

TRABAJO FIN DE GRADO

INGENIER´IA INFORMÁTICA

**Modelos de Generación de Texto en Bases de Datos Orientadas a Grafos**

**Autor**

Alberto López Povedano

**Directores**

Juan Francisco Huete Guadix

**Ciencias de la computación e Inteligencia Artificial**



Escuela Técnica Superior de Ingenierías Informática y de Telecomunicación

—

Granada, 8 de julio de 2022

**Modelos de Generación de Texto**

Alberto López Povedano

**Palabras clave:** orientado a grafos, lenguaje natural, generación de texto, grafos, inteligencia artificial

**Resumen:**

Desarrollo de un sistema recomendador de texto basado en la implementación de un n-grama para la modelización del lenguaje castellano.

Yo, **Alberto López Povedano**, alumno de la titulación **Ingeniería Informática de**

**la Escuela Técnica Superior de Ingenierías Informática y de Telecomunicación** de la Universidad de Granada, con DNI 77448870G, autorizo la ubicación de la siguiente

copia de mi Trabajo Fin de Grado en la biblioteca del centro para que pueda ser

consultado por las personas que lo deseen.

Fdo: Alberto López Povedano

Granada a 08 de Julio de 2022

D. **Juan Francisco Huete Guadix**, Catedrático en el departamento de Ciencias de la Computación e Inteligencia Artificial en la Universidad de Granada.

**Informan:**

Que el presente trabajo, titulado **Modelos de Generación de Texto Basados en Bases de Datos Orientados a Grafos**, ha sido realizado bajo su supervisión por **Alberto López Povedano**, y autorizamos la defensa de dicho trabajo ante el tribunal que corresponda.

Y para que conste, expiden y firman el presente informe en Granada a 6 de julio de

2020.

**Los directores:**

**Juan Francisco Huete Guadix**

**Agradecimientos**

A Juan Francisco Huete Guadix por la supervisión en la realización del trabajo de final de grado.

A Ernesto Martínez del Prieto por su inestimable ayuda ofreciéndome recursos didácticos, asesoramiento y recomendaciones en la realización del trabajo de final de grado.

Índice

[Figuras 13](#_Toc115966805)

[Términos 14](#_Toc115966806)

[Abreviaciones 15](#_Toc115966807)

[1. Introducción 16](#_Toc115966808)

[1.1. Propósito 16](#_Toc115966809)

[1.2. Justificación 16](#_Toc115966810)

[1.3. Objetivos 17](#_Toc115966811)

[1.3.1. Objetivos Primarios 17](#_Toc115966812)

[1.3.2. Objetivos Secundarios 17](#_Toc115966813)

[1.3.3. Requisitos no funcionales 17](#_Toc115966814)

[1.4. Metodología 17](#_Toc115966815)

[1.4.1. Terminología 18](#_Toc115966816)

[1.5. Estado del arte 18](#_Toc115966817)

[1.5.1. GPT-3 18](#_Toc115966818)

[2. Fundamentos Teóricos 19](#_Toc115966819)

[2.1. Modelación del lenguaje 19](#_Toc115966821)

[2.1.1. Modelos de N-gramas 19](#_Toc115966822)

[2.2. Grafos 20](#_Toc115966823)

[3. GPG – Grafo Preentrenado Generativo 22](#_Toc115966824)

[3.1. Datos de entrenamiento 22](#_Toc115966826)

[3.2. Estructura del sistema 22](#_Toc115966827)

[3.2.1. Estructura de directorios 22](#_Toc115966828)

[3.2.2. Estructura de datos 22](#_Toc115966829)

[3.3. Resultados 22](#_Toc115966830)

[3.3.1. Predicción 1: Palabra más frecuente 22](#_Toc115966831)

[3.3.2. Predicción 2: Palabra según distribución 22](#_Toc115966832)

[3.3.3. Predicción 3: Recorrido entre dos palabras 22](#_Toc115966833)

[4. Conclusiones 23](#_Toc115966834)

[4.1. Conclusiones sobre objetivos 23](#_Toc115966836)

[4.2. Trabajo Futuro 23](#_Toc115966837)

[4.3. Valoración personal 23](#_Toc115966838)

[5. Tecnologías 24](#_Toc115966839)

[5.1. Python 24](#_Toc115966841)

[5.2. Docker 25](#_Toc115966842)

[5.3. ArangoDB 25](#_Toc115966843)

[5.4. Git 25](#_Toc115966844)

[6. Bibliografía 27](#_Toc115966845)

# Figuras

[Figura 1.1: Tablero Kanban a día 14/08/2022 18](#_Toc115966846)

[Figura 2.1: Ejemplo ilustrado de formación de n-gramas 20](#_Toc115966847)

[Figura 2.2: Visualización de grafo de ejemplo 21](#_Toc115966848)

# Términos

**Chatbot** Sistema software cuyo propósito es simular conversaciones con

usuario humano.

**Sprint** Ciclo de tiempo acotado por un equipo en el desarrollo del software

siguiendo los principios de las metodologías ágiles.

**Corpus** Colección de texto o audio oficial perteneciente a un lenguaje

organizado en conjuntos de datos.

**Tokenización** Acción de convertir datos complejos en una unidad individual

conocida como token.

**Token** Representación tangible de un hecho, atributo o sentimiento.

**Smoothing** En el terreno estadístico hace referencia al hecho de crear una

función que otorgue los considerados como patrones importantes en los

datos a la vez que deja lo considerado como ruido fuera de ellos.

---

**Plugin** Extensión de un programa software que añade nueva funcionalidad sin

alterar la proporcionada por el programa original.

**Object Oriented Programming** Modelo de programación que organiza el

diseño del software en torno a la idea de objeto, una entidad que

contiene datos y código en forma de métodos.

**Object-Relational Mapping** Método para convertir los datos almacenados en

una base de datos a un modelo definido en un lenguaje orientado a

objetos.

# Abreviaciones

**-----**

**OOP** Object Oriented Programming.

**ORM** Object-Relational Mapping.

# Introducción

Los primeros sistemas conversaciones datan de 1966 y 1971, con los *chatbots* ELIZA y PARRY respectivamente. Ambos sistemas tuvieron cierta relevancia y plantearon nuevas cuestiones éticas en cuanto a la privacidad de datos de los usuarios, sin embargo, fue con el desarrollo de la arquitectura del sistema GUS (1977) que se establecería el paradigma predominante en el desarrollo de sistemas conversaciones durante los 30 próximos años.

(Jurafsky & Martin, 2022)

Las aplicaciones del procesamiento del lenguaje natural por una computadora son increíblemente extensas, por mencionar alguna de ellas:

* Traducción automática de textos
* Clasificación de información
* Detección de estados emocionales
* Generación de texto
* Etcétera

Por sus grandes aplicaciones y debido al gran interés que despertó el terreno de la inteligencia artificial, el procesamiento del lenguaje natural se ha convertido en un campo de la inteligencia artificial con décadas de estudio a sus espaladas. Varias son las técnicas que se han aplicado en su investigación, hablamos de sistemas basados en reglas, aprendizaje automático mediante métodos estadísticos y redes neuronales.

En particular la modelización del lenguaje, una técnica de aprendizaje automático, ha sido la seleccionada para el desarrollo de este proyecto.

## Propósito

El propósito de este trabajo es el de crear un software que nos sugiera recomendaciones, en forma de palabras u oraciones, en función de las palabras introducidas por el usuario. Este proyecto surge de la idea de crear un sistema que facilite la escritura de textos en un idioma distinto al del usuario. Idea propuesta por el antiguo trabajador de la empresa 8Belts Ernesto Martínez del Prieto. Al día de publicarse, y durante gran parte del desarrollo, no existe ningún enlace entre este trabajo y dicha empresa.

## Justificación

Las razones que me han llevado a seleccionar este tema para el desarrollo de mi tesis de final de grado son:

* El interés y deseo de introducirse en el estudio de las técnicas de aprendizaje automático.
* Mejorar mi habilidad con el lenguaje de programación Python y la tecnología de contenedores Docker.

## Objetivos

Teniendo como objetivo general de este proyecto aprender, reforzar, y poner en práctica los conocimientos más fundamentales de los modelos de aprendizaje por métodos estadísticos, mediante la propuesta y desarrollo de un sistema básico de generación de texto, se plantean los siguientes objetivos:

### Objetivos Primarios

1. Obtener una **fuente de datos** para el entramiento del modelo.
2. Implementar una forma de leer y almacenar los datos en una **base de datos** adecuada.
3. Representar los datos en una **estructura de datos** que nos permita representar las conexiones entre las diferentes palabras.
4. Ofrecer sugerencias a las entradas de datos proporcionadas por los usuarios mediante **tres diferentes modos de predicción**.

### Objetivos Secundarios

1. Configurar un **contenedor** que permita un despliegue rápido y sencillo de la base de datos.

### Requisitos no funcionales

Durante el proceso de desarrollo al igual que se consiguen los objetivos mencionados se han de tener los siguientes principios en mente:

1. Como en todo proyecto software se ha de mantener una estructura de ficheros y directorios que sea clara y organizada. Esta es detallada en la Sección 3.2.1
2. El código ha de ser reutilizable y sencillo de comprender. La longitud de comentarios en el texto ha de ser breve y resaltar aquello que no se puede extraer con facilidad al leer el código.
3. El sistema ha de ser poder desplegado con relativa facilidad en diferentes equipos haciendo pocas instalaciones de dependencias, para facilitar tanto el desarrollo como para posibilitar el uso por diferentes usuarios.

## Metodología

La metodología utilizada en el desarrollo del proyecto no ha respetado estrictamente las normas y formas de proceder de una metodología ya conocida, pero si se ha inspirado y aproximado al terreno de las conocidas como metodologías ágiles.

En este grupo encontramos principios como la organización del trabajo en torno a *sprints,* identificar el trabajo en pequeñas tareas lo más atómicas posibles otorgándoles una estimación de tiempo o el uso de herramientas como los tableros Kanban.

De las prácticas mencionadas anteriormente se ha llevado a cabo principalmente la definición de pequeñas tareas a realizar siendo representadas en tarjetas de un tablero Kanban proporcionado por la plataforma Trello. A continuación, en la Figura 1.1

se aporta un ejemplo de cómo se vería el tablero Kanban utilizado:

Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente

Figura 1.1: Tablero Kanban a día 14/08/2022

### Terminología

Con el fin de evitar malentendidos y búsquedas confusas de términos usados en este documento se utilizarán términos en inglés, especialmente en los casos que la traducción de un término al castellano no produzca búsquedas satisfactorias o sea incluso confundido con acepciones relativas a otros campos del conocimiento. La terminología se utiliza, así, en castellano e inglés indistintamente. A la hora de utilizar términos en inglés, estos serán escritos utilizando cursiva.

En las primeras secciones no numeradas de este documento se han incluido un listado de Términos y otro de Abreviaciones, que exponen de forma breve los conceptos referenciados.

## Estado del arte

\*\* Mencionar transformadores

### GPT-3

\*\* Hablar sobre el modelo GPT-3 desarrollado por la empresa OpenAI

# Fundamentos Teóricos



## Modelación del lenguaje

La modelación del lenguaje es un término que engloba todos aquellos métodos que intentan solucionar el problema de modelar el lenguaje natural. La modelación del lenguaje tiene como objetivo permitir el aprendizaje de la maquina del lenguaje asociando altas probabilidades a frases bien estructuradas y formadas.

En este terreno frecuentemente se asume la asunción de Márkov. Según (Venkat N. Gudivada, 2015) la asunción de Márkov indica, en el contexto de modelos del lenguaje, que al predecir la siguiente palabra en una secuencia esta depende directamente de la palabra previa o secuencia de palabras previas, siendo un número pequeño de palabras. El tamaño de indica el -orden del proceso de Márkov. Escrito en forma de ecuación:

Entendiendo que hace referencia a la probabilidad de ver tras la secuencia de palabras

Dentro de la modelación del lenguaje se han desarrollado varios modelos diferentes con sus propias ventajas y desventajas. Encontramos modelos exponenciales, modelos de n-gramas y modelos del lenguaje neuronales, siendo estos dos últimos aquellos con más relevancia.

### Modelos de N-gramas

En los modelos de n-gramas se asume la asunción de Márkov y se trabaja con una distribución de probabilidad sobre las secuencias de palabras o caracteres. Estos modelos se caracterizan por trabajar a base de n-gramas. Los n-gramas son conjuntos de datos formados por palabras, caracteres o sílabas, esto depende en la *tokenización* elegida a la hora de analizar el *corpus.*

Dependiendo de la *tokenización* elegida se pueden clasificar diferentes tipos de modelos. Lo más conocidos son los que dividen el *corpus* en palabras haciendo uso de espacios y signos de puntuación como separadores. Diferenciamos entre unigramas, bigramas, trigramas, … en función del número de palabras que formen el n-grama. Un ejemplo de lo explicado se puede visualizar en la Figura 2.1:

Texto

Descripción generada automáticamente

Figura 2.1: Ejemplo ilustrado de formación de n-gramas

Los modelos de n-gramas afrontan el problema conocido como *data sparsity* el cual aparece al querer tener un mayor tamaño en los n-gramas construidos, pues el modelo tendrá un mayor parecido con la realidad. El problema de intentar hacer los n-gramas de gran tamaño es que cuanto mayor es el tamaño más posibilidad hay de que aparezcan n-gramas que no existan para ser observados en el conjunto de datos utilizado para el entrenamiento. Para enfrentar este problema se hace uso comúnmente de técnicas de *smoothing*.

En cambio, podemos encontrar los modelos de lenguaje neuronales como una solución al problema mencionado anteriormente. Pues con el modelo expuesto en (Vincent, 2003) se introdujo la idea de que las redes neuronales son capaces de aprender representaciones similares de palabras dentro de un contexto, otorgando así respuestas a n-gramas que no se hayan observado en el *corpus* de entrenamiento usado.

Para el desarrollo de este proyecto se ha construido un modelo del lenguaje a base de unigramas, en este caso el problema de *data sparsity* puede ser solucionado con una muestra de datos lo suficientemente alta pues se considera cada palabra individualmente. El problema de este enfoque mediante unigramas es que asumimos que toda palabra es independiente lo cual puede dar malos resultados en modos de predicción que sugieran frases de palabras individuales o sugiriendo palabras individuales ante una secuencia de palabras que establezca un contexto.

## Grafos

Los n-gramas y las conexiones entre ellos pueden ser almacenados y visualizados como un grafo. Un grafo es una estructura constituida por nodos y arcos entre esos nodos. El ámbito de estudio de los grafos es conocido como teoría de Grafos, la cual es una rama de la matemáticas y ciencias de la computación.

En este caso las palabras serían representadas por los nodos y los arcos establecerían relaciones de sucesión entre ellas, es decir, imaginemos el nodo que simboliza a la palabra “por” dicho nodo podría presentar arcos que lo relacionaría con las palabras “lo” y “ejemplo” tras haber analizado las frases: “… por lo tanto esta comida está fría” y “por ejemplo, aquí tienes mi estuche”, pues como se puede ver la palabra “lo” y “ejemplo” suceden a la palabra “por”. La *Figura 2.2* ilustra el ejemplo descrito, además nótese como los arcos reflejan conexiones unidireccionales, si estudiáramos una sentencia donde “lo” precediera a “por” entonces el arco entre ellas pasaría a reflejar una conexión bidireccional.

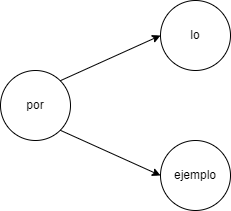


Figura 2.2: Visualización de grafo de ejemplo

La representación de las conexiones entre unigramas puede ser aplicable a muchas estructuras de datos diferentes, matrices de vectores, vectores de instancias de objetos, etcétera. La razón por la cuál se ha escogido el grafo como estructura para representar dichas conexiones es dado a la facilidad que presenta para recorrerlo, sobre todo teniendo en cuenta que la base de datos elegida para el proyecto es una base de datos orientada a grafos.

# GPG – Grafo Preentrenado Generativo



## Datos de entrenamiento

## Estructura del sistema

### Estructura de directorios

### Estructura de datos

\*\* Especificar el proceso de análisis por el cual el conjunto de datos es analizado y almacenado en la base de datos ArangoDB.

\*\* Explicar como se calcula el valor y atributo de los arcos del grado. Escribir formula.

\*\* Especificar una tabla con los n-gramas más comunes obtenidos tras analizar los libros.

## Resultados

\*\* Métodos de predicción realizados, ofrecer lista de pasos de como se procede en cada método.

### Predicción 1: Palabra más frecuente

### Predicción 2: Palabra según distribución

### Predicción 3: Recorrido entre dos palabras

# Conclusiones



## Conclusiones sobre objetivos

## Trabajo Futuro

A continuación, se listan una serie de objetivos que de aplicarse podrían mejorar el producto final con el que se ha concluido este proyecto.

1. Integración del sistema en herramientas de escritura normalmente utilizadas en forma de *plugin* o en un entorno especifico que permita su utilización sin necesidad de uso mediante terminal. Este punto es realmente importante si se pretende llevar al proyecto a un punto donde pueda ser usado por el usuario promedio.
2. Añadir nuevos métodos de predicción. Sería útil que el sistema autocompletara frases tras una introducción de una secuencia de caracteres.
3. Implementar un modelo del lenguaje basado en n-gramas que utilice trigramas y bigramas en vez de únicamente unigramas.
4. Solventar el problema de *data sparsity* que se originaría al hacer uso de bigramas o trigramas. Como se expone en (Suzuki, 2019) esto puede ser realizado con los ejemplos generados por una red neuronal para expandir el conjunto de datos para entrenar al modelo basado en n-gramas

## Valoración personal

# Tecnologías

En este capítulo se introducirán las distintas tecnologías utilizadas en el desarrollo del proyecto.

La elección del lenguaje es una decisión fundamental. En este caso los lenguajes más populares cuentan con las librerías más completas y diversas. Enfocándonos en el ámbito de análisis de datos C++, Python o Java presentan muy buenas opciones. En este caso se ha decidido optar por Python debido a su facilidad de aprendizaje y de uso.

El uso de la base de datos ArangoDB se debe a que nos aporta la posibilidad de producir modelos de grafos de los datos almacenados en la misma, quitando dicha carga de programación en la parte del lenguaje. Relacionado con el uso de la misma se ha usado Docker para poder aislar dicha tecnología en un contenedor y permitir su despliegue en distinto equipos de trabajo.



## Python

Python es un lenguaje de programación interpretado creado usando C y C++ por Guido van Rossum. Python es un lenguaje multiparadigma, es decir, podemos programar desde diferentes paradigmas de programación, como OOP o programación funcional.

Se caracteriza por ser uno de los lenguajes más sencillos de utilizar. El código escrito en Python es legible y se asemeja al lenguaje humano. También presenta reglas de sintaxis más laxas que otros lenguajes populares como C++.

Python también es uno de los lenguajes más utilizados por programadores en 2022, como nos indica la entrevista anual realizada por Stack Overflow.

(Stack Overflow, 2022)

La versión de Python utilizada es 3.10.2. Para realizar una sencilla, instalación y gestión de paquetes se ha usado el famoso gestor de paquetes Pip en su versión 21.2.4.

Las librerías instaladas utilizadas para el desarrollo del sistema han sido:

* **Python-Arango**: Implementa un *ORM* proporcionándonos una interfaz para interactuar con la base de datos desde el lenguaje. Versión 7.3.1
* **Python-Dotenv**: Habilita el uso de variables de entorno. Usado para poder desarrollar en diferentes equipos de trabajo. En este caso contiene las direcciones absolutas de los libros a leer, ya que estás rutas cambian según el dispositivo en el que trabajemos. Versión 0.20.0
* **SciPy**: Implementa una gran cantidad de algoritmos y modelos matemáticos. En concreto estamos utilizando esta librería por el módulo de estadística de la misma para poder generar una distribución discreta. Versión 1.9.1

## Docker

Docker es un software de código abierto, en cuyo desarrollo encontramos que han participado grandes titanes de la industria, hablamos de Microsoft, Cisco Systems, Google, IBM, o Red Hat.

Docker intenta resolver el famoso problema en el de desarrollo de aplicaciones software resumido en la frase “en mi ordenador si funciona”. Esta tecnología nos permite aislar conjuntos de dependencias en los conocidos como “contenedores Docker”. En estos contenedores podemos desarrollar y ejecutar nuestros proyectos, evitando posibles conflictos con otras librerías o programas que pudiéramos haber instalado para previos desarrollos.

Pero sobre todo una de las características clave que ha hecho que sea tan popular es que permite hacer esto de manera automática, es decir, una vez realizada la configuración de dicho contendor el mismo puede ser utilizado en cualquier otro dispositivo que presente el mismo sistema operativo y el proyecto Docker instalado.

En este proyecto se ha utilizado Docker para poder realizar un despliegue automático de la base de datos Arango en diferentes equipos.

## ArangoDB

La base de datos Arango es una base de datos multimodelo lo cual permite su utilización como una base de datos orientada a grafos, una base de datos orientada a documentos y una base de datos orientada a parejas clave-valor.

Aparte Arango cuenta con su propio lenguaje de *queries* (consultas), común a los tres modelos mencionados anteriormente. Dicho lenguaje aspira por parecerse a los lenguajes de alto nivel multipropósito, utilizando palabras reservadas similares como, por ejemplo, *for* para iterar entre los documentos de una colección en la base de datos. Gracias a esto la familiarización con sus consultas resulta más fácil.

Debido a dichas características ArangoDB es una base de datos realmente flexible, que permite una gran eficiencia y escalabilidad horizontal en las aplicaciones donde sea utilizado.

En este caso su uso ha resultado muy conveniente en el desarrollo del modelo, la base de datos nos proporciona las herramientas para el almacenamiento y tratamiento de los datos en un grafo, incorporando herramientas recorrer al mismo.

## Git

Git es un sistema de control de versiones ampliamente conocido por los desarrolladores. Este es de código abierto y libre.

Normalmente un sistema de control de versiones suele ser utilizado junto a un servicio de almacenamiento en la nube de esas mismas versiones. Entre lo más conocidos encontramos servicios como Github, Gitlab o BitBucket. Para facilitar el desarrollo entre diferentes equipos se ha creado un repositorio público en la plataforma Github al que se puede acceder buscando “alberto-lopov/bachelor-thesis” o mediante el siguiente hiperenlace: <https://github.com/alberto-lopov/bachelor-thesis>

# Bibliografía

Jurafsky, D., & Martin, J. H. (2022). Speech and Language Processing. En *Bibliographical and Historical Notes* (pág. 552).

Stack Overflow. (2022). *Stack Overflow Developer Survey.* Obtenido de https://survey.stackoverflow.co/2022/

Suzuki, M. a. (2019). Improvements to N-gram Language Model Using Text Generated from Neural Language Model. En *ICASSP 2019 - 2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)* (págs. 7245-7249).

Venkat N. Gudivada, D. R. (2015). Handbook of Statistics. En V. V. Venu Govindaraju, *Chapter 9 - Big Data Driven Natural Language Processing Research and Applications* (págs. 203-238). Elsevier.

Vincent, J. (2003). Journal of Machine Learning Research. En D. R. Bengio Y, *A neural probabilistic language model* (págs. 1137-1155).