para ello, se construye una plantilla o estructura de entrada que combina la pregunta del usuario con los fragmentos de texto más relevantes obtenidos en la fase de recuperación. Este contexto adicional actúa como una fuente de conocimiento explícita que guía al LLM, permitiéndole generar una respuesta más precisa, coherente y sustentada en la evidencia. De esta

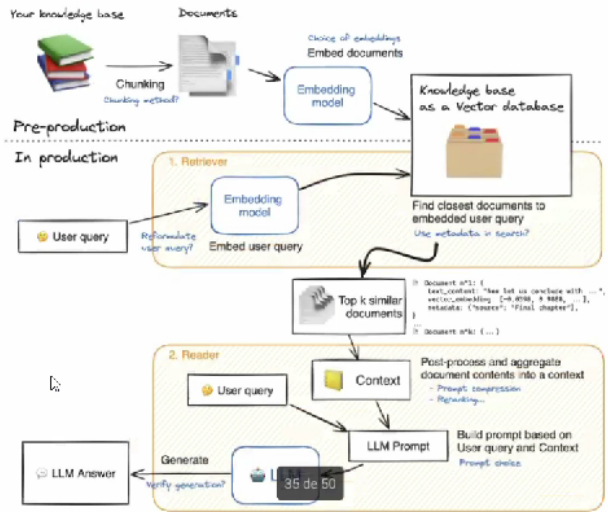

Figura 4. Diagrama del proceso del RAG

manera, el modelo no depende únicamente de su conocimiento preentrenado, sino que se apoya en información actualizada y específica al dominio, lo cual mejora la fiabilidad y reduce la alucinación de respuestas [17].

La Figura 4 ilustra de forma general el funcionamiento del proceso *Retrieval-Augmented Generation* (RAG). Este enfoque combina la recuperación de información relevante desde una base vectorial con la generación de texto asistida por un modelo de lenguaje. A partir de una consulta del usuario, el sistema identifica los fragmentos más relacionados semánticamente, los integra dentro del *prompt* y genera una respuesta fundamentada en dichas evidencias. De esta manera, el modelo puede ofrecer respuestas más precisas, actualizadas y contextualizadas que las obtenidas únicamente a partir del conocimiento interno del LLM [1].

VI. Beneficios

El uso de arquitecturas basadas en *Retrieval-Augmented Generation* (RAG) ofrece múltiples beneficios frente al uso de modelos de lenguaje puros. En primer lugar, permite una significativa **reducción de las alucinaciones**, ya que el modelo genera sus respuestas apoyándose en evidencia documental verificable en lugar de depender únicamente de su conocimiento implícito. Además, posibilita la **actualización continua del conocimiento**, dado que la base de recuperación puede ser renovada con información reciente sin necesidad de reentrenar el modelo. Este enfoque también contribuye a una mayor **eficiencia de costos**, al disminuir la necesidad de entrenamientos extensivos y aprovechar modelos preexistentes combinados con fuentes dinámicas de datos. Finalmente, el paradigma RAG favorece la **aplicabilidad en dominios especializados**, permitiendo adaptar el comportamiento del sistema a contextos como medicina, derecho o ingeniería mediante la incorporación de bases de conocimiento específicas [18].

VII. Casos de uso

Los sistemas basados en *Retrieval-Augmented Generation* (RAG) presentan aplicaciones prácticas en múltiples dominios. Por ejemplo, en el ámbito corporativo, los **asistentes empresariales enriquecidos** pueden ofrecer información precisa y contextualizada a partir de bases de conocimiento internas, mejorando la productividad y la toma de decisiones. En el campo de la **investigación**, los RAG facilitan la recuperación de literatura relevante y la síntesis de información compleja, acelerando el análisis de grandes volúmenes de datos textuales. Asimismo, en el área de **soporte al cliente**, estos sistemas permiten generar respuestas fundamentadas y coherentes a consultas de usuarios, reduciendo errores y mejorando la experiencia de atención mediante información verificada y actualizada [18].

VIII. Tarea 4: Agente Conversacional

Al final de la clase se presentó la asignación y revisión del enunciado de la Tarea 4, centrada en el desarrollo de un agente conversacional. La fecha de entrega se ha establecido para el jueves 6 de noviembre.

Referencias

[1] P. Lewis, E. Perez, A. Piktus, F. Petroni, V. Karpukhin, N. Goyal, H. Küttler, M. Lewis, W. tau Yih, T. Rocktäschel, S. Riedel, and D. Kiela, "Retrieval-augmented generation for knowledge-intensive nlp tasks," *Advances in Neural Information Processing Systems (NeurIPS)*, 2020.

[2] S. Wang, Y. Qin, W. Chen, Z. Wu, Z. Xi, Y. Xu, T. Gui, X. Qiu, and Z. Zhang, "A survey on large language model based autonomous agents," *arXiv preprint arXiv:2401.03428*, 2024.

[3] OpenAI, "Gpt-4 technical report," arXiv preprint arXiv:2303.08774, 2023.

[4] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, pp. 436–444, 2015.

[5] R. Sennrich, B. Haddow, and A. Birch, "Neural machine translation of rare words with subword units," in *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (ACL)*, 2016, pp. 1715–1725.

[6] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient estimation of word representations in vector space," in *Proceedings of the International Conference on Learning Representations (ICLR)*, 2013.

[7] J. Pennington, R. Socher, and C. D. Manning, "Glove: Global vectors for word representation," in *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2014, pp. 1532–1543.

[8] N. Reimers and I. Gurevych, "Sentence-bert: Sentence embeddings using siamese bert-networks," in *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2019, pp. 3982–3992.

[9] J. Wei, Y. Tay, R. Bommasani, C. Raffel, B. Zoph, S. Borgeaud, D. Yogatama, M. Bosma, D. Zhou, D. Metzler, E. Chi, T. Hashimoto, O. Vinyals, P. Liang, J. Dean, and W. Fedus, "Emergent abilities of large language models," *arXiv preprint arXiv:2206.07682*, 2022.

[10] V. Karpukhin, B. Oguz, S. Min, P. Lewis, L. Wu, S. Edunov, D. Chen, and W. tau Yih, "Dense passage retrieval for open-domain question answering," in *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2020, pp. 6769–6781.

[11] G. Izacard and E. Grave, "Leveraging passage retrieval with generative models for open domain question answering," in *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics (EACL)*, 2021, pp. 874–880.

[12] L. Gilardi and D. Steiner, "Recursive chunking strategies for improved context retrieval in large language models," arXiv preprint arXiv:2309.02706, 2023.

[13] Y. Liu, S. Kumar, and P. Gupta, "Semantic chunking with cosine similarity for enhanced context preservation in rag systems," arXiv preprint arXiv:2403.11892, 2024.

[14] J. Johnson, M. Douze, and H. Jégou, "Billion-scale similarity search with gpus," *IEEE Transactions on Big Data*, vol. 7, no. 3, pp. 535–547, 2019.

[15] P. S. Inc., "Pinecone documentation," https://docs.pinecone.io/, 2024.

[16] Q. Team, "Qdrant: Vector database documentation," https://qdrant.tech/documentation/, 2024.

[17] G. Izacard, P. Lewis, M. Lomeli, L. Hosseini, F. Petroni, T. Schick, S. Riedel, and D. Kiela, "Atlas: Few-shot learning with retrieval augmented language models," *arXiv preprint arXiv:2208.03299*, 2022.

[18] Y. Gao, S. Li, J. Lin, J. Callan *et al.*, "Retrieval-augmented generation for large language models: A survey," *arXiv preprint arXiv:2312.10997*, 2023.