Generación de texto para historias de vida basadas en narrativa

Trabajo de Fin de Grado
Curso 2021–2022

Autor
María Cristina Alameda Salas

Director
Raquel Hervás Ballesteros
Gonzalo Méndez Pozo

Grado en Ingeniería Informática
Facultad de Informática
Universidad Complutense de Madrid

Generación de texto para historias de vida basadas en narrativa

Trabajo de Fin de Grado en Ingeniería Informática

Departamento de Ingeniería del Software e Inteligencia Artificial

Autor

María Cristina Alameda Salas

Director

Raquel Hervás Ballesteros

Gonzalo Méndez Pozo

Convocatoria: *Febrero/Junio/Septiembre 2022*

Calificación: *Nota*

Grado en Ingeniería Informática

Facultad de Informática

Universidad Complutense de Madrid

17 de abril de 2022

Dedicatoria

Texto de la dedicatoria...

Agradecimientos

Texto de los agradecimientos

Resumen

Resumen en español del trabajo

Palabras clave

Máximo 10 palabras clave separadas por comas

Abstract

Abstract in English.

Keywords

10 keywords max., separated by commas.

Índice

1. Introduction **1**

1. Introducción **3**

1.1. Motivación 3

1.1.1. Explicaciones adicionales 3

1.1.1.1. Texto de prueba 3

2. Estado de la Cuestión **9**

2.1. Alzheimer e historias de vida 9

2.1.1. Descripción general 10

2.1.2. Síntomatología y pérdida de la memoria 11

2.1.3. Tratamientos: historias de vida 12

2.2. Generación de lenguaje natural 14

2.2.1. Generación *text-to-text* (T2T) 15

2.2.2. Generación *data-to-text* (D2T) 16

2.3. Arquitectura tradicional y herramientas de un sistema GLN 16

2.3.1. Macroplanificación 18

2.3.1.1. Selección del contenido 18
2.3.1.2. Estructuración del documento 19
2.3.2. Microplanificación 20
2.3.2.1. Agregación de oraciones 20
2.3.2.2. Lexicalización 20
2.3.2.3. Generación de expresiones de referencia 21
2.3.3. Realización 21
2.3.3.1. Realización lingüística 22
2.3.3.2. Realización de la estructura 22
2.3.4. Herramientas en la arquitectura tradicional 22
2.4. Algoritmos y modelos para la generación de lenguaje 25
2.4.1. Historia 25
2.4.2. Modelos estadísticos 27
2.4.2.1. Modelo Markov Chain 27
2.4.2.2. Modelo N-gram 29
2.4.3. Algoritmos de *Deep Learning* par la generación de texto 31
2.4.3.1. Redes Neuronales Recurrentes (RNNs) 31
2.4.3.2. *Long Short-Term Memory (LSTM)* 33
2.4.3.3. Modelos Seq2Seq y mecanismos de atención 34
2.4.3.4. Modelos preentrenados: Transformers 37
2.4.4. Otras técnicas de *Deep Learning* aplicadas al modelado de lenguaje 39
2.4.4.1. Redes Neuronales Convolucionales (CNNs) 39
2.4.4.2. Redes Neuronales Generativas Adversarias (GANs) . 43

2.4.5. Efectos de las aplicación de redes neuronales a la generación de texto . . . 44
2.4.5.1. Alucinaciones . . . 45
2.4.5.2. Degeneración . . . 48
2.4.5.3. Falta de representación de los datos de entrada . . . 49
2.5. Proyectos relacionados . . . 49

3. Planteamiento e hipótesis de trabajo **53**
3.1. Requisitos . . . 53
3.2. Hipótesis de trabajo . . . 53

4. Conjuntos de datos y preparación para el entrenamiento **55**
4.1. Wiki2bio . . . 56
4.2. WebNLG . . . 58
4.3. KEML . . . 60

5. Modelado de lenguaje y *finetuning* **65**
5.1. Modelos pre-entrenados: Transformers . . . 65
5.2. Tipos de modelos de lenguaje . . . 68
5.2.1. *Casual Language Models* (CLM) . . . 69
5.2.2. *Masked Language Models* (MLM) . . . 70
5.3. Aprendizaje por transferencia (*Transfer Learning*) . . . 71
5.3.1. *One-shot Learning* . . . 72
5.3.2. *Zero-shot Learning* . . . 72
5.3.3. *Multi-task Learning* . . . 73
5.4. *Finetuning* como estrategia del Aprendizaje por Transferencia . . . 73

5.4.1. Pre-entrenamiento del modelo 73
5.4.1.1. GPT-2 74
5.4.1.2. BERT 76
5.4.2. *Supervised fine-tuning* 80
5.5. T5 (*Text-to-Text Transfer Transformer*) 80
5.5.1. Arquitectura 80
5.5.2. Pre-entrenamiento 81

6. Clustering **83**

7. Interfaz y resultados **85**

8. Conclusiones y Trabajo Futuro **87**

9. Modelos de lenguaje aplicados a la generación a partir de datos biográficos **89**
9.1. Ajuste de los modelos de lenguaje 89
9.2. Wiki2bio 90
9.3. KEML 93
9.4. WebNLG 94

9. Conclusions and Future Work **97**

A. Título **99**

B. Título **101**

Bibliografía **111**

Índice de figuras

2.1. Reducción del cerebro asociada al Alzheimer (Mattson, 2004) 11

2.2. Sistema *data-to-text* FoG 17

2.3. Ejemplo de D2T utilizado por Sai et al. (2020) 18

2.4. Arquitectura de referencia para sistema GLN (Vicente et al., 2015) . . . 19

2.5. Entrada y salida de la herramienta SimpleNLG 24

2.6. Etapas del Procesamiento de Lenguaje Natural 26

2.7. Ejemplo de cadena de Markov (Make School, 2017) 29

2.8. Neurona recurrente desplegada en el tiempo 32

2.9. Neurona LSTM 34

2.10. Arquitectura de un sistema Seq2Seq 35

2.11. Arquitectura de un sistema Seq2Seq con mecanismo de atención 37

2.12. Modelos surgidos a partir de BERT 40

2.13. Primera Convolución de Red Neuronal Convolucional 42

2.14. Segunda Convolución de Red Neuronal Convolucional 42

2.15. Arquitectura de una CNN 43

2.16. Alucinaciones en distintos sistemas 46

2.17. Tipos de alucinaciones 47

2.18. Ejemplo de degeneración con Beam Search 48

2.19. Arquitectura del sistema DICE 50

2.20. Entrada y salida del sistema T5 50

2.21. Arquitectura modelo conversacional (de Jesús y García, 2020). 51

4.1. Ejemplo de entrada de wiki2bio 57

4.2. Ejemplo de wiki2bio con y sin limpieza de datos 58

4.3. Ejemplo de WebNLG 59

4.4. Ejemplo procesado de WebNLG 61

4.5. Diversos ejemplos del conjunto de datos KELM 62

5.1. Capa de atención de *Transformers* (Zhou et al., 2019) 67

5.2. Arquitectura del modelo Transformer 68

5.3. Distintos tamaños de GPT-2 y número de decodificadores que emplean 70

5.4. Arquitectura BERT para MLM 77

5.5. Constitución de la entrada del modelo BERT 78

9.1. Ejemplo de entrada de wiki2bio 92

9.2. Resultados de ajuste de GPT-2 en Wiki2bio 95

Índice de tablas

Chapter 1

Introduction

Capítulo 1

Introducción

"Frase célebre dicha por alguien inteligente"
— Autor

1.1. Motivación

1.1.1. Explicaciones adicionales

Si quieres cambiar el **estilo del título** de los capítulos, abre el fichero `TeXiS\TeXiS_pream.tex` y comenta la línea `\usepackage[Lenny]{fncychap}` para dejar el estilo básico de LaTeX.

Si no te gusta que no haya **espacios entre párrafos** y quieres dejar un pequeño espacio en blanco, no metas saltos de línea (`\\`) al final de los párrafos. En su lugar, busca el comando `\setlength{\parskip}{0.2ex}` en `TeXiS\TeXiS_pream.tex` y aumenta el valor de 0,2*ex* a, por ejemplo, 1*ex*.

El siguiente texto se genera con el comando `\lipsum[2-20]` que viene a continuación en el fichero .tex. El único propósito es mostrar el aspecto de las páginas usando esta plantilla. Quita este comando y, si quieres, comenta o elimina el paquete *lipsum* al final de `TeXiS\TeXiS_pream.tex`

1.1.1.1. Texto de prueba

Nam dui ligula, fringilla a, euismod sodales, sollicitudin vel, wisi. Morbi auctor lorem non justo. Nam lacus libero, pretium at, lobortis vitae, ultricies et, tellus. Donec

aliquet, tortor sed accumsan bibendum, erat ligula aliquet magna, vitae ornare odio metus a mi. Morbi ac orci et nisl hendrerit mollis. Suspendisse ut massa. Cras nec ante. Pellentesque a nulla. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Aliquam tincidunt urna. Nulla ullamcorper vestibulum turpis. Pellentesque cursus luctus mauris.

Nulla malesuada porttitor diam. Donec felis erat, congue non, volutpat at, tincidunt tristique, libero. Vivamus viverra fermentum felis. Donec nonummy pellentesque ante. Phasellus adipiscing semper elit. Proin fermentum massa ac quam. Sed diam turpis, molestie vitae, placerat a, molestie nec, leo. Maecenas lacinia. Nam ipsum ligula, eleifend at, accumsan nec, suscipit a, ipsum. Morbi blandit ligula feugiat magna. Nunc eleifend consequat lorem. Sed lacinia nulla vitae enim. Pellentesque tincidunt purus vel magna. Integer non enim. Praesent euismod nunc eu purus. Donec bibendum quam in tellus. Nullam cursus pulvinar lectus. Donec et mi. Nam vulputate metus eu enim. Vestibulum pellentesque felis eu massa.

Quisque ullamcorper placerat ipsum. Cras nibh. Morbi vel justo vitae lacus tincidunt ultrices. Lorem ipsum dolor sit amet, consectetuer adipiscing elit. In hac habitasse platea dictumst. Integer tempus convallis augue. Etiam facilisis. Nunc elementum fermentum wisi. Aenean placerat. Ut imperdiet, enim sed gravida sollicitudin, felis odio placerat quam, ac pulvinar elit purus eget enim. Nunc vitae tortor. Proin tempus nibh sit amet nisl. Vivamus quis tortor vitae risus porta vehicula.

Fusce mauris. Vestibulum luctus nibh at lectus. Sed bibendum, nulla a faucibus semper, leo velit ultricies tellus, ac venenatis arcu wisi vel nisl. Vestibulum diam. Aliquam pellentesque, augue quis sagittis posuere, turpis lacus congue quam, in hendrerit risus eros eget felis. Maecenas eget erat in sapien mattis porttitor. Vestibulum porttitor. Nulla facilisi. Sed a turpis eu lacus commodo facilisis. Morbi fringilla, wisi in dignissim interdum, justo lectus sagittis dui, et vehicula libero dui cursus dui. Mauris tempor ligula sed lacus. Duis cursus enim ut augue. Cras ac magna. Cras nulla. Nulla egestas. Curabitur a leo. Quisque egestas wisi eget nunc. Nam feugiat lacus vel est. Curabitur consectetuer.

Suspendisse vel felis. Ut lorem lorem, interdum eu, tincidunt sit amet, laoreet vitae, arcu. Aenean faucibus pede eu ante. Praesent enim elit, rutrum at, molestie non, nonummy vel, nisl. Ut lectus eros, malesuada sit amet, fermentum eu, sodales cursus, magna. Donec eu purus. Quisque vehicula, urna sed ultricies auctor, pede lorem egestas dui, et convallis elit erat sed nulla. Donec luctus. Curabitur et nunc. Aliquam dolor odio, commodo pretium, ultricies non, pharetra in, velit. Integer arcu est, nonummy

in, fermentum faucibus, egestas vel, odio.

Sed commodo posuere pede. Mauris ut est. Ut quis purus. Sed ac odio. Sed vehicula hendrerit sem. Duis non odio. Morbi ut dui. Sed accumsan risus eget odio. In hac habitasse platea dictumst. Pellentesque non elit. Fusce sed justo eu urna porta tincidunt. Mauris felis odio, sollicitudin sed, volutpat a, ornare ac, erat. Morbi quis dolor. Donec pellentesque, erat ac sagittis semper, nunc dui lobortis purus, quis congue purus metus ultricies tellus. Proin et quam. Class aptent taciti sociosqu ad litora torquent per conubia nostra, per inceptos hymenaeos. Praesent sapien turpis, fermentum vel, eleifend faucibus, vehicula eu, lacus.

Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Donec odio elit, dictum in, hendrerit sit amet, egestas sed, leo. Praesent feugiat sapien aliquet odio. Integer vitae justo. Aliquam vestibulum fringilla lorem. Sed neque lectus, consectetuer at, consectetuer sed, eleifend ac, lectus. Nulla facilisi. Pellentesque eget lectus. Proin eu metus. Sed porttitor. In hac habitasse platea dictumst. Suspendisse eu lectus. Ut mi mi, lacinia sit amet, placerat et, mollis vitae, dui. Sed ante tellus, tristique ut, iaculis eu, malesuada ac, dui. Mauris nibh leo, facilisis non, adipiscing quis, ultrices a, dui.

Morbi luctus, wisi viverra faucibus pretium, nibh est placerat odio, nec commodo wisi enim eget quam. Quisque libero justo, consectetuer a, feugiat vitae, porttitor eu, libero. Suspendisse sed mauris vitae elit sollicitudin malesuada. Maecenas ultricies eros sit amet ante. Ut venenatis velit. Maecenas sed mi eget dui varius euismod. Phasellus aliquet volutpat odio. Vestibulum ante ipsum primis in faucibus orci luctus et ultrices posuere cubilia Curae; Pellentesque sit amet pede ac sem eleifend consectetuer. Nullam elementum, urna vel imperdiet sodales, elit ipsum pharetra ligula, ac pretium ante justo a nulla. Curabitur tristique arcu eu metus. Vestibulum lectus. Proin mauris. Proin eu nunc eu urna hendrerit faucibus. Aliquam auctor, pede consequat laoreet varius, eros tellus scelerisque quam, pellentesque hendrerit ipsum dolor sed augue. Nulla nec lacus.

Suspendisse vitae elit. Aliquam arcu neque, ornare in, ullamcorper quis, commodo eu, libero. Fusce sagittis erat at erat tristique mollis. Maecenas sapien libero, molestie et, lobortis in, sodales eget, dui. Morbi ultrices rutrum lorem. Nam elementum ullamcorper leo. Morbi dui. Aliquam sagittis. Nunc placerat. Pellentesque tristique sodales est. Maecenas imperdiet lacinia velit. Cras non urna. Morbi eros pede, suscipit ac, varius vel, egestas non, eros. Praesent malesuada, diam id pretium elementum, eros sem dictum tortor, vel consectetuer odio sem sed wisi.

Sed feugiat. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Ut pellentesque augue sed urna. Vestibulum diam eros, fringilla et, consectetuer eu, nonummy id, sapien. Nullam at lectus. In sagittis ultrices mauris. Curabitur malesuada erat sit amet massa. Fusce blandit. Aliquam erat volutpat. Aliquam euismod. Aenean vel lectus. Nunc imperdiet justo nec dolor.

Etiam euismod. Fusce facilisis lacinia dui. Suspendisse potenti. In mi erat, cursus id, nonummy sed, ullamcorper eget, sapien. Praesent pretium, magna in eleifend egestas, pede pede pretium lorem, quis consectetuer tortor sapien facilisis magna. Mauris quis magna varius nulla scelerisque imperdiet. Aliquam non quam. Aliquam porttitor quam a lacus. Praesent vel arcu ut tortor cursus volutpat. In vitae pede quis diam bibendum placerat. Fusce elementum convallis neque. Sed dolor orci, scelerisque ac, dapibus nec, ultricies ut, mi. Duis nec dui quis leo sagittis commodo.

Aliquam lectus. Vivamus leo. Quisque ornare tellus ullamcorper nulla. Mauris porttitor pharetra tortor. Sed fringilla justo sed mauris. Mauris tellus. Sed non leo. Nullam elementum, magna in cursus sodales, augue est scelerisque sapien, venenatis congue nulla arcu et pede. Ut suscipit enim vel sapien. Donec congue. Maecenas urna mi, suscipit in, placerat ut, vestibulum ut, massa. Fusce ultrices nulla et nisl.

Etiam ac leo a risus tristique nonummy. Donec dignissim tincidunt nulla. Vestibulum rhoncus molestie odio. Sed lobortis, justo et pretium lobortis, mauris turpis condimentum augue, nec ultricies nibh arcu pretium enim. Nunc purus neque, placerat id, imperdiet sed, pellentesque nec, nisl. Vestibulum imperdiet neque non sem accumsan laoreet. In hac habitasse platea dictumst. Etiam condimentum facilisis libero. Suspendisse in elit quis nisl aliquam dapibus. Pellentesque auctor sapien. Sed egestas sapien nec lectus. Pellentesque vel dui vel neque bibendum viverra. Aliquam porttitor nisl nec pede. Proin mattis libero vel turpis. Donec rutrum mauris et libero. Proin euismod porta felis. Nam lobortis, metus quis elementum commodo, nunc lectus elementum mauris, eget vulputate ligula tellus eu neque. Vivamus eu dolor.

Nulla in ipsum. Praesent eros nulla, congue vitae, euismod ut, commodo a, wisi. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Aenean nonummy magna non leo. Sed felis erat, ullamcorper in, dictum non, ultricies ut, lectus. Proin vel arcu a odio lobortis euismod. Vestibulum ante ipsum primis in faucibus orci luctus et ultrices posuere cubilia Curae; Proin ut est. Aliquam odio. Pellentesque massa turpis, cursus eu, euismod nec, tempor congue, nulla. Duis viverra gravida mauris. Cras tincidunt. Curabitur eros ligula, varius ut, pulvinar in, cursus faucibus, augue.

Nulla mattis luctus nulla. Duis commodo velit at leo. Aliquam vulputate magna et leo. Nam vestibulum ullamcorper leo. Vestibulum condimentum rutrum mauris. Donec id mauris. Morbi molestie justo et pede. Vivamus eget turpis sed nisl cursus tempor. Curabitur mollis sapien condimentum nunc. In wisi nisl, malesuada at, dignissim sit amet, lobortis in, odio. Aenean consequat arcu a ante. Pellentesque porta elit sit amet orci. Etiam at turpis nec elit ultricies imperdiet. Nulla facilisi. In hac habitasse platea dictumst. Suspendisse viverra aliquam risus. Nullam pede justo, molestie nonummy, scelerisque eu, facilisis vel, arcu.

Curabitur tellus magna, porttitor a, commodo a, commodo in, tortor. Donec interdum. Praesent scelerisque. Maecenas posuere sodales odio. Vivamus metus lacus, varius quis, imperdiet quis, rhoncus a, turpis. Etiam ligula arcu, elementum a, venenatis quis, sollicitudin sed, metus. Donec nunc pede, tincidunt in, venenatis vitae, faucibus vel, nibh. Pellentesque wisi. Nullam malesuada. Morbi ut tellus ut pede tincidunt porta. Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Etiam congue neque id dolor.

Donec et nisl at wisi luctus bibendum. Nam interdum tellus ac libero. Sed sem justo, laoreet vitae, fringilla at, adipiscing ut, nibh. Maecenas non sem quis tortor eleifend fermentum. Etiam id tortor ac mauris porta vulputate. Integer porta neque vitae massa. Maecenas tempus libero a libero posuere dictum. Vestibulum ante ipsum primis in faucibus orci luctus et ultrices posuere cubilia Curae; Aenean quis mauris sed elit commodo placerat. Class aptent taciti sociosqu ad litora torquent per conubia nostra, per inceptos hymenaeos. Vivamus rhoncus tincidunt libero. Etiam elementum pretium justo. Vivamus est. Morbi a tellus eget pede tristique commodo. Nulla nisl. Vestibulum sed nisl eu sapien cursus rutrum.

Nulla non mauris vitae wisi posuere convallis. Sed eu nulla nec eros scelerisque pharetra. Nullam varius. Etiam dignissim elementum metus. Vestibulum faucibus, metus sit amet mattis rhoncus, sapien dui laoreet odio, nec ultricies nibh augue a enim. Fusce in ligula. Quisque at magna et nulla commodo consequat. Proin accumsan imperdiet sem. Nunc porta. Donec feugiat mi at justo. Phasellus facilisis ipsum quis ante. In ac elit eget ipsum pharetra faucibus. Maecenas viverra nulla in massa.

Nulla ac nisl. Nullam urna nulla, ullamcorper in, interdum sit amet, gravida ut, risus. Aenean ac enim. In luctus. Phasellus eu quam vitae turpis viverra pellentesque. Duis feugiat felis ut enim. Phasellus pharetra, sem id porttitor sodales, magna nunc aliquet nibh, nec blandit nisl mauris at pede. Suspendisse risus risus, lobortis eget, semper at, imperdiet sit amet, quam. Quisque scelerisque dapibus nibh. Nam enim.

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Nunc ut metus. Ut metus justo, auctor at, ultrices eu, sagittis ut, purus. Aliquam aliquam.

Capítulo 2

Estado de la Cuestión

En este capítulo se realizará una aproximación a las historias de vida como terapia de reminiscencia de la enfermedad de Alzheimer. A continuación, se presentarán las técnicas de generación de lenguaje natural con mayor relevancia como método alternativo de composición de dichas historias de vida. Para ello, navegaremos a lo largo de la historia de la generación, concretamente del modelado de lenguaje como motor del sistema. Se abordarán las técnicas más antiguas de la generación, como el análisis de la arquitectura tradicional de un sistema de generación de lenguaje genérico, hasta los modelos de lenguaje más novedosos.

2.1. Alzheimer e historias de vida

La pirámide poblacional modifica su estructura continuamente debido al progresivo envejecimiento generalizado de la población. Según proyecciones de la Alzheimer's Association International (2021), en el año 2050 las personas mayores de 65 años constituirán el 16 por ciento de la población mundial frente al 8 por ciento del año 2010. El aumento de la esperanza de vida en todo el mundo, principalmente en las sociedades más avanzadas, y la disminución de la natalidad, se encuentran entre las causas de la modificación de la distribución demográfica hacia edades más avanzadas. Este fenómeno es conocido como *inversión de la pirámide poblacional* (Vea, 2017).

La realidad detrás de estas estadísticas: el incremento del número de personas de edad avanzada, y asociándose al envejecimiento la acumulación a lo largo del tiempo de una gran variedad de daños moleculares y celulares que lleva a un descenso gra-

dual de las capacidades mentales y físicas, deriva en un mayor riesgo de determinadas enfermedades.

La pérdida de la audición, las cataratas, la artritis y la artrosis son solo algunas de las enfermedades con mayor incidencia. Sin embargo, una de las dolencias más comunes y serias dentro de este rango de población es la enfermedad de Alzheimer, cuya prevalencia a nivel global se espera que supere todo dato conocido hasta ahora, dado que se estima que en el año 2050 se incremente el número de casos a 152,8 millones, sobrepasando considerablemente los 57,4 millones del año 2019 (Alzheimer's Disease International, 2019).

2.1.1. Descripción general

La enfermedad de Alzheimer es un trastorno neurológico caracterizado por cambios degenerativos en diferentes sistemas neurotransmisores que abocan finalmente a la muerte de las células nerviosas del cerebro encargadas del almacenamiento y procesamiento de la información. Las regiones del cerebro involucradas con la memoria y los procesos de aprendizaje, asociadas a los lóbulos temporal y frontal, reducen su tamaño como consecuencia de la degeneración de las sinapsis y la muerte de las neuronas (Romano et al., 2007; Mattson, 2004). En las etapas finales de esta patología, este proceso, también denominado *atrofia cerebral* se extiende y provoca una pérdida significativa del volumen cerebral (figura 2.1 a).

En numerosas ocasiones son utilizadas imágenes similares a las mostradas en la figura 2.1 como indicativos de la enfermedad del Alzheimer. La figura 2.1 b representa unas *tomografías por emisión de positrones* o *PET scans* en inglés. En ellas se reflejan los patrones de distribución espacial de la glucosa en el cerebro. En el cerebro de la persona con Alzheimer, el flujo glucolítico cerebral se minimiza provocando los síntomas de la enfermedad. Esta prueba se utiliza en el diagnóstico de la gravedad de la patología.

El proceso de detección de la enfermedad de Alzheimer es una tarea ardua de realizar dado que, por lo general, los síntomas iniciales de la enfermedad suelen atribuirse a un olvido puntual o la vejez. Nada más lejos de la realidad. Según avanza la enfermedad, sus síntomas lo hacen con ella, agravándose y aumentando cada vez más hasta que el deterioro cognitivo ocasionado llega a afectar significativamente a las actividades de la vida diaria y finalmente a las necesidades fisiológicas básicas.

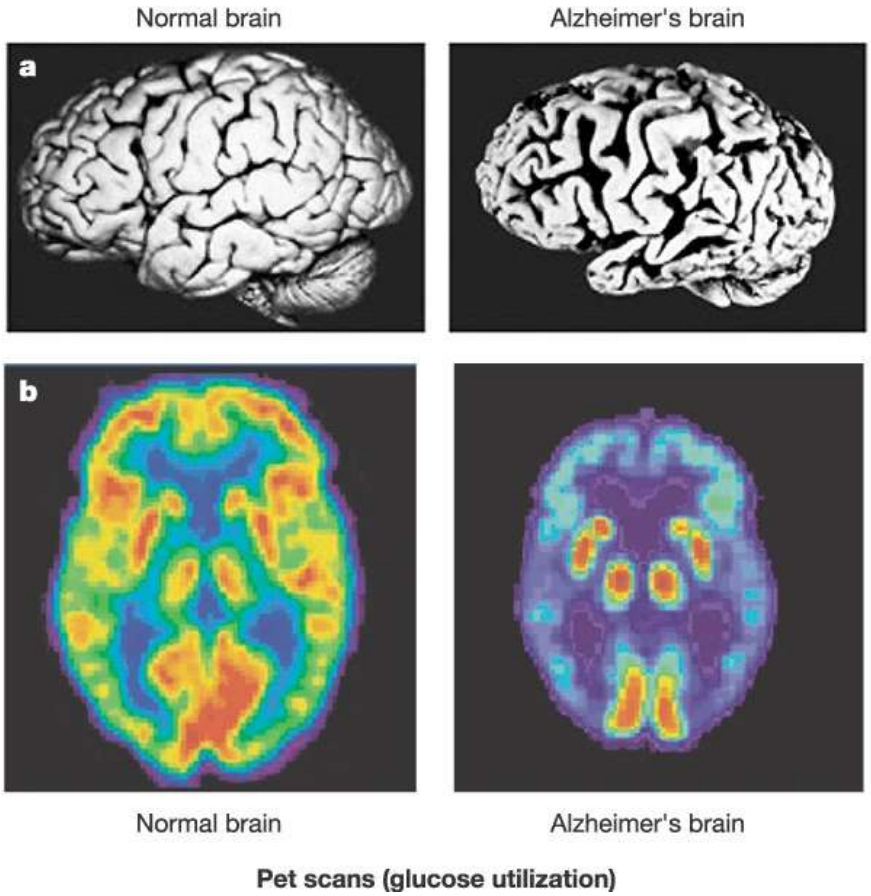


Figura 2.1: Reducción del cerebro asociada al Alzheimer (Mattson, 2004)

La evolución del Alzheimer se puede dividir en tres fases o etapas. En una primera instancia, se comienza a observar un deterioro cognitivo leve como puede ser la pérdida paulatina de la memoria episódica, seguido de pérdidas de la memoria reciente asociadas a un deterioro mayor así como otras funciones mentales y de la personalidad. Para terminar, se produce una pérdida progresiva de la memoria referida a los acontecimientos más antiguos, acompañando además un importante deterioro físico.

2.1.2. Síntomatología y pérdida de la memoria

La amnesia o pérdida de la memoria es uno de los síntomas más representativos del Alzheimer. Sin embargo, se trata tan solo de la punta del iceberg debido a todos los desordenes que también se producen y que no son considerados o tenidos en cuenta por el personal no profesional: alteraciones del estado de ánimo y la conducta, dificultad de toma de decisiones, desorientación, problemas del lenguaje, dificultad para

comer, movilidad reducida y un largo etcétera son algunos de los síntomas que acompañan a esta enfermedad durante todo su camino. Todos estos síntomas dependen de la fase evolutiva de la enfermedad.

Podemos distinguir en cuanto a sintomatología dos fases marcadas por las alteraciones neurológicas: en una primera fase, conocida como fase predemencial, los signos de desordenes neurológicos todavía no se encuentran presentes; y la fase demencial, en la que se pueden observar grandes alteraciones motoras, cognitivas, sensoriales y emocionales.

En la etapa predemencial, durante la cual en numerosas ocasiones el paciente no se encuentra diagnosticado de la enfermedad, comienzan a producirse lesiones microscópicas en el cerebro. Sin embargo, no es hasta entre 10 y 20 años después que pueden aparecer las primeras alteraciones cognitivas. El conjunto de síntomas presentes en esta fase comprende principalmente alteraciones en la conducta como trastorno de la personalidad, apatía o cambios en el estado de ánimo; y deterioro gradual de la memoria, comenzando el paciente a olvidar pequeñas cosas hasta llegar a no ser capaz de recordar familia o amigos.

A medida que progresa el daño cerebral aparece progresivamente un deterioro más pronunciado del paciente, comenzando entonce la fase demencial de la enfermedad. En esta etapa comienzan a aparecer alteraciones neurológicas como pérdida del movimiento, temblores, alucinaciones, trastornos en el lenguaje oral y escrito o alteraciones de la personalidad (Alberca Serrano y López Pousa, 2010).

2.1.3. Tratamientos: historias de vida

En la actualidad el Alzheimer es una enfermedad irreversible. Sin embargo, existen diversos tratamientos disponibles para ralentizar el avance de la enfermedad, así como mejorar la calidad de vida de los pacientes. Estos tratamientos se pueden dividir en dos ramas diferenciadas: tratamientos farmacológicos o farmacoterapia, que hacen uso de medicamentos; y tratamientos no farmacológicos o psicosociales, que no hacen uso de sustancias químicas. Ambos tipos de tratamientos resultan eficaces para tratar la enfermedad de Alzheimer. Sin embargo, de la combinación de ambos resulta el procedimiento más recomendado debido a su mayor efectividad. Esto es posible gracias a que ambos tipos de tratamientos no son mutuamente excluyentes (Romano et al., 2007).

Existen una gran variedad de terapias no farmacológicas. Algunas de las más utilizadas son el entrenamiento y estimulación cognitiva, ejercicio físico o musicoterapia. Además, en cada una de estas terapias podemos encontrar una enorme cantidad de técnicas, siendo la reminiscencia la más utilizada como terapia de estimulación cognitiva.

Según O'Rourke et al. (2013), la reminiscencia es el acto o proceso de recordar sucesos, eventos o información del pasado. Esto puede implicar el recuerdo de episodios particulares o genéricos que pueden o no haber sido olvidados previamente, y que son acompañados por la sensación de que estos episodios son relatos verídicos de las experiencias originales. Esta técnica es empleada en la estimulación del la memoria episódica autobiográfica mediante el encadenamiento de recuerdos, que se agrupan en categorías y se archivan en el tiempo mediante la elaboración de la *historia de vida*.

La historia de vida es una técnica narrativa que se basa en organizar y estructurar recuerdos de una persona para componer una autobiografía. Según Linde et al. (1993), una historia de vida debe cumplir dos criterios: primero, debe incluir algunos puntos de evaluación que comuniquen los valores morales de la persona; y segundo, los eventos incluidos en la historia de vida deben tener un significado especial y ser de importancia para ella. Estos eventos deben ser aspectos significativos de la vida pasada de la persona, su presente y su futuro.

Para componer la historia de vida de una persona con Alzheimer se recopilan historias a través de familiares u otras personas cercanas. Posteriormente, se documentan en forma de un libro o cuaderno, incluyendo experiencias y logros junto con fotografías y escritos sobre hechos importantes para la vida de la persona, a través de los cuales se muestra quién es esa persona.

Cada persona tiene su propia historia de vida única. Nuestras experiencias nos modelan y construyen la persona que somos. Las historias de vida ayudan a las personas con Alzheimer a conectar con su identidad recordando épocas felices. El miedo y la frustración provocados por el olvido de las tareas de la vida cotidiana, nombres y rostros, se mitigan recordando quiénes eran a través de estas historias. Les ayuda a ser conscientes de los momentos especiales que han marcado su vida, las personas que han conocido en su infancia o trabajo. También pueden ser utilizados por los cuidadores para comprender más sobre ellos, quiénes son, y ayudarles en la reminiscencia de recuerdos (Karlsson et al., 2014).

Existen diferentes formatos en los que se pueden registrar estas experiencias de la

persona. Ninguno de ellos es mejor o peor que otro, sino que lo ideal es utilizar aquel que mejor se adapte a la persona y a los hechos que se quieran transmitir.

Por una parte encontramos historias de vida más visuales, compuestas enteramente de imágenes (*collages*) o videos, dirigidas especialmente a las personas con Alzheimer que se encuentran en una etapa tardía de la enfermedad. Otro formato se centra especialmente en textos. Los *libros de vida*, destinados a los cuidadores y visitantes tanto como a la propia persona, combina las *historias de vida*, en forma de texto claro y fácil de leer, con algunas imágenes. También nos encontramos los documentos de perfil personal que se centran en pequeñas versiones cortas de las historias de vida excluyendo las imágenes. Estos documentos son utilizados a menudo en hospitales y están diseñados para ayudar al personal a comprender las necesidades de la persona.

El contenido de una historia de vida es variable, aunque existen algunos temas básicos en los que se debe centrar: el perfil de la persona, incluyendo datos e información básica como es el nombre, edad, lugar de nacimiento o de residencia son esenciales para aproximarse de manera inicial a la persona. Otros temas como las relaciones significativas familiares y de amistad, infancia, lugares y eventos significativos y gustos o preferencias y aficiones son incluidos dentro de esta lista de posibles temas a tratar el la historia de vida (Thompson, 2011).

2.2. Generación de lenguaje natural

La Generación de Lenguaje Natural (GLN) se define como el "subcampo de la inteligencia artificial y la lingüística computacional que se ocupa de la construcción de sistemas informáticos que pueden producir textos comprensibles en inglés u otros lenguajes humanos a partir de alguna representación no lingüística subyacente de la información" (Reiter y Dale, 1997). Si bien esta definición estuvo generalmente aceptada como la más conveniente al hablar de generación de lenguaje natural durante muchos años, Gatt y Krahmer (2018) puntualizan que es una afirmación que solo engloba una parte de la generación de textos, ya que se refiere únicamente a aquellos sistemas cuya entrada es una "representación no lingüística [...] de la información" o datos, como veremos más adelante en el apartado 2.2.2.

Desde hace muchos años, la GLN es empleada en numerosos proyectos de distinta naturaleza como la traducción de textos (Cho et al., 2014), realización de resúmenes y fusión de documentos (Clarke y Lapata, 2010), corrección automática de ortografía

y gramática (Islam et al., 2018), redacción de noticias (Leppänen et al., 2017), informes meteorológicos (Sripada et al., 2014) y financieros (Ren et al., 2021), generación de resúmenes sobre la información de recién nacidos en un contexto clínico (Gatt et al., 2009a)... Todos estos sistemas tienen en común la generación de un texto (normalmente de una alta calidad) a partir de muy diferentes fuentes de información.

En los ejemplos de proyectos listados con anterioridad que emplean la generación de lenguaje natural para redactar distintos textos, los datos utilizados como fuente de información son muy dispares, no solo en su contenido sino también en el tipo de dato. Así, si para la traducción de textos se utiliza texto ya existente como entrada, en otros sistemas como en la generación de informes meteorológicos se emplean datos no lingüísticos. De esta manera, se consideran dos posibles enfoques en los sistemas GLN dependiendo del tipo de entrada: texto a texto (*text-to-text*) y dato a texto (*data-to-text*).

2.2.1. Generación *text-to-text* (T2T)

Los sistemas de generación texto a texto, conocidos como *text-to-text* en inglés o T2T por sus siglas, toman textos escritos en lenguaje natural como entrada y producen un texto nuevo y coherente como salida. La entrada de estos sistemas puede abarcar desde pequeñas oraciones a extensos escritos. Existen muchas aplicaciones en los sistemas GLN que utilizan T2T. Además de los mencionados anteriormente, pertenecen a este tipo la fusión de documentos y generación de resúmenes (Clarke y Lapata, 2010), simplificación de textos complejos (Sulem et al., 2018), o autocorrectores gramaticales (Ge et al., 2019), entre otros.

Sin embargo, el ejemplo más claro de este tipo de generación de lenguaje corresponde a un traductor automático. Este tipo de sistema ampliamente utilizado en la vida cotidiana toma una entrada textual correspondiente a un escrito en un idioma y genera un texto de salida en otro idioma. La traducción automática es un proceso muy complejo, puesto que no solamente tiene en cuenta el significado del corpus, sino que también hace falta interpretar y analizar de manera correcta todos los elementos del texto, así como comprender la influencia de unas palabras en otras con la finalidad de generar un texto fluido y coherente.

2.2.2. Generación *data-to-text* (D2T)

Estos tipos de sistemas permiten la generación de texto como salida a partir de entradas no textuales. Además, el formato de los datos que pueden tomar como entrada son muy diversos. Aunque es muy común encontrar sistemas que parten de datos numéricos como hojas de cálculo, hay que considerar otros orígenes de datos de tipo estructurado tales como bases de datos, simulaciones de sistemas físicos o grafos de conocimientos. De manera general, podemos referirnos a la representación de la información de esta clase de sistema como datos estructurados o procesables.

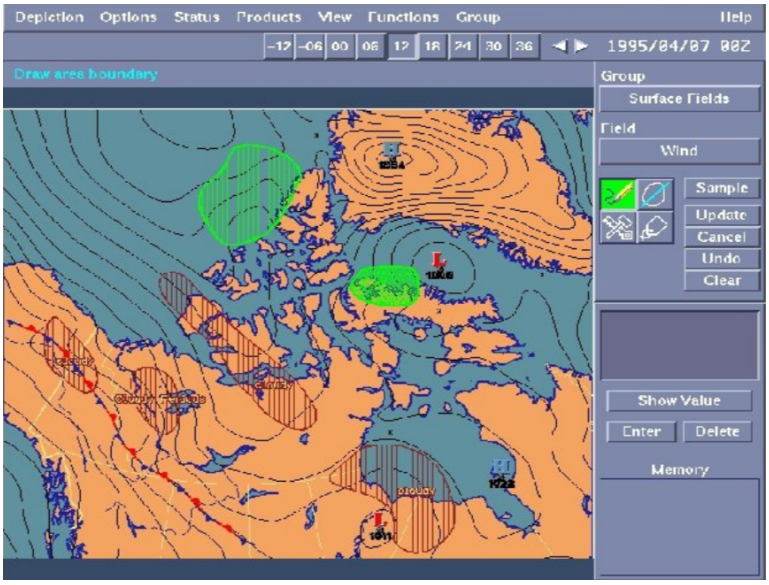
Algunos autores prefieren emplear el término *concepto* en lugar de *data*, motivo por el que algunos se refieren a este enfoque como generación concept-to-text (C2T) (Vicente et al., 2015).

Uno de los ejemplos más visuales que nos permite comprender este tipo de sistema sería el *Forecast Generator*, sistema que forma parte del *Forecaster's Production Assistant*, entorno desarrollado por *CoGenTex* en 1992 para *Environment Canada* con el fin ayudar a los meteorólogos a aumentar su productividad al redactar por ellos un informe meteorológico textual en inglés y en francés (Goldberg et al., 1994). En la figura 2.2a se muestra el entorno sobre el que los meteorólogos modifican valores como la presión atmosférica, situación de frentes y otros datos (datos no textuales). Una vez se pulsa sobre *Generar*, el sistema muestra el texto correspondiente al informe (figura 2.2b).

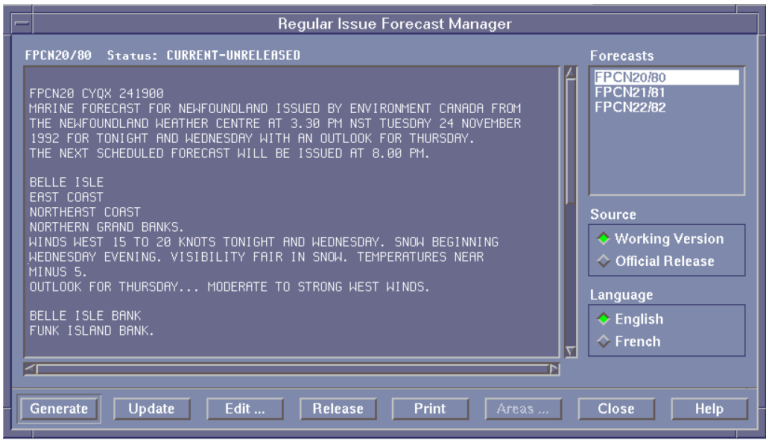
En la figura 2.3, explicada con más detalle en Sai et al. (2020), se muestran los datos de entrada y de salida de un sistema GLN D2T acercándonos a la generación de lenguaje desde una perspectiva distinta al ejemplo explicado anteriormente. Los datos de entrada de este tipo de sistema toman la forma de grafo o cualquier otro tipo de datos semiestructurados como tablas (conjunto de tuplas del tipo [entidad, atributo, valor]). En la fila inferior, se muestran diferentes posibles soluciones como salida del sistema. Además, el autor introduce la necesidad de métodos de evaluación de la calidad del texto redactado ya que de las diferentes salidas, solo la tercera opción cubre toda la información de entrada y resulta ser fluida.

2.3. Arquitectura tradicional y herramientas de un sistema GLN

El objetivo final de un sistema de generación de lenguaje natural es mapear unos datos de entrada a un texto de salida (Reiter y Dale, 1997). Sin embargo, este proceso,



(a) Entrada del sistema FoG



(b) Salida del sistema FoG

Figura 2.2: Sistema *data-to-text* FoG

aunque pueda parecer sencillo de entender, resulta complicado de llevar a cabo. Al principio del desarrollo de sistemas GLN, no había un consenso entre autores a la hora de establecer un proceso para construir este sistema. Finalmente, Reiter y Dale (1997) propusieron una arquitectura asociada a una lista de tareas recomendables que se deben realizar a la hora de llevar a cabo dicha construcción. Esta arquitectura surgió de la observación de los diferentes sistemas que se habían llevado a cabo hasta la fecha. Actualmente, es la solución más extendida y reconocida.

La arquitectura presentada por Reiter y Dale (1997), como se puede observar en la figura2.4, se divide en tres módulos: macroplanificación, microplanificación y realización. Además, cada módulo contiene una lista de tareas. Esta asignación tareas-

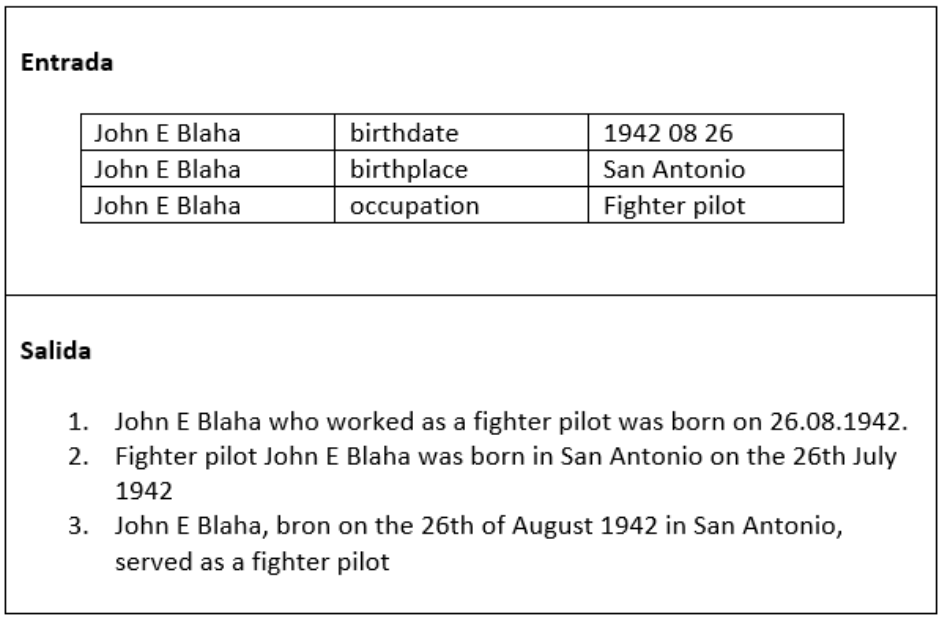


Figura 2.3: Ejemplo de D2T utilizado por Sai et al. (2020)

módulo no es inamovible. Una tarea asociada a un módulo se puede realizar en otro si así se considera, incluso implementar su desarrollo a lo largo de varios módulos. Los módulos que se corresponden con las tareas iniciales suelen estar relacionados con adaptar datos o estructura al sistema de generación, mientras que los módulos finales corresponden a la transformación de los resultados intermedios en el texto final.

2.3.1. Macroplanificación

Este es el primer módulo de un sistema de generación de lenguaje. Debe determinar qué decir, seleccionando para ello la información de entrada necesaria y organizarla en una estructura coherente, resultando de este proceso el plan del documento. Las tareas que intervienen se describen en los apartados siguientes.

2.3.1.1. Selección del contenido

La selección o determinación del contenido puede definirse como el proceso de decidir qué información debe ser incluida en el texto generado y cuál no. Por lo general, la información de la que partimos contendrá más información de la que nos interesa, así debemos decidir qué información resulta innecesaria y por tanto tenemos que eli-

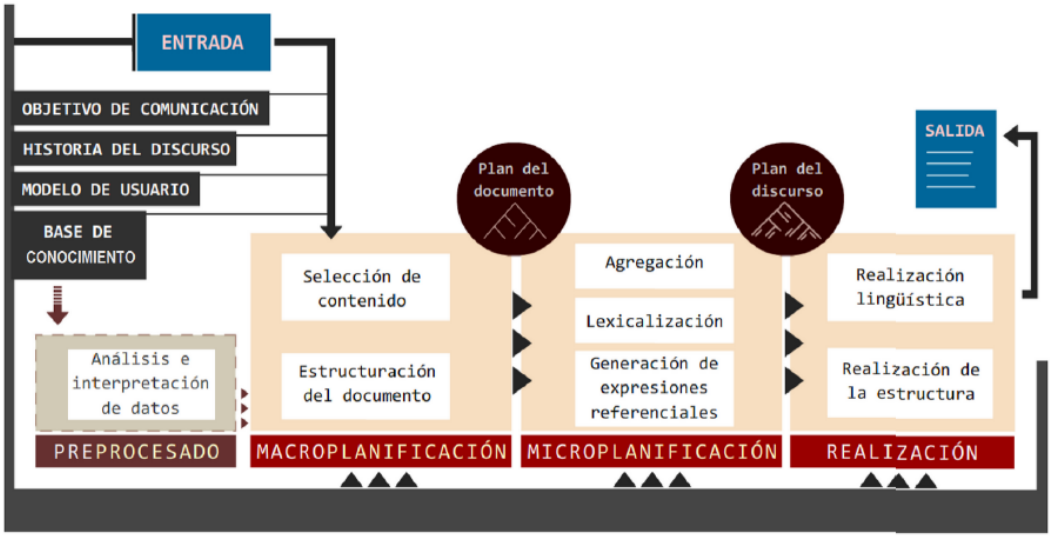


Figura 2.4: Arquitectura de referencia para sistema GLN (Vicente et al., 2015)

minar para la generación del texto final. También hay que tener en cuenta el público al que está dirigido el texto generado, ya que dependiendo de este podremos incluir cierta información de los datos entrantes o no.

Este proceso de selección de la información lleva a cabo el filtrado y resumen de esta en un conjunto de *mensajes*. Cada uno de estos mensajes corresponde al significado de una palabra u oración y se le asigna una entidad, concepto o relación dominante.

2.3.1.2. Estructuración del documento

Definiendo el concepto *texto* como "unidad de comunicación completa, formada habitualmente por una sucesión ordenada de enunciados que transmiten un mensaje con las siguientes propiedades: adecuación, coherencia y cohesión", podemos advertir que un texto no es un conjunto aleatorio de oraciones, sino que es necesaria la existencia de un orden en la presentación del texto final.

Dependiendo de la información que se comunique, este orden puede verse modificado o alterado. Es por ello que no hay una estructura fija, sino que hay que adecuarla al tipo de documento.

Una vez realizada la estructuración del texto, se obtiene un plan de discurso que corresponde a una representación estructurada y ordenada de los mensajes obtenidos en la tarea anterior.

2.3.2. Microplanificación

La microplanificación es el segundo módulo de la arquitectura. Parte del plan del documento resultante del módulo anterior para generar las oraciones evitando información redundante e innecesaria en el discurso. El resultado de este módulo es el plan de discurso. El proceso de generación de oraciones lo realiza mediante tres tareas.

2.3.2.1. Agregación de oraciones

La generación de una oración por cada uno de los mensajes puede resultar en la generación de un texto redundante y excesivamente estructurado. Una tarea en el proceso de construcción de un sistema GLN es la agregación de oraciones que pretender paliar este problema mediante la unión o agregación de contenidos de distintos mensajes en una sola oración. De esta manera los mensajes se combinan para obtener oraciones más largas y complejas, resultando en conjunto un texto menos estructurado y más fluido.

2.3.2.2. Lexicalización

En esta fase del proceso se empieza a generar el texto en lenguaje natural como tal. Para ello se debe decidir que palabras o estructuras sintácticas expresan mejor los conceptos y relaciones de las etapas anteriores. La dificultad de la generación en esta etapa reside en la gran cantidad de alternativas que encontramos para expresar cada uno de estos conceptos o bloques de mensajes.

Además, hace falta tener en cuenta un todavía mayor número de posibilidades ya que se debe considerar numerosas variables que podrían afectar al resultado final de la generación. Algunas de las variables a tener en cuenta pueden ser las necesidades o conocimiento de los usuarios, si el objetivo de la generación es generar textos con variaciones sintácticas o semánticas a lo largo del mismo, si es preferible un texto repetitivo y simple o diverso mediante la utilización de palabras sinónimas y para terminar, una apropiada selección de adjetivos.

2.3.2.3. Generación de expresiones de referencia

La diferenciación de unas entidades de otras para poder generar expresiones que se refieran a ellas es tratada en esta tarea con el objetivo de evitar la ambigüedad. Para realizar esta tarea se debe conseguir encontrar características particulares que contribuyan a diferenciar a una entidad del resto de entidades. Esta etapa está bastante consensuada en el campo GLN.

La generación de expresiones de referencia (REG, por sus siglas en inglés) debe llevarse a cabo una vez que el plan del documento se haya generado y depende de este, esto implica que esta fase debe llevarse a cabo desde el primer momento después de que se hayan analizado los datos. Debemos adaptar el plan de documento del primer módulo a lo que necesita REG, es por ello que debemos tener conocimiento de ello desde el comienzo.

Un caso especialmente estudiado que aplica esta técnica es la descripción de imágenes, ya que debe tener en cuenta si un elemento se encuentra a la derecha de otro, detrás de otro, etc, para poder enriquecer el texto. Para ello es necesario reconocer y distinguir los elementos en escena unos de otros y así, obtener una descripción lo más fidedigna posible a la imagen real.

2.3.3. Realización

La realización constituye el último módulo de la arquitectura de un sistema GLN. El objetivo final corresponde en generar oraciones gramaticalmente correctas para comunicar mensajes. En este módulo deberán tenerse en cuenta reglas a cerca de la formación de verbos (elección del tiempo verbal adecuado y por tanto generación de las palabras correspondientes), reglas sobre concordancia de género y número entre palabras (Reiter y Dale (1997) no tienen en cuenta el género de las palabras ya que focaliza la generación del lenguaje al inglés), generación de pronombres...

La entrada sobre la que se trabaja es el plan de discurso que contiene información sobre las oraciones generadas y la estructura utilizada en el texto final. En esta fase se traduce esta entrada en la salida que el usuario final recibirá.

Algunos autores consideran una única tarea de realización que engloba el convertir las especificaciones en oraciones y el dar un formato final al texto. Otros prefieren separar estas etapas para diferenciarlas y que sea más sencillo su estudio.

2.3.3.1. Realización lingüística

Con el objetivo de transformar las especificaciones de oraciones en las oraciones finales, en esta fase se ordenan los diferentes elementos constitutivos de una oración y se les asigna un formato correcto. Para elegir la forma morfológica correcta de una palabra se debe conjugar verbos, establecer concordancias de palabras, añadir formas pronominales en los lugares adecuados de las oraciones y establecer los signos de puntuación adecuados.

2.3.3.2. Realización de la estructura

Esta etapa no está considerada por algunos autores como tal aunque aquí se muestra ya que puede ser relevante en ciertos contextos. En algunos documentos, es necesario añadir o modificar algunas líneas del texto para darle estructura al documento. Un ejemplo muy sencillo de entender es la generación de texto que utilice html o Latex como formato de salida. En ambos casos, la adición de etiquetas a lo largo del texto generado resulta crucial para un texto de cualquiera de estas naturalezas.

2.3.4. Herramientas en la arquitectura tradicional

Los módulos constituyentes de la anterior arquitectura presentada pueden ser complicados de construir. Debido a la existencia de profesionales que quieren desarrollar un sistema de generación de lenguaje completo, basándose en esta arquitectura genérica de sistema NLG, o investigadores que se encuentren estudiando sus propias implementaciones de alguno de estos módulos y deseen enfocarse plenamente en él, se crearon una serie de herramientas que pueden ayudarlos con el arduo trabajo de implementación de las tareas que componen la arquitectura. De esta manera, algunas de las herramientas presentadas a continuación pueden utilizarse para tomar inspiración del proceso de desarrollo, mientras que otras se construyen con el objetivo de relegar por completo este proceso en la propia utilidad. En este apartado se realizará un breve análisis de las herramientas consideradas más relevantes, asociadas a cada uno de los módulos descritos. Cabe decir, que debido a la antigüedad de esta arquitectura, la mayoría de las herramientas encontradas son aproximadamente de entre diez y veinte años atrás, sin embargo, algunas de ellas todavía hoy en día conservan gran fama y usuarios.

En primer lugar, para la macroplanificación de la arquitectura tradicional de un

sistema NLG, dedicado a la selección de la información de entrada necesaria para el sistema junto a su posterior organización, se encontraron diferentes herramientas de diversa índole. Algunas como SPUR se construyeron expresamente para la realización de estas únicas tareas de macroplanificación. Concretamente, esta herramienta se desarrolló como parte de un sistema mayor, conocido como MATCH (Johnston et al., 2002), construido con el objetivo de realizar comparaciones entre restaurantes en Nueva York y obtener recomendaciones personalizadas. De esta manera, SPUR recibía como entrada los atributos de las opciones a comparar entre restaurantes y la salida correspondía al plan del documento constituido por los atributos más importantes de cada una de las opciones. Otras proyectos como PESCaDO (Wanner et al., 2012) o NaturalOWL (Androutsopoulos et al., 2014) además de tareas de planificación también incluyen componentes para el proceso de microplanificación e incluso realización .

En segundo lugar, para el módulo de microplanificación se desarrollaron herramientas como SPaRKy, utilizada también en el sistema MATCH anteriormente mencionado. En este caso, la entrada a la herramienta corresponde al plan de documento de la herramienta SPUR y su salida corresponde al plan de discurso con las afirmaciones a incluir en la salida del sistema. Otros proyectos mucho mas grandes como BabyTALK (Gatt et al., 2009b) o NaturalOWL también incluyen entre sus funcionalidades herramientas para el desarrollo de la microplanificación.

Para acabar, son muchos los proyectos mencionado anteriormente que incluyen herramientas internas que realizan funciones de realización, sin embargo, destaca la existencia de la herramienta SimpleNLG, que no está ligada al empleo de un proyecto en concreto sino que puede utilizarse con cualquier tipo de datos.

SimpleNLG (Gatt y Reiter, 2009) es una API de Java que proporciona interfaces que ofrecen un control directo sobre la tarea de realización. Define un conjunto de tipos léxicos, correspondientes a las principales categorías gramaticales, así como formas de combinarlos y establecer valores de características. Está orientada a la generación de oraciones gramaticalmente correctas en sistemas *data-to-text*. Aunque originalmente solo estaba disponible para textos de lengua inglesa, actualmente se encuentra versionado para muchos idiomas, entre ellos el español. La versión española de esta herramienta es conocida como SimpleNLG-ES (Ramos-Soto et al., 2017) y realmente se trata de una adaptación bilingüe de la versión original en inglés.

Esta herramienta se basa en la flexibilidad a la hora de generar textos mediante la utilización de manera combinada de sistemas basados en esquemas y otros enfoques más avanzados; la robustez generando salidas (aunque en ocasiones incorrectas) pese

subject = "Manuel"
verb = "bake"
object = "a cake"

Parameters	*Mood*	*Ouput*
Tense = present		Manuel bakes a cake
Tense = present	Negated = true	Manuel doesn't bake a cake
Tense = past	Interrogative_type = YES_NO	Did Manuel bake a cake?
Tense = future	Interrogative_type = WHAT_OBJECT	What did Manuel bake?*

Complements	
very quickly	Manuel bakes a cake very quickly

**No haría falta establecer el objeto*

Figura 2.5: Entrada y salida de la herramienta SimpleNLG

a que las entradas sean erróneas o incompletas; y por último, la independencia entre las operaciones de decisión de morfología y sintácticas.

Para definir una oración con SimpleNLG, se parte de sus constituyentes sintácticos básicos: sujeto, verbo y objeto. Una vez definido cada uno de ellos, por ejemplo estableciendo el sujeto como "Manuel", el verbo como "bake" y el objeto como "a cake", puede generarse una oración simple, siendo la salida resultante "Manuel bakes a cake". Cabe destacar la existencia de métodos que permiten establecer el tiempo verbal: pasado, presente o futuro, así como modificar el tipo de oración: enunciativa (por defecto), negativa o interrogativa. En el caso de este último tipo de oración, también permite seleccionar el tipo de pregunta formalizando el complemento circunstancial (quién, dónde, cómo...) o si debe formularse para que la respuesta sea sí o no. También permite establecer complementos a la oración y modificadores verbales. En la figura 2.5 se puede apreciar a través del ejemplo presentado, el resultado de todas estas funcionalidades.

2.4. Algoritmos y modelos para la generación de lenguaje

Cualquier sistema informático dedicado al tratamiento del lenguaje natural necesita un modelo que le permita caracterizar y representar la lengua que trata. En esta sección se realiza una aproximación a los algoritmos y modelos utilizados a lo largo de la historia para la generación de lenguaje. Se comenzará estudiando a través del tiempo las distintas aproximaciones mayormente empleadas durante la época y se estudiará de una manera más profunda aquellos enfoques más relevantes en la actualidad como son los modelos estadísticas basados en cadenas de Markov o N-Grams, así como enfoques neuronales a través del estudio de diferentes tipos de redes. Para terminar, se presentarán algunas imprecisiones de generación que comenten las redes neuronales al ser aplicadas a la generación de lenguaje.

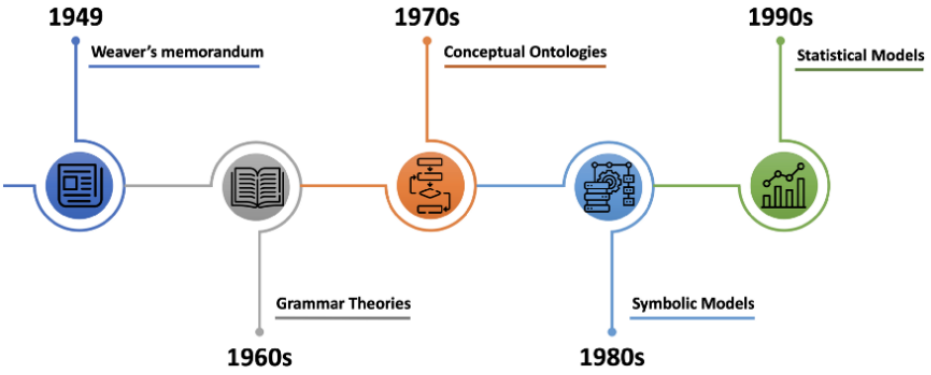
2.4.1. Historia

Antes de comenzar a describir los distintos tipos de modelos utilizados en tareas de generación de lenguaje natural, es interesante conocer como ha ido evolucionando este campo a lo largo de la historia y los modelos más relevantes en cada una de las etapas. En general, la historia del procesamiento de lenguaje natural se divide en dos grandes períodos marcados por la aparición del aprendizaje profundo o *Deep Learning* (Louis, 2021).

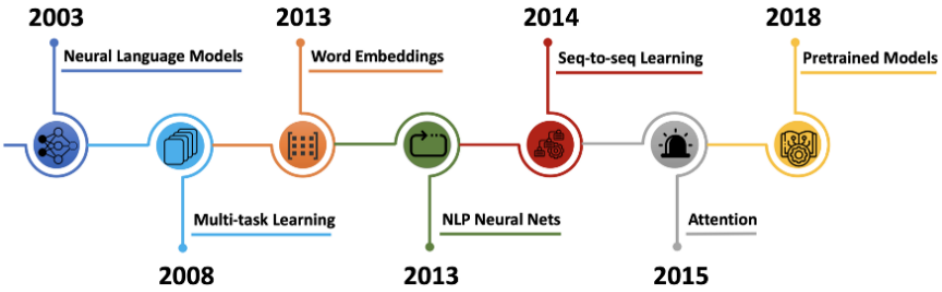
La era *pre Deep Learning* (figura 2.6a) comienza aproximadamente en el 1949, momento en el que Warren Weaver sugería en su memorando "Translation" [1] que la traducción automática computacional era posible. Esta fue la primera aproximación estadística al procesamiento y generación de lenguaje. Supuso una revolución e inspiró numerosos experimentos y proyectos que probaron que realmente esto era posible aunque a muy pequeña escala. Estos sistemas se basaban principalmente en la búsqueda en diccionarios de las palabras necesarias para la traducción y la posterior reordenación de las palabras para ajustarse a las reglas sintácticas del idioma destino de la traducción.

Después de una década de investigaciones para conseguir mejores resultados en este campo y de pérdida de financiación, ya que las soluciones encontradas hasta ahora conseguían resultados muy pobres, Surgieron nuevas *teorías de la gramática* mucho más manejables computacionalmente, y más tarde las *ontologías conceptuales* que es-

[1] Lectura disponible en https://web.stanford.edu/class/linguist289/weaver001.pdf



(a) Era pre *Deep Learning*



(b) Era *Deep Learning*

Figura 2.6: Etapas del Procesamiento de Lenguaje Natural

tructuraban la información del mundo real en datos comprensibles por la computadora.

En la década de 1980, surgieron los *Modelos Simbólicos* basados en reglas. Estos sistemas asignaban manualmente los significados de las palabras y de esta manera determinista se creaban oraciones. Debido a la complejidad de creación de estas reglas, ya que se debían crear a mano, estos modelos fueron ampliamente sustituidos por los *Modelos Estadísticos* que supusieron una revolución para el procesamiento de lenguaje en aquella época y que hoy en día todavía tienen una gran relevancia en la lingüística computacional.

Con el avance computacional de las *Redes Neuronales*, comienzan a usarse en la década del 2000 en el modelado del lenguaje para la generación de textos y dan lugar a la era *Deep Learning* (figura 2.6b). Bengio et al. (2000) propuso el primer modelo de lenguaje neuronal utilizando una Red Neuronal Prealimentada *(FeedForward Neural Network)* de una capa oculta. Otros autores sustituyeron progresivamente esta arquitectura de red por Redes Neuronales Recurrentes (*Recurrent Neural Networks*) y Redes

de Memoria a Corto Plazo (*Long Short-Term Memory*) aunque los componentes básicos de la arquitectura original se encuentran todavía en la mayoría de modelos de lenguaje neuronales.

Más tarde Collobert y Weston (2008) introdujeron el *Aprendizaje Multitarea* al procesamiento de lenguaje, utilizando una Red Neuronal Convolucional (*Convolutional Neural Network*) para conseguir que varias tareas de aprendizaje se resolvieran de manera simultánea, resultando en una mejora de la eficiencia.

Tras varios avances, como la introducción de modelos *Word Embeddings* o la adopción general de redes neuronales para el modelado de lenguaje, surge la arquitectura Secuencia a Secuencia (*Seq2Seq*). Estos sistemas estaban compuesto por dos componentes clave: el codificador o *encoder* y el decodificador o *decoder*. La revolución que supuso esta arquitectura fue significativa y todavía se siguen utilizando.

En 2014, Bahdanau et al. (2014) introduce los *mecanismos de atención* que alivia el problema de cuello de botella de los modelos predecesores, los *Seq2Seq*.

La última innovación en el mundo del Procesamiento de Lenguaje son los grandes Modelos de Lenguaje Preentrenados (*Pretrained Moldels*). Debido a todo el esfuerzo computacional de días, semanas e incluso meses que supone entrenar un modelo de lenguaje, se proponen una serie de modelos que ya tienen realizado este entrenamiento. La finalidad de estos sistemas es el ajuste o *fine-tune* de los mismos de acuerdo al objetivo que se quiera conseguir.

2.4.2. Modelos estadísticos

Estos tipos de modelos utilizan técnicas estadísticas y reglas lingüísticas para aprender las distribuciones de probabilidad de las palabras pertenecientes a un conjunto finito conocido como *vocabulario* y, de esta manera, generar secuencias de palabras específicas. Entre las técnicas más utilizadas y que mejores resultados han arrojado en este ámbito encontramos los modelos basados en cadenas de Markov y en N-gramas.

2.4.2.1. Modelo Markov Chain

Este modelo, introducido por el matemático ruso Andrey Markov en 1913, es un modelo estocástico discreto que describe una secuencia de posibles eventos. Aplicado a la generación de texto, podemos resumirlo en un sistema que se basa en una distri-

bución de probabilidades aleatorias para generar la siguiente palabra a un grupo de palabras o secuencia.

Para que un proceso se considere Markov debe satisfacer una condición conocida como la *propiedad de Markov*. Esta propiedad establece que la probabilidad del siguiente evento (en el caso de generación de lenguaje, la siguiente palabra) depende únicamente del evento actual. Si se considera, como es el caso, que las palabras futuras no dependen en absoluto de las palabras previas, la secuencia de palabras generada evoluciona de manera no determinista y entonces puede considerarse que la palabra X_n es una variable aleatoria. La colección de variables aleatorias es la definición de *proceso estocástico*, y sirve como modelo para representar la evolución aleatoria de una frase a lo largo del tiempo (Moyotl-Hernández y Macías-Pérez, 2016).

La fórmula 2.1 representa los fundamentos de la *propiedad de Markov*, donde X es una variable aleatoria (palabra) que toma un valor en el espacio de estado dado s (vocabulario); y n es una variable discreta que representa el paso de tiempo (Howell, 2022). Como se puede observar, suponiendo que nos encontramos en el paso de tiempo n, las probabilidades teniendo en cuenta los estados de los eventos anteriores y la probabilidad teniendo en cuenta únicamente el estado actual para modelar la siguiente palabra son equivalentes, por lo que la información anterior al estado actual carece de relevancia para el cálculo de la probabilidad del siguiente evento.

$$P(X_{n+1} = s_{n+1} | X_n = s_n, X_{n-1} = s_{n-1}, ..., X_0 = s_0) = P(X_{n+1} = s_{n+1} | X_n = s_n) \tag{2.1}$$

Viendo un ejemplo concreto, en la figura 2.7 se puede apreciar una cadena de Markov aprendida del corpus "A man, a plan, a canal: Panama! A dog, a panic in a pagoda!". Esta cadena está formada por un conjunto de estados, cada uno representa una palabra única del corpus, unidas por transiciones, que simbolizan la probabilidad de pasar de un estado a otro.

Para generar una secuencia de palabras con esta cadena, se comienza eligiendo aleatoriamente un estado inicial, supongamos "A" como elección. A continuación, se deben tomar una de las transiciones posibles en base a una elección aleatoria. Esta elección será "man," o "dog," con probabilidad 1/2; digamos que tomamos "dog,". Ahora debemos escoger "a" ya que no tenemos más posibilidades y seleccionar la siguiente palabra en base a las probabilidades. Tenemos cuatro opciones con similar probabilidad igual 1/4: "plan,","canal,", "pagoda!" y "panic". Suponiendo que esco-

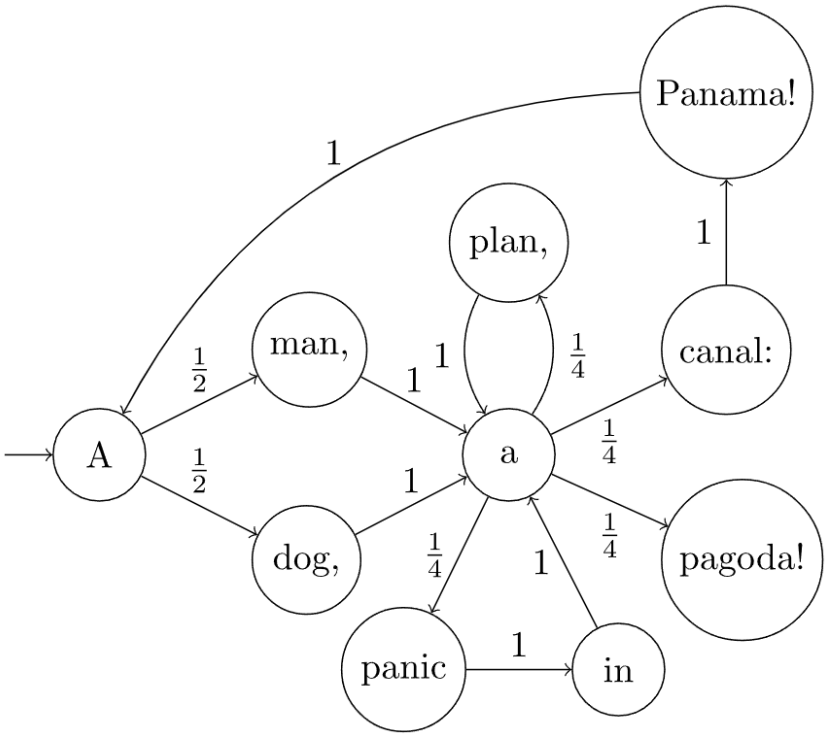


Figura 2.7: Ejemplo de cadena de Markov (Make School, 2017)

gemos "pagoda!", hasta ahora se ha formado la frase "A dog, a pagoda!". Este proceso se puede realizar indefinidamente hasta que se decida parar o se llegue a un estado sin transiciones de salida.

Debido a las características particulares de la propiedad de Markov, este modelo es un modelo sin memoria, ya que se desprecian todos los estados anteriores, por lo que no se retiene información relevante de un texto como las posiciones de las palabras en una oración o la relación entre palabras. Sin embargo, precisamente por carecer de memoria, es simple de comprender y rápido de ejecutar (Fumagalli, 2020).

2.4.2.2. Modelo N-gram

Otro modelo estadístico de gran transcendencia es el modelo *N-Gram*. Este modelo va un poco más allá del modelo *Markov Chain*, ya que se basa en realizar una predicción estadística de una secuencia de palabras teniendo en cuenta un conjunto de palabras anteriores. Esta secuencia de N palabras es representada mediante un N-grama. Así, este modelo trata de predecir el N-grama más probable dentro de cualquier secuencia de palabras dado el historial de las N-1 palabras anteriores. Para su mejor comprensión, se expone un ejemplo ejemplo concreto a partir del siguiente extracto de "La vida

es sueño" de Calderón de la Barca:

¿Qué es la vida? Un frenesí.
¿Qué es la vida? Una ilusión,
una sombra, una ficción,
y el mayor bien es pequeño;
que toda la vida es sueño,
y los sueños, sueños son.

Podemos construir N-gramas a partir del texto anterior teniendo en cuenta que un N-grama está formado por N palabras que aparecen consecutivas en el corpus. A continuación se muestran unigramas, bigramas y trigramas extraídos del texto.

Unigramas
{ qué, es, la vida, un frenesí, una, ilusión, sombra, ficción, y, el, mayor,... }
Bigramas
{ qué es, es la, la vida, vida un, un frenesí, frenesí qué, una ilusión,... }
Trigramas
{ qué es la, es la vida, la vida un, vida un frenesí, un frenesí qué,...}

Este proceso se realiza de manera iterativa hasta llegar al final del corpus teniendo en cuenta que no se pueden repetir dos N-gramas iguales dentro de un mismo conjunto. Una vez construidos los N-gramas se puede calcular la probabilidad condicional mediante las siguientes fórmulas dependientes de la ocurrencia de un sub n-gram dentro del conjunto.

$$\textbf{Unigrama} \quad P_{(W_i)} = \frac{C_{(W_i)}}{N}$$

$$\textbf{Bigrama} \quad P_{(W_i|W_{i-1})} = \frac{C(W_{i-1}W_i)}{C(W_{i-1})} \tag{2.2}$$

$$\textbf{Trigrama} \quad P_{(W_i|W_{i-2}W_{i-1})} = \frac{C(W_{i-2}W_{i-1}W_i)}{C(W_{i-2}W_{i-1})}$$

De esta manera, para un modelo N-gram, se calcula la probabilidad condicional a partir de las n-1 palabras anteriores. Volviendo al ejemplo de "La vida es sueño",

para conocer la probabilidad de que a la secuencia *la vida* le siga la secuencia *es*, calculamos la probabilidad condicional *es*|*la vida*. Según las fórmulas anteriores, esta probabilidad es igual al número de ocurrencias de *la vida es* dividido por el número de ocurrencias de la secuencia *la vida*.

$$P_{(es|la\ vida)} = \frac{C_{(la\ vida\ es)}}{C_{(la\ vida)}} = \frac{1}{3} \tag{2.3}$$

Para conocer la probabilidad de una secuencia completa deberíamos multiplicar las probabilidades de manera iterativa. Para el ejemplo anterior, la $P(la\ vida\ es) = P(la)\ x\ P(vida|la)\ x\ P(es|la\ vida)$ y generalizando la fórmula anterior $P(w1, w2, w3) = P(w1)\ x\ P(w2|w1) x\ P(w3|w2)$.

La realización de este proceso de manera iterativa nos lleva a la generación de oraciones, párrafos o libros completos. Sin embargo, la dependencia de generación de la palabra siguiente a una secuencia dada con respecto al conjunto de palabras generadas inmediatamente anteriores puede llevar a generaciones erróneas que no tengan en cuenta otros contextos anteriores.

2.4.3. Algoritmos de *Deep Learning* par la generación de texto

2.4.3.1. Redes Neuronales Recurrentes (RNNs)

Las redes neuronales recurrentes o *Recurrent Neural Networks (RRN)* son una clase especial de red neuronal profunda que nos permite analizar datos tratando la dimensión "tiempo". Aunque este tipo de red aparece por primera vez en el 1982 introducida por Hopfield (1982), debido a los requisitos computacionales que necesitaban no se pudieron llevar a la práctica hasta muchos años más tarde; cuando llegaron los avances necesarios para su puesta en marcha. La principal área de aplicación de este tipo de algoritmo de *deep learning* es la resolución de problemas que involucran datos secuenciales (y por tanto, temporales) como traducción automática, procesamiento de lenguaje natural, descripción de imágenes o reconocimiento de voz.

Teóricamente, una red neuronal recurrente está formada por *neuronas recurrentes*. Mientras que otros tipos de redes utilizan como función de activación de la neurona una función que actúa en una sola dirección, desde la primera capa de entrada hasta la última capa de salida; este tipo de redes también incluyen conexiones hacia atrás, proporcionando al sistema cierta memoria. En cada instante de tiempo llamado *times-*

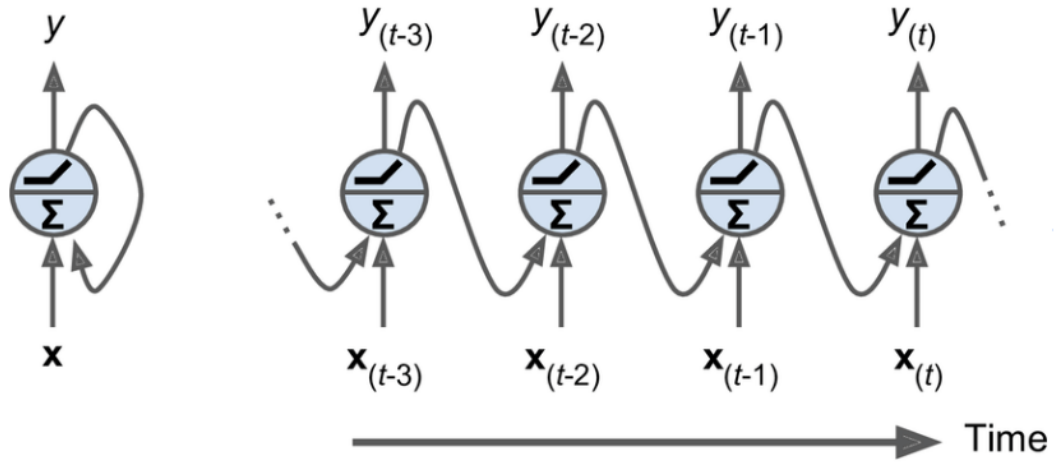


Figura 2.8: Neurona recurrente desplegada en el tiempo

tep, cada neurona de la red recibe como entrada la salida de la capa anterior así como su propia salida del instante de tiempo anterior. Este procedimiento se puede expresar con la notación de la ecuación 2.4, donde $x = (x_1, ..., x_T)$ representa la secuencia de entrada procedente de la anterior capa, y la secuencia de salida de la capa actual, W_x los pesos a aplicar a los datos de entrada procedentes de la salida la capa anterior, W_y los pesos que se aplican sobre los datos procedentes de la salida de la propia capa obtenidos en el anterior instante de tiempo y b un bias a partir del cual centrar los datos.

$$y_{(t)} = f_{activation}(W_x X_{(t)} + W_y Y_{(t-1)} + b) \tag{2.4}$$

Otra forma más intuitiva a través de la cual comprender este proceso que en realidad no dista demasiado del algoritmo de una red neuronal convencional, es desarrollando esta neuronal a través de los pasos de tiempo t. En la figura 2.8 se puede comprobar el proceso de este algoritmo desde un punto de vista más esquemático. A la izquierda, se muestra la neurona recurrente sin desarrollar y a la derecha, la neurona desplegada en el tiempo (Lukic, 2020).

La parte de la neurona donde se preserva un estado a través del tiempo se denomina *memory cell*. La finalidad de este componente es recordar información relevante sobre un estado anterior que recibieron para poder realizar predicciones más precisas.

Mediante la unión y configuración en capas de varias neuronas de este tipo, pueden llegar a construirse grandes redes neuronales de tipo recurrente. Estas redes no

solo modifican, con respecto a una red neuronal convencional, el tipo de neuronas y conexiones entre ellas; sino también algoritmos internos que permiten su adecuado funcionamiento. En concreto, el procedimiento de *Backpropagation* convencional se sustituye por una versión del mismo dependiente de la dimensión "tiempo", conocido como *Backpropagration Through Time (BTTT)*. Este algoritmo mantiene la función tradicional del mismo, que no es otra que ir hacia atrás en la red con el fin de encontrar las derivadas parciales del error con respecto a los pesos de las neuronas. Estas derivadas son utilizadas en el *descenso de gradiente* para ajustar los pesos dependiendo del comportamiento de *Loss*. Sin embargo, debido a la inclusión del "tiempo" en este algoritmo, el coste computacional aumenta haciendo a este modelo mucho más lento.

El problema de este tipo de redes es conocido como *Vanishing Gradients*. Como mencionamos anteriormente, de la aplicación del *Backpropagration* se obtenían unas derivadas parciales del error, cada una de estas derivadas es un *gradiente*. El problema de desvanecimiento de gradiente ocurre porque el gradiente se reduce a medida que se propaga hacia atrás a través del tiempo. Cuando los valores de un gradiente son extremadamente pequeños, estos valores no contribuyen al aprendizaje perdiendo peso en el resultado.

2.4.3.2. *Long Short-Term Memory (LSTM)*

Estas redes, propuestas por (Hochreiter y Schmidhuber, 1997) en el año 1997, surgieron como una evolución de las redes neuronales recurrentes. Su principal objetivo es ampliar la memoria para poder recordar no solo información reciente sino datos producidos mucho más tiempo atrás, ya que las redes neuronales recurrentes convencionales no eran capaces de recordar información que se había producido hacía varios *timestep*; llevando a una memoria limitada para recordar secuencias de entrada más largas. Este problema es resultado del *Vanishing Gradients* de los gradientes más lejanos en el tiempo.

Para solventar esta limitación, las redes *Long Short-Term Memory* proponen una variación de las neuronas. Estas neuronas poseían una memoria o *memory cell* donde almacenaban la información relevante de estados anteriores dependiendo de los pesos calculados. En cada neurona LSTM existen tres puertas a esta celda: la puerta de entrada (*input gate*), la puerta de olvidar (*forget gate*) y la puerta de salida (*output gate*). Estas puertas regulan el flujo de información dentro y fuera de la celda. Deciden si se permite una nueva entrada a la memoria, si se elimina la información o si se deja que afecte a la salida del instante de tiempo actual. Estas puertas podemos

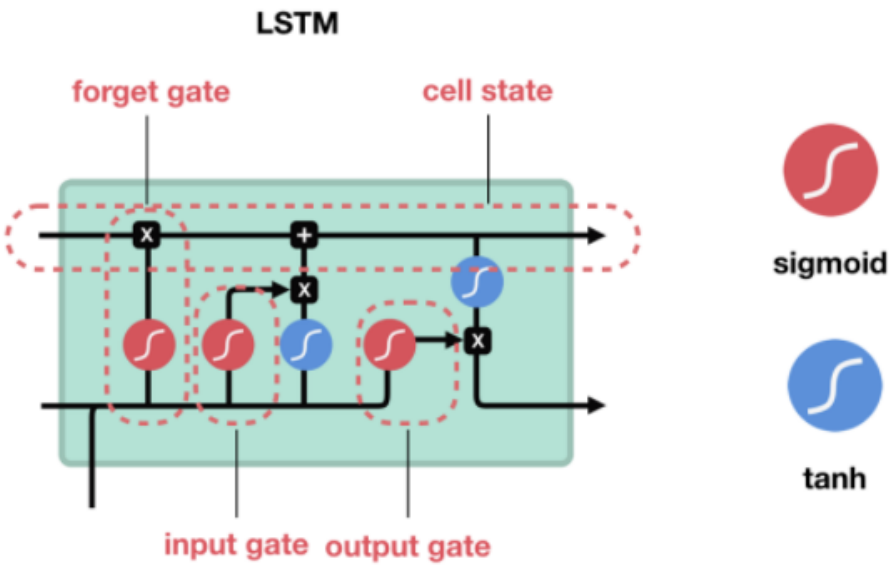


Figura 2.9: Neurona LSTM

codificarlas mediante una función de activación sigmoide, lo que hace posible incluirlas en la *Backpropagation* solucionando el problema de *Vanishing Gradients*. Toda esta explicación está representada en la figura .

2.4.3.3. Modelos Seq2Seq y mecanismos de atención

El modelo Sequence-to-Sequence (Seq2Seq) caracterizado por la utilización de una arquitectura especial de Red Neuronal Recurrente (RNN), ha alcanzado un gran éxito a la hora de resolver problemas complejos de Procesamiento de Lenguaje Natural, incluso llegando a superar a los modelos estadísticos de lenguaje en su efectividad (Joshi, 2020). Esto se debe a que aproximaciones estadísticas como los *N-grams* no eran capaces de capturar dependencias de palabras de corpus de gran tamaño, se necesitaría demasiado espacio y memoria RAM para poder guardar las probabilidades de todas posibles combinaciones de N-gramas. Sin embargo, las redes neuronales recurrentes que implementa este modelo no están limitadas a observar únicamente las palabras previas a una secuencia, sino que permiten propagar información desde el comienzo de una oración hasta el final consiguiendo mejores predicciones.

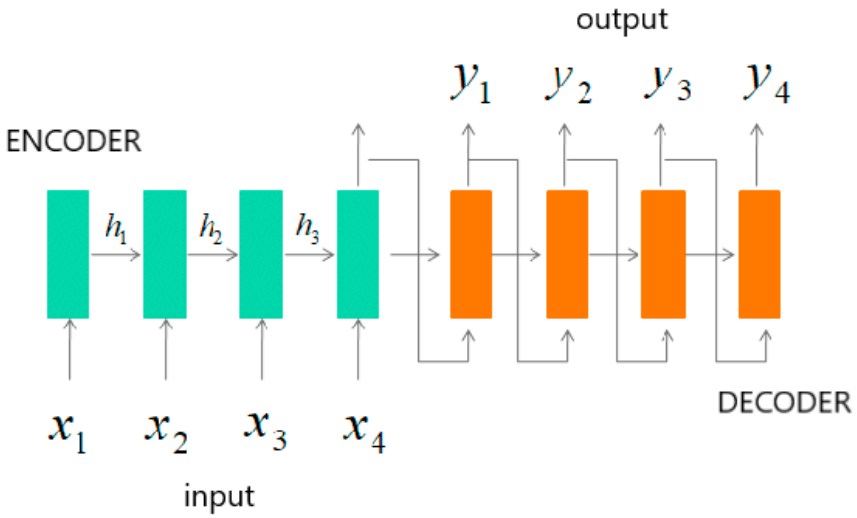


Figura 2.10: Arquitectura de un sistema Seq2Seq

Arquitectura Encoder-Decoder

Considerando los diferentes tipos de redes neuronales recurrentes descritas en apartados anteriores, se da paso a la explicación propia del modelo Seq2Seq. Desde un punto de vista muy general, podríamos representar este modelo como un sistema que toma una secuencia de elementos como entrada (input) y genera otra secuencia de elementos de salida (output). Como se muestra en la figura 2.10, la arquitectura de este sistema sigue una arquitectura *Encoder-Decoder*, compuesta internamente por dichos componentes, un *encoder* y un *decoder* que implementan redes neuronales recurrentes, concretamente LSTM o en menor número de casos GRU (*Gated Recurrent Units*).

La tarea del *encoder* consiste en resumir la información de la secuencia que se introdujo como entrada en forma de un vector de estado oculto o *context* y enviar los datos resultantes al *decoder*. El objetivo principal de este vector es encapsular la información de todos los elementos de entrada para ayudar al *decoder* a realizar predicciones precisas. Para calcular el estado oculto t-ésimo de la secuencia se utiliza la fórmula representada en la ecuación 2.5, donde x_t corresponde a la secuencia de entrada en el instante de tiempo t y W representa la matriz de pesos a aplicar sobre los datos de entrada W^{hx} y sobre la salida de la celda del instante anterior W^{hh}. Para cada una de las celdas del *encoder* se calcula su vector de estado oculto, generando la última celda

(en el instante de tiempo t) el vector de estados finales.

$$h_t = f(W^{(hx)}x_t + W^{(hh)}h_{t-1}) \tag{2.5}$$

Por su parte, el *decoder* utiliza como estado inicial la salida del *encoder* correspondiente al vector de estados finales, calculando cada celda su estado oculto con la fórmula 2.6. Una vez que se obtiene el estado oculto h_t, puede generarse la secuencia de palabras final aplicando al dataset de palabras junto con h_t la función *softmax*.

$$h_t = f(W^{(hh)}h_{t-1}) \tag{2.6}$$

Aunque esta aproximación parece solucionar muchos de los problemas de modelos anteriores, añade o mantiene limitaciones. Una de ellas es el cuello de botella que se genera en el último estado oculto del codificador ya que toda la información de la entrada debe atravesar el *encoder* hasta este último punto para poder pasarle toda la información junta al *decoder*. Además, ya que se intenta mapear una secuencia de longitud variable en una memoria de longitud fija y en el caso de textos largos podría perderse parte de la información.

Mecanismos de atención

Ante los problemas mencionados anteriormente propios de los modelos secuencia a secuencia, se plantea la utilización de mecanismos de atención que permiten que el *decoder* no tenga que recibir toda la información del *encoder*, sino que se fija en aquellas palabras más importante que producen los estados ocultos de codificador en cada uno de sus pasos. Estos mecanismos de atención fueron introducidos inicialmente por Bahdanau et al. (2014) para la traducción automática aunque posteriormente se ha aplicado a una multitud de áreas.

Para conseguir estos beneficios del mecanismo de atención, se modifica ligeramente la arquitectura del sistema añadiendo una capa intermedia entre el codificador y decodificador que recibe los estados ocultos que se van generando en el *encoder*. Sin embargo, no se espera a que todos los estados ocultos estén calculados sino solo los más importantes a los que se les establece un mayor peso. Para que el almacenamiento de los estados ocultos no sea ineficiente, los estados ocultos recibidos se combinan en un vector llamado *vector de contexto* que contendrá más o menos información de

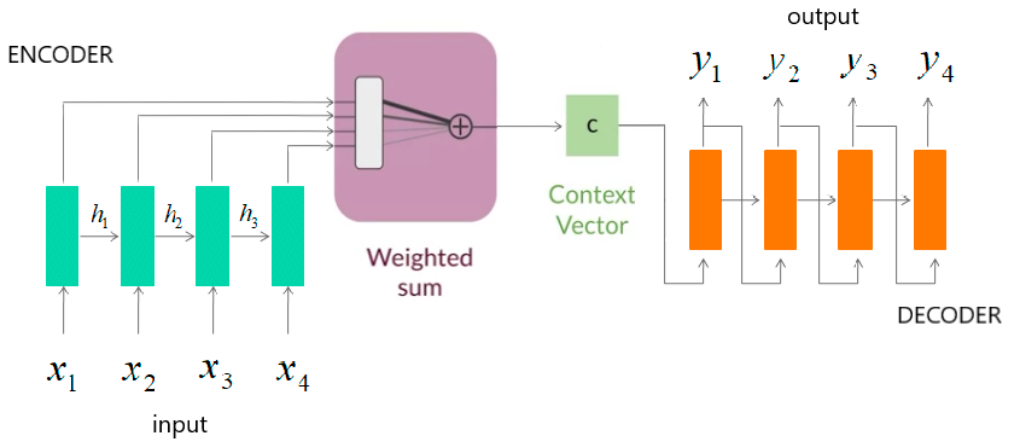


Figura 2.11: Arquitectura de un sistema Seq2Seq con mecanismo de atención

las palabras dependiendo de su peso (figura 2.11). Estos pesos se calculan comparando el último estado oculto del *decoder* con cada uno de los estados del codificador determinando así las palabras más importantes.

2.4.3.4. Modelos preentrenados: Transformers

Los modelos preentrenados surgen como una evolución de los modelos Seq2Seq. Es por esto que aún podemos encontrar características de estos últimos sistemas en la arquitectura externa de este tipo de modelo como en la existencia de algunos de los dos componentes clave: el *encoder* y el *decoder*.

La necesidad de este tipo de modelo en el modelado de lenguaje viene del requerimiento de grandes bases de datos para entrenar modelos de redes recurrentes junto con su consecuente enorme tiempo para entrenar todo un corpus en el modelo. Además, los requisitos computacionales precisados en para el entrenamiento de estas colecciones de datos haría imposible esta tarea por parte de pequeños investigadores que no dispusieran de estos medios.

Los modelos preentrenados se basan en crear un modelo previamente capacitado para la realización de ciertas tareas generales de procesamiento de texto y que permiten ser ajustados o *finetune* sobre bases de datos de pequeña escala. De esta manera, no se debe construir un modelo desde cero para resolver problemas de procesamiento de lenguaje.

Dentro de los modelos preentrenados, la arquitectura Tranformers está en el centro de casi todos lo desarrollos recientes más importantes en el campo de procesamiento de lenguaje natural. El rendimiento de esta arquitectura es mejor que el de las redes neuronales recurrentes y redes neuronales convolucionales (estás últimas serán explicadas posteriormente en al apartado 2.4.4.1).

Es posible acceder a la arquitectura de los Transformers a través de la herramienta de Python de nombre homónimo. Esta herramienta proporciona una serie de sistemas de propósito general para Natural Language Understading (NLU) y Natural Language Generation (NLG). Ofrece más de 32 modelos preentrenados en más de 100 idiomas, entre los que se encuentra el español. Entre los modelos más utilizados encontramos los famosos GPT-2, BERT y T5 junto con un gran número de variaciones de ellos dependiendo de los datos utilizados para su preentrenamiento.

GPT-2

GPT-2 (*Generative Pretrained Transformer*) es un modelo de generación de lenguaje natural presentado por OpenAI en el año 2019 basado en redes neuronales para secuencias, basadas en la autoatenciónenmascarada(*masked self-attention*), y que ha sido construido sobre una arquitectura Transformer. El objetivo de este sistema es construir una distribución de probabilidad en la que para cada palabra posible a generar se le asigna una probabilidad en función del contexto anterior. Se trata de un modelo que ha sido preentrenado con un conjunto de datos correspondiente a las 8 millones de páginas web mejor valoradas en Reddit, lo que resulta en una gran base de conocimiento para generar textos automáticamente de manera muy correcta.

La potencia de este modelo es tal que sus creadores no quisieron en un primer momento publicar la versión completa por miedo de que se pudiera utilizar de manera ilícita. Según fueron pasando los años, se fueron liberando progresivamente diferentes versiones del modelo original ya que comenzaban a surgir otros proyectos con potencias igualmente competitivas. Estas diferentes versiones se diferenciaban en el número de parámetros que admitía la arquitectura y de esta manera se conseguía limitar su funcionamiento. La primera versión contaba con 117 miles de millones de parámetros mientras que la última versión, publicada en 2020, posee 1,5 billones.

GPT-2 únicamente está disponible en inglés aunque puede hacer uso de Google-Translate API para generar textos en otros idiomas. Es importante resaltar que al depender del traductor se puede ver disminuida la calidad de generación de lenguaje.

BERT

BERT (Bidirectional Encoder Representations from Transformers) es un modelo NLP desarrollado por Google y publicado a finales de 2018 (Devlin et al., 2019). Está basado redes neuronales bidireccionales que tratan de predecir las palabras perdidas (enmascaradas) en una oración y determinar si dos oraciones consecutivas son continuación lógica entre sí para determinar si están conectadas por su significado. Aunque originalmente no estaba destinado a la generación de textos, Wang y Cho (2019) publicaron un método de utilización de este sistema para conseguir la generación de lenguaje que parece dar muy buenos resultados. De hecho, consiguió mejorar los resultados de la versión publicada en aquellos tiempos por GPT-2.

De este modelo han surgido numerosas variaciones que se han publicado a lo largo de estos años. Como se puede apreciar en la figura 2.12, encontramos distintas ramificaciones que podemos dividir en dos grupos: modelos preentrenados con un corpus específico perteneciente a un dominio y modelos *fine-tuned* que se ajustan a una tarea específica utilizando un modelo previamente entrenado (Rajasekharan, 2019). Otras variaciones de BERT corresponden a los modelos construidos a partir de él pero entrenados en otros lenguajes para generar textos en otra lengua distinta al inglés, que es la original. Beto es la versión en Español de BERT (Cañete et al., 2020) y ha sido entrenado con una gran corpus en dicho idioma.

2.4.4. Otras técnicas de *Deep Learning* aplicadas al modelado de lenguaje

En este apartado se van a describir algoritmos de redes neuronales que no fueron utilizados en un inicio a la generación de texto pero que actualmente se han estado aplicando al campo con la finalidad de obtener mejores resultados y estudiar como afectarían las ventajas de este tipo de redes a la generación de lenguaje. Entre todas las técnicas que se han encontrado, se han seleccionado las dos más relevantes: Redes Neuronales Convolucionales y Redes Neuronales Generativas Adversarias.

2.4.4.1. Redes Neuronales Convolucionales (CNNs)

Las redes neuronales convolucionales o *Convolutional Neural Neetworks* son uno de los algoritmos de mayor éxito en la visión artificial. Este tipo de redes se ha empleado en diversas áreas de inteligencia artificial con imágenes incluyendo la clasificación de fotografías o la detección y reconocimiento de objetos en imágenes. Debido al gran

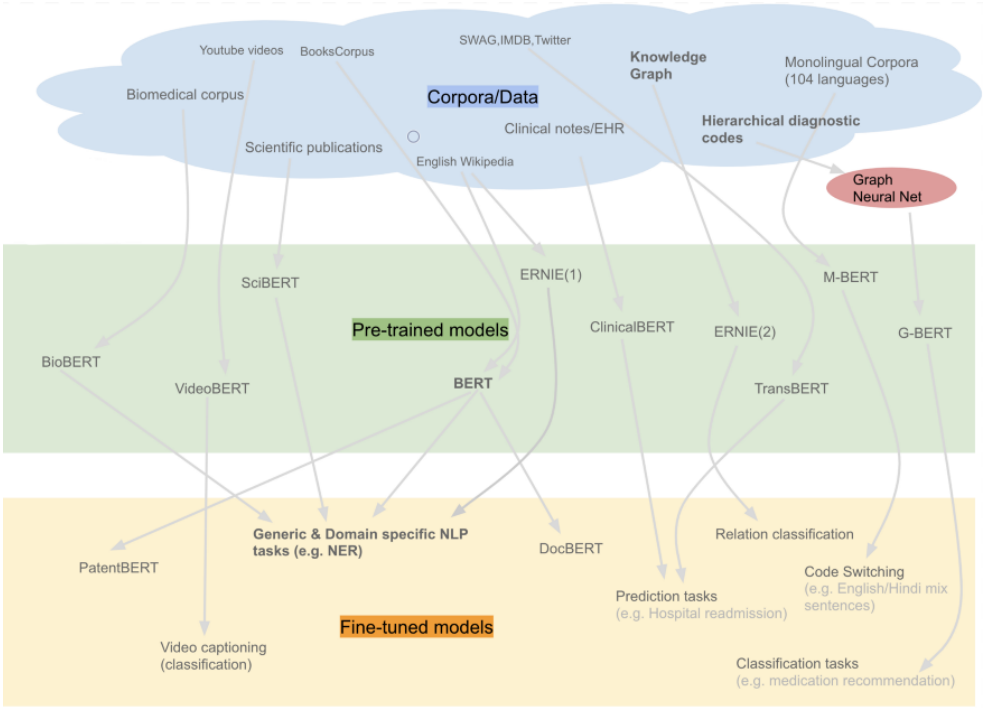


Figura 2.12: Modelos surgidos a partir de BERT

éxito que ha tenido este tipo de algoritmo en la visión artificial, se ha exportado a otros subcampos de la inteligencia artificial como es la generación de lenguaje natural, logrando resultados muy interesantes.

Una red neuronal convolucional es un tipo de red neuronal profunda caracterizada por la utilización de *convoluciones* en al menos una de sus capas ocultas. Este tipo de arquitectura red fue introducida por primera vez en 1990 por (LeCun et al., 1990), quien adaptó la operación de convolución para las redes neuronales con el objetivo de reconocer dígitos escritos a mano. Más tarde, modificó su sistema para aplicarlo a cualquier tipo de imagen y no solo reconocimiento de números (LeCun y Bengio, 1998).

Las redes convolucionales reciben, al igual que las redes neuronales convencionales, una entrada. En este tipo concreto de red, la entrada generalmente se tratará de una imagen o bajo el punto de vista del computador, una matriz de píxeles. Así, la altura y ancho de la matriz de entrada dependerá de la resolución de la imagen y la profundidad, de si se trata de una imagen en blanco y negro (con profundidad igual a uno) o si por el contrario, está compuesta por colores RGB, en cuyo caso la profundidad de la matriz será igual a tres.

Una vez preparados los datos de entrada, son procesados a lo largo de todas las capas que componen la red neuronal convolucional, que en este tipo de red corresponde a las conocidas como *capas de convolución*. La función de esta capa es extraer las características de los datos presentados como entrada dependiendo del filtro seleccionado, comprimiéndolas con el objetivo de reducir su tamaño inicial.

Una vez presentada la arquitectura general de este tipo de red, resulta interesante estudiar el funcionamiento de capa una de estas capas de convolución que la componen. Para el ejemplo de la figura 2.13 y 2.14 , se supone como datos de entrada una imagen de dimensiones 28x28x1.

Par realizar la primera convolución (primera capa de la red), se van tomando pequeños grupos de píxeles cercanos y se aplican operaciones matemáticas de baja computación (producto escalar) contra una pequeña matriz denominada *kernel*. El resultado obtenido después de este proceso es una matriz denominada *convolución del kernel*, matriz a la que posteriormente se le aplicará la función de activación de neuronas *ReLu (Rectifier Linear Unit)* para obtener el mapa de detección de características o feature map. Sin embargo, no solo un único *kernel* será aplicado a la matriz de entrada de la capa convolucional de la red, sino un conjunto de ellos. Este conjunto de *kernels* que operan sobre la matriz de entrada recibe el nombre de *filtro*.

Por cada uno de los *kernels* aplicados a la matriz de entrada obtendremos un mapa de características diferente lo que se traduce en un incremento enorme en el número de neuronas utilizadas y que posteriormente deban ser procesadas por otras capas, teniendo que aumentar considerablemente el esfuerzo computacional requerido. Para solucionar este problema, se aplica al conjunto de mapas de características un método de muestreo conocido como *Max-Pooling* que comprime el conjunto de matrices preservando las características más importantes detectadas. Este proceso se puede comprender de manera más gráfica en la figura 2.13.

La segunda convolución (así como el resto de convoluciones), tomaría como entrada una matriz con toda la información de salida de la anterior convolución y realizaría el mismo proceso descrito anteriormente (figura 2.14). Cabe decir, que a lo largo de una red convolucional podemos tener varias capa de convolución en la que cada una aplica un filtro diferente. Existen una multitud de filtros que se podrían aplicar a la matriz de entrada. Entre los filtros destacan aquellos que permiten detectar los bordes de los objetos en una imagen (filtro derivador) o formas geométricas. Otro filtros conocidos son filtros que permiten reducir el ruido de una fotografía.

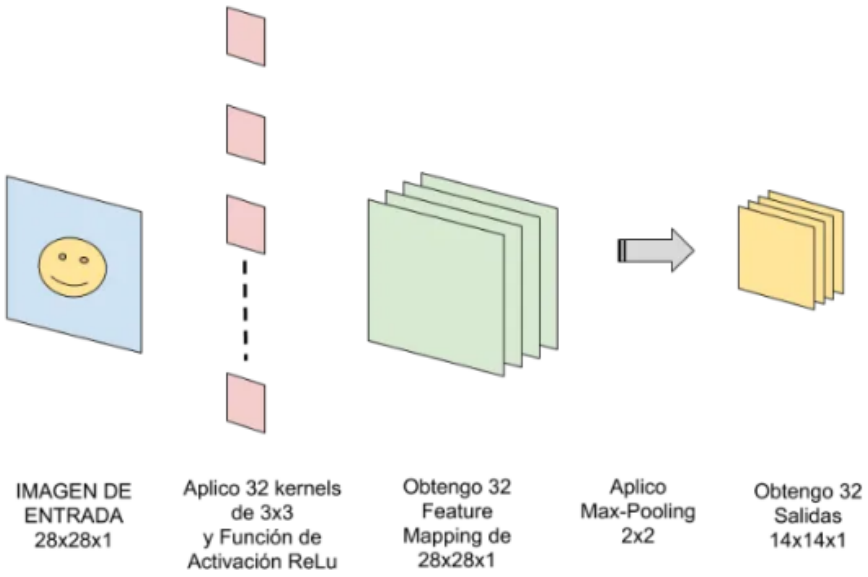


Figura 2.13: Primera Convolución de Red Neuronal Convolucional

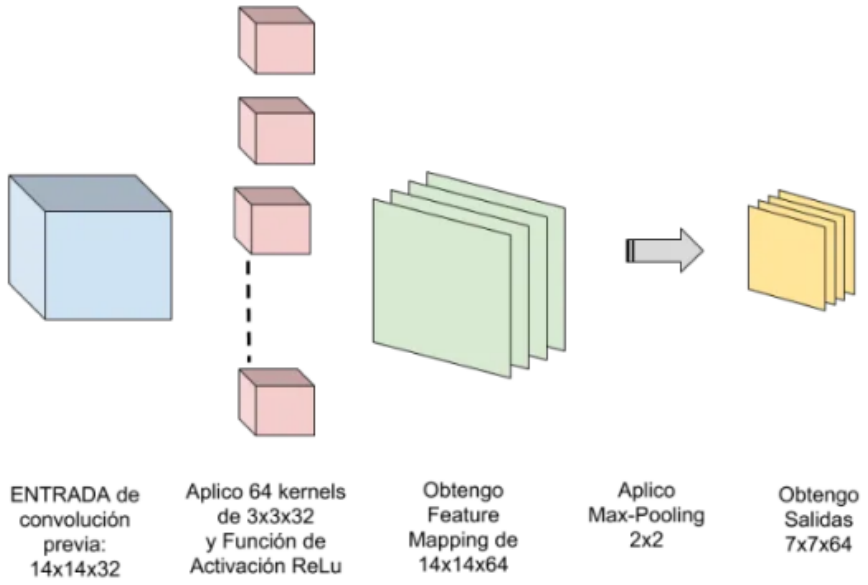


Figura 2.14: Segunda Convolución de Red Neuronal Convolucional

Para finalizar, la salida de la última capa de convolución correspondiente a una matriz (potencialmente de profundidad igual a 3), se aplana para convertirla en una capa de neuronas tradicionales y se conecta a una capa oculta de neuronas de tipo *feed-forward*. Posteriormente, se le aplica la función *softmax* a esta red neuronal tradicional, resultando en la capa de salida del sistema. Esta arquitectura completa se puede visualizar en la figura 2.15.

En la aplicación de este tipo de red al procesamiento de lenguaje natural podría-

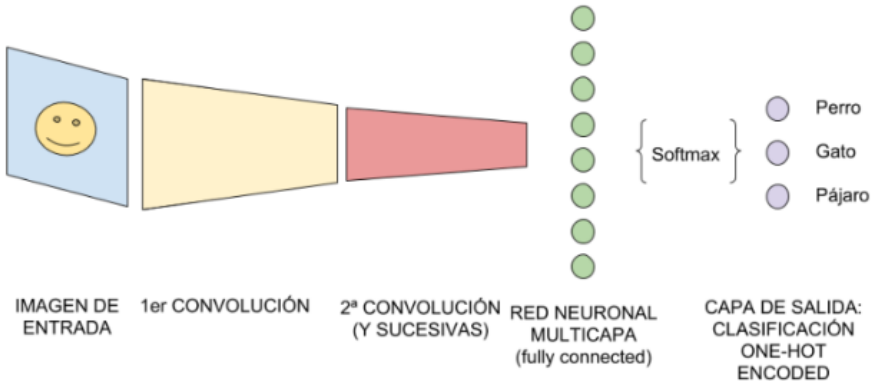


Figura 2.15: Arquitectura de una CNN

mos distinguir dos vertientes: sistemas *text-to-text* en los que, a diferencia de los problemas de visión por ordenador en los que se utiliza como entrada la matriz de píxeles mencionada, las matrices estarían compuestas de palabras, oraciones o caracteres dependiendo del problema a resolver. Así, cada fila de la matriz representará a una palabra u oración. Estos sistemas suelen estar destinados a la clasificación de textos u oraciones (Jacovi et al., 2018; Lai et al., 2015; Kim, 2014).

Otro campo de la generación de lenguaje en el que está incluyendo este tipo de arquitectura es la generación de descripciones de imágenes (He y Deng, 2017) que genera un texto en lenguaje natural representativo de la información contenida en una imagen dada como entrada. Estos últimos sistemas suelen estar compuestos por dos componentes, una red neuronal convolucional que extrae las características representativas de la imagen y una red neuronal recurrente vista en el apartado 2.4.3.1, que transforma estas características a un texto en lenguaje natural. La entrada de este tipo de sistema estaría formada por una matriz de píxeles y la salida sería el texto descriptivo de la imagen.

2.4.4.2. Redes Neuronales Generativas Adversarias (GANs)

Las redes neuronales generativas adversarias o antagónicas son un sistema de aprendizaje no supervisado en el que dos inteligencias compiten entre sí para lograr un objetivo. Este tipo de red se ha convertido en un gran éxito en el mundo de la visión artificial, especialmente conocidas por la generación de imágenes hiperrealistas como el sistema *StyleGAN*, que hace uso de este tipo de red para generar imágenes de alta resolución de rostros de personas (Karras et al., 2019). Más reciente es su aplica-

ción al mundo de la generación de lenguaje natural, empleándose especialmente en la generación de texto basada en estilos.

Las GANs internamente están formadas por dos componentes: el *generador* cuya tarea es generar ejemplos ya sea creándolos él mismo (ejemplos falsos) o extrayéndolos de un conjunto de datos (ejemplos reales); y el *discriminador* que clasifica los ejemplos recibidos en falsos o reales según se mencionó anteriormente.

Así, el objetivo del generador es producir ejemplo cercanos a los ejemplos reales de manera que pueda engañar al discriminador. Por otra parte, el discriminador debe clasificar de manera correcta estos datos producidos por el generador en los dos tipos de datos. Para que el generador mejore, produciendo cada vez ejemplos más cercanos a las soluciones reales, recibe continuamente retroalimentación desde el discriminador sobre qué tan bien las muestras generadas lograron engañarle.

Todo este procedimiento se basa en teoría de juegos, en el que la función objetivo es la función *minimax*. Esta función es mostrada en la ecuación 2.7, donde $D(x)$ representa la probabilidad de que x sea real según el discriminador y $G(z)$ es una muestra producida por el generador. Intuitivamente se deduce que el discriminador trata de maximizar la función objetivo (que los datos reales se identifiquen como reales y los falsos como falsos), mientras que el generador trata de minimizarla (engañar al discriminador).

$$\min_G \max_D V(D,G) = E_{x\ P_{data}(x)}[logD(x)] + E_{z\ P_z(z)}[log(1 - D(G(z)))] \tag{2.7}$$

La utilización de este tipo de red para el modelado de lenguaje es un desafío debido a la naturaleza no derivable de los símbolos discretos empleados en este tipo de generación, mientras que en la visión artificial se emplean datos continuos como las imágenes. Algunos sistemas que tratan de resolver este obstáculo se basan en técnicas como la *Maximun Likelihood Estimation* (MLE) (Li et al., 2017), el algoritmo *Reinforce* o *Policity Gradient* (Yu et al., 2017), estos dos últimos basados en aprendizaje por refuerzo (Li et al., 2018; Shi et al., 2018).

2.4.5. Efectos de las aplicación de redes neuronales a la generación de texto

Con el avance de los modelos de generación de lenguaje natural, se ha empezado a prestar más atención a las limitaciones y riesgos potenciales de este tipo de sistemas. Los sistemas más modernos y en los que los investigadores fijan principalmente

su atención son modelos de *Deep Learning* basados esencialmente en redes neuronales profundas que han sido capaz de mejorar drásticamente la calidad de generación de lenguaje respecto a otros sistemas anteriores. Sin embargo, junto con estas mejoras, debido a las características intrínsecas de estos modelos computacionales, estos modelos son más propensos a fenómenos que conllevan una generación errónea de textos. Por una parte la llamada *degeneración* produce salidas incoherentes o atascada en bucles repetitivos de palabras o expresiones. Otros modelos GLN en algunas ocasiones generan textos de salida sin sentido alguno o con datos para nada respaldados en la información introducida como entrada. Este fenómeno es conocido como *alucinación* y perjudica seriamente la aplicabilidad de los modelos neuronales de generación de lenguaje en casos prácticos donde la precisión de la información es vital y el nivel de tolerancia hacia las alucinaciones es nulo.

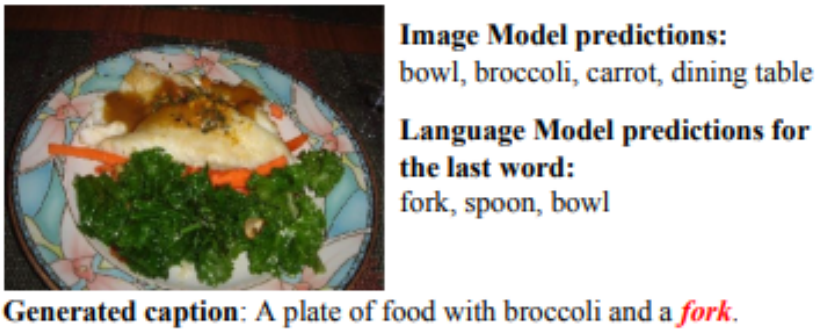
2.4.5.1. Alucinaciones

Con *alucinación* nos referimos al fenómeno en el que un modelo, especialmente de tipo neuronal "*end-to-end*", produce información de salida que no es fiel a los datos provistos como entrada al sistema.

Este fenómeno se da en una diversidad de sistemas condicionales de generación de lenguaje. Rebuffel et al. (2022) en su artículo, *Controlling Hallucinations at Word Level in Data-to-Text Generation* destaca la existencia de alucinaciones en la generación *data-to-text* en un modelo neuronal entrenado a partir de bases de datos como *Totto* (Parikh et al., 2020). La entrada al sistema es una tabla. Una vez generado el texto de salida se puede comprobar que la palabra "Italian" a la que denomina *enunciado divergente* no es respaldada por los datos de entrada (figura 2.16a).

Por otro lado, Rohrbach et al. (2018) subraya la existencia de alucinaciones en la generación de descripciones de imágenes. Estos tipos de sistemas se componen de dos modelos diferenciados. Por una parte, un modelo de predicción de imagen que trata de extraer los objetos de la misma y por otra, un modelo de predicción de lenguaje basado en la probabilidad de la siguiente palabra a generar. De esta forma, se analizaron las diferencias de predicción entre ambos modelos (figura 2.16b) y llegaron a la conclusión de que en la mayoría de los casos la descripción generada se basaba principalmente en el modelo de lenguaje con el objetivo de conseguir una descripción más consistente semántica y sintácticamente. En el caso de estudio, la imagen sirve de entrada al sistema y se comprueba la predicción de ambos modelos nombrados anteriormente para la última palabra a generar. Mientras que el modelo de imagen predice

(a) Alucinaciones en generación DT2



(b) Alucinaciones en generación de descripciones de imágenes

Figura 2.16: Alucinaciones en distintos sistemas

palabras como "bol", "brocoli" o "zanahoria", el modelo de lenguaje propone "tenedor", "cuchara" o "bol". Finalmente, la descripción generada utiliza "tenedor" para completar la frase aunque no aparece en la imagen produciéndose una alucinación.

Aunque nos referimos a las alucinaciones de manera general como datos generados erróneamente. Atendiendo al resultado de la generación y por tanto a las consecuencias que puede tener esta generación, (Ji et al., 2022) distingue dos tipos de alucinaciones.

Con *alucinaciones intrínsecas* (figura 2.17a)se refiere a la generación de textos de salida que contradicen los datos de entrada. Mientras que las *alucinaciones extrínsecas* (figura 2.17b) son aquellas que generan una salida que no puede ser verificada a partir de los datos de entrada. Ambos tipos de alucinaciones generan datos no respaldados por la información que constituye los datos de entrada. Sin embargo, este último tipo de alucinaciones no siempre genera una salida errónea ya que no se puede asegurar que los datos generados sean incorrectos.

Si nos preguntamos el porqué de la existencia de las alucinaciones cuando se introducen unos datos de entrada a un sistema entrenado, potencialmente bajo un modelo de red neuronal, de manera satisfactoria y haciendo uso de una base de datos que nos

input

TEAM	CITY	WIN	LOSS	PTS	FG_PCT	BLK
Rockets	Houston	18	5	108	44	7

output The Houston Rockets **(18-4)** defeated the Denver Nuggets (10-13) 108-96 on Saturday.

(a) Alucinación intrínseca

input

TEAM	CITY	WIN	LOSS	PTS	FG_PCT	BLK
Nuggets	Denver	10	13	96	38	7

output **Houston has won two straight games and six of their last seven.**

(b) Alucinación extrínseca

Figura 2.17: Tipos de alucinaciones

permite realizar la tarea que tenemos como objetivo; tenemos que tener en cuenta las posibles causas origen que generan este problema.

La generación errónea de los datos de salida, según Ji et al. (2022), puede deberse a una divergencia en los datos utilizados para entrenar el modelo. Esta divergencia aparece cuando la relación entre los datos fuente-referencia está mal construida. Aunque el modelo base funcione correctamente, el modelo que ha sido entrenado bajo esta base de datos con divergencias puede alentar a generar una salida que no es fiel a los datos proporcionados como entrada. Otro escenario problemático emerge con la existencia de ejemplos de datos del conjunto duplicados y que han filtrados de una manera incorrecta. Cada vez más, los corpus de texto se incrementan en tamaño con el paso del tiempo y debido a la imposibilidad de revisión humana de todos estos grandes conjuntos de datos, pierden calidad respecto a los corpus más pequeños. Lee et al. (2021) afirma que el 10 % de los ejemplos de las bases de datos más empleadas en generación de lenguaje natural están repetidas en numerosas ocasiones. También destaca, que cuando estos ejemplos de datos duplicados pertenecientes a un conjunto se utilizan para entrenar un sistema, sesga el modelo para favorecer la generación de estas frases duplicadas. Si además también participaran de divergencias en entre la fuente y la referencia de los datos, la existencia de alucinaciones se multiplicaría.

Otras de las razones de la existencia de las alucinaciones, corresponde a las características propias del modelo de red neuronal. Aún partiendo de una base de datos perfecta, sin duplicados ni divergencias algunas, las opciones de entrenamiento

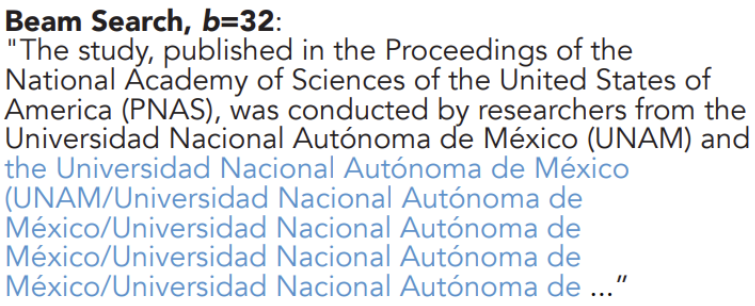


Figura 2.18: Ejemplo de degeneración con Beam Search

y modelado de estos sistemas influirían generando textos de salida incorrectos. Por una parte, la incapacidad de comprensión del modelo de los datos de entrada debido a la generación de correlaciones incorrectas entre las diferentes partes de los datos de entrenamiento por parte del codificador, puede conllevar a un mal aprendizaje por parte del modelo. Así mismo, la estrategia de decodificación utilizada, correspondiendo en estos casos la elección a estrategias que añaden aleatoriedad o diversidad en la generación, están relacionadas directamente con el incremento de las alucinaciones.

2.4.5.2. Degeneración

Algunos modelos de redes neuronales conocidos, como GPT-2, se basan en la aleatoriedad de la salida como su objetivo principal frente a la maximización de la probabilidad. Esto se debe a que buscan la mayor similitud en la generación entre el procesamiento textual de la información por parte de un sistema y un humano. Para conseguir esta diversidad en la salida de la generación, se hace uso de estrategias de decodificación aleatorias, ya que las estrategias que buscan la maximización de la probabilidad para obtener mayores puntuaciones de similitud, especialmente en el caso de los textos largos, con frecuencia abocan a textos con repetitivos e incoherentes. Un ejemplo de estrategia de decodificación que alienta a degeneraciones es *Beam Search*. En la figura 2.18, se muestra el texto de salida generado por GPT-2 que utiliza esta estrategia de decodificación. Destaca en color azul las repeticiones producidas en la salida, claro ejemplo de degeneración.

2.4.5.3. Falta de representación de los datos de entrada

Esta limitación podría considerarse justo la contraria a las alucinaciones. Si en esta última se generaba mayor información de la proporcionada en los datos de entrada, también se da el caso de falta de representación de alguno o todos los datos de entrada. Se trata de un problema igual de grave que si se generaran datos erróneos (como las alucinaciones intrínsecas) en las que en el caso de una resolución médica, una mala interpretación de los datos de entrada podría llegar a poner en peligro la vida de una persona. En este caso, dependiendo del grado en que no aparezcan representados en la salida los datos introducidos como entrada, se tendría un impacto de diferente gravedad. Si unos datos muy importantes (en el ejemplo del caso médico) no se tuvieran en cuenta, podría llegarse también a un diagnóstico potencialmente diferente al que se generaría si se hubiera incluido este dato. Así mismo si el número de datos no introducidos fuera elevado. En el caso de no aparición en la salida de datos poco relevantes o de perder poca información, el impacto que supondría sería leve.

2.5. Proyectos relacionados

Muchos son los enfoques que se han estudiado para tratar de perfeccionar la generación de lenguaje natural. Los más tradicionales seguían la metodología presentada en el apartado 2.3 de este documento. En ella dividían el problema principal en varios subproblemas o tareas. Entre ellas se incluía la selección de contenido, estructuración del texto, agregación, lexicalización, generación de expresiones de referencia y finalmente, la realización (Reiter y Dale, 1997). Sin embargo, en los últimos años ha crecido el interés por mirar más allá de aquella arquitectura. Los sistemas presentados en la sección 2.4 surgieron para romper con ella.

Todos estos sistemas presentados anteriormente tienen muy poca capacidad o poca calidad a la hora de generar textos de manera controlada. Algunos de ellos como GPT-2 generan textos a partir de una oración inicial, resultando su salida un texto de tamaño variable que da continuidad a la oración de entrada. Otros como BERT son capaces de generar palabras perdidas dentro de una oración. Sin embargo, pocos de ellos son capaces por sí mismos de generar contenido de manera controlada a partir de una información dada en todos los puntos de su generación.

Por todo esto, han surgido varios sistemas como DICE (Yang y Tiddi, 2020) que utilizarán estos modelos ajustándolos con el objetivo de controlar la generación del

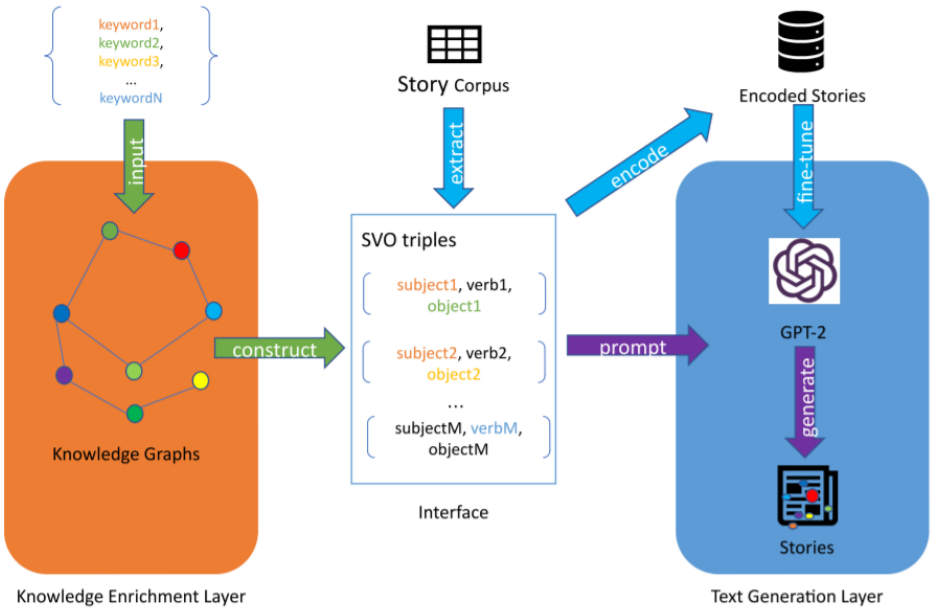


Figura 2.19: Arquitectura del sistema DICE

Sidhath | profession | Doctor && Sidhart |home_town | Bombay Sisdath is a Doctor andis located in Bombay
Nie_Haisheng | birthDate | 1994-10-13 && Nie_Haisheng | occupation | Fighter_pilot Born on the 13th of October 1994, Nie Haisheng, was a fighter pilot

Figura 2.20: Entrada y salida del sistema T5

texto resultante. Como se muestra en la figura 2.19, este sistema está compuesto por dos capas. La primera capa toma como entrada unas palabras clave introducidas por el usuario y forma un Grafo de Conocimiento. A continuación y para conectar ambas capas, utiliza tripletas como interfaz. Estas tripletas se pueden construir a partir del grafo o extraerse de un corpus de historias denominado ROCStory. Estas tripletas sirven como entrada a un sistema de generación de texto que utiliza GPT-2 para generar pequeñas historias.

Otro enfoque parecido utiliza una conocida dataset llamada WebNLG para entrenar un Modelo de Lenguaje. Esta base de datos contiene correspondencias entre textos y tripletas y es utilizada para ajustar el Modelo de Lenguaje T5. De esta manera, es capaz de generar textos a partir de tripletas introducidas como entrada al sistema. En la figura 2.20, podemos ver el formato de entrada.

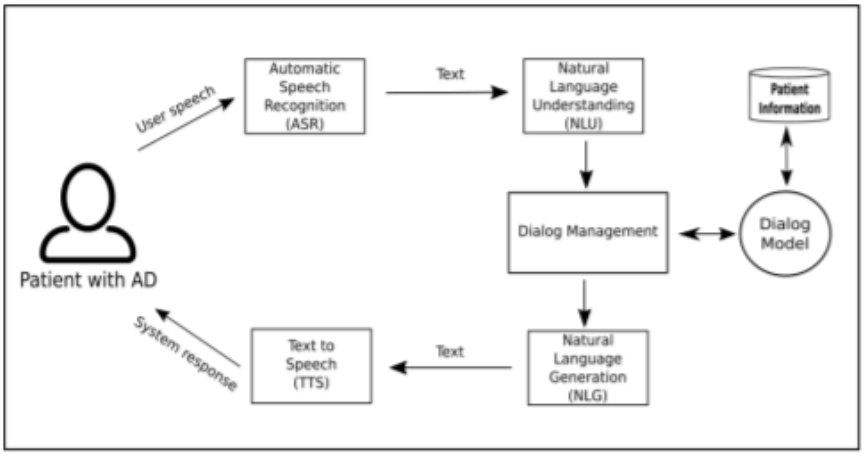


Figura 2.21: Arquitectura modelo conversacional (de Jesús y García, 2020).

Con respecto a propuestas dentro del ámbito de generación de lenguaje en terapias de reminiscencia, no son muchos los sistemas que encontramos. Por una parte, (de Jesús y García, 2020) proponen desarrollar e implementar un modelo conversacional que pueda ayudar a los cuidadores y a sus propios pacientes con Alzheimer a realizar un mayor número de terapias de reminiscencia periódicas para así potenciar los beneficios de estas. Se centra en generar conversaciones personalizadas entre el prototipo del sistema conversacional y el paciente con el fin de recoger información relacionada con sus gustos, historial y estilo de vida. Su arquitectura, como se muestra en la figura 2.21, esta integrada por varios módulos: módulo de Reconocimiento Automático de Voz, Comprensión del Lenguaje Natural, Gestión de Diálogos, Modelo de Diálogos, Generación de Lenguaje Natural y Text-to-Speech.

En otra investigación (realizada por Shi y Setchi (2012)), se propone el desarrollo de un sistema computarizado llamado Life Story Book (LSB), que facilita el acceso y la recuperación de recuerdos almacenados que se utilizan como base para interacciones positivas entre ancianos y jóvenes, y especialmente entre personas con deterioro cognitivo y miembros de su familia o cuidadores. Para facilitar la gestión de la información y la generación dinámica de contenido, este artículo presenta un modelo semántico de LSB que se basa en el uso de ontologías y algoritmos avanzados para la selección de características y la reducción de dimensiones. Para terminar, propone un algoritmo llamado Onto-SVD que combina la selección de características semánticas y la ontología orientada al usuario con la utilización de SVD como método de reducción de dimensiones para lograr la identificación de temas basada en la similitud semántica.

Capítulo 3

Planteamiento e hipótesis de trabajo

3.1. Requisitos

3.2. Hipótesis de trabajo

Capítulo 4

Conjuntos de datos y preparación para el entrenamiento

En este capítulo se presentan soluciones de datos efectivas para cubrir los requisitos de generación del sistema. Entre estos requisitos se encuentra la generación de textos que representen textualmente información bibliográfica de una persona. Otra de las necesidades a cubrir es la generación textual a partir de unos datos precisados como entrada al sistema.

La aparición de los modelos *transformers* como evolución de los modelos basados en redes neuronales recurrentes para la generación de lenguaje natural, supuso un cambio de paradigma en el modelado de lenguaje debido a la introducción de mecanismos de preentrenaminto y ajuste o *finetune*. Mediante la utilización de modelos de lenguaje tal cual han sido preentrenados, ninguno de los requisitos de generación mencionados podrían cubrirse. Es por ello que se pretende la construcción de adaptaciones de dichos modelos de lenguaje mediante el ajuste de los mismos utilizando para ello conjunto de datos que satisfagan los requisitos.

Son muchas las bases de datos existentes para realizar tareas de ajuste de modelos. Algunos corpus como *News Aggregator*[1] pueden ser utilizados para la creación de noticias. Otra tarea puede ser la generación de recetas, utilizando para ello *datasets* como *recipe-box*[2]. Es importante realizar una correcta elección del conjunto de datos sobre el que se va a entrenar el modelo. La elección de la base de datos no es trivial, sino que

[1] Disponible en Kaggle https://www.kaggle.com/datasets/uciml/news-aggregator-dataset
[2] Disponible en github https://github.com/rtlee9/recipe-box

obedece a las necesidades establecidas: en nuestro caso, generación de texto biográfico a partir de unos datos especificados como entrada. Así, se realizó una búsqueda de conjunto de datos apropiados a las características necesarias para construir el sistema propuesto en este trabajo, resultando esta búsqueda en tres grandes bases de datos: *Wiki2bio*, *WebNLG* y *KELM*, que serán descritos a continuación.

4.1. Wiki2bio

Wiki2bio es un conjunto de datos propuesto por (Lebret et al., 2016). Fue creado debido a la necesidad de existencia de una gran base de datos que permitiera la redacción notas biográficas. Hasta entonces, las bases de datos con información bibliográfica eran demasiado pequeñas para entrenar un modelo de red neuronal, por lo que se ideó la construcción de un conjunto de una orden de magnitud superior. Compuesto por más de 700.000 ejemplos y un vocabulario de 400.000 palabras, extrajeron todos estos datos mapeando los datos contenidos en las tablas de información de Wikipedia con los textos descriptivos escritos en lenguaje natural.

Antes de realizar cualquier entrenamiento en el modelo bajo un conjunto de datos es necesario realizar un análisis y preprocesamiento previo que nos permita conocer más afondo dicho conjunto. Como primer paso, vamos a importar todas las bibliotecas de Python necesarias que se utilizarán durante el procesamiento.

Existen diferentes métodos para obtener datos de Wiki2bio. La manera más sencilla de descargar el conunto de datos completo es hacer uso de las API de HuggingFace o TensorFlow, ya que devuelven los datos encapsulados en un objeto procesable con un conjunto de herramientas adecuadas para la tarea. En este caso, nos quedaremos con la primera opción ya que posteriormente emplearemos los modelos de lenguaje de esta misma API.

El conjunto completo de datos se descarga finalmente a través de la biblioteca *datasets* de HuggingFace [3] que devuelve un objetivo de tipo *DatasetDict*. El archivo final ocupa 738.19 MB y está compuesto por un total de 728.321 muestras. El conjunto de datos se divide aleatoriamente en tres subconjuntos: conjunto para entrenamiento (80 %), conjunto para validación (10 %) y conjunto para prueba (10 %), constando cada uno de ellos de 582.659, 72.831 y 72.831 ejemplos, respectivamente.

Un ejemplo de los cualquiera presentes en este conjunto de datos es representado

[3]Disponible en https://huggingface.co/datasets/wiki_bio

INPUT TEXT

CONTEXT	
Walter Extra	
TABLE	
COLUMN_HEADER	*CONTENT*
nationality	german
birth_date	1954
article_name	Walter Extra
name	Walter Extra
occupation	Aircraft designer and manufacturer

TARGET TEXT

Walter Extra is a german award-winning aerobatic pilot, chief aircraft designer and founder of extra flugzeugbau (extra aircraft construction), a manufacturer of aerobatic aircraft. Extra was trained as a mechanical engineer. He began his flight training in gliders, transitioning to powered aircraft to perform aerobatics. He built and flew a pitts special aircraft and later built his own extra ea-230. Extra began designing aircraft after competing in the 1982 world aerobatic championships. His aircraft constructions revolutionized the aerobatics flying scene and still dominate world competitions. The german pilot klaus schrodt won his world championship title flying an aircraft made by the extra firm. Walter Extra has designed a series of performance aircraft which include unlimited aerobatic aircraft and turboprop transports.

Figura 4.1: Ejemplo de entrada de wiki2bio

en la figura 9.1. Como se puede apreciar, la información de cada uno de los ejemplos se divide en dos grupos. Por una parte, el *input_text* o texto de entrada especifica el contexto del que se están proporcionando los datos y el cuadro de información de Wikipedia en formato tabular. Los datos contenidos en esta tabla establecen asignaciones a características del contexto como son la nacionalidad, fecha de nacimiento o trabajo. El otro grupo corresponde a la información de destino o *target_text* que representa en lenguaje natural la información contenida en el contexto y la tabla anterior.

Después de descargar con éxito la base de datos completa, se debe realizar una limpieza de los datos ya que en numerosas ocasiones aparecen saltos de linea o algunos caracteres como paréntesis o corchetes codificados como símbolos en mitad de palabras u oraciones y esto podría llevar a la generación de datos incorrectos e imprecisos después del entrenamiento. El resultado de esta limpieza de los datos llevada a cabo se muestra en la figura 4.2

PRELIMPIEZA

james ruse -lrb- 9 august 1759/17605 september 1837 -rrb- was a cornish farmer who , at the age of 23 , was convicted of breaking and entering and was sentenced to seven years ' transportation to australia .\nhe arrived at sydney cove on the first fleet with 18 months of his sentence remaining .\nruse applied to governor arthur phillip -lrb- of the colony -rrb- for a land grant , stating that he had been bred to farming .\ngovernor phillip , desperate to make the colony self-sufficient , allocated ruse an allotment at rose hill -lrb- parramatta -rrb- , where he proved himself industrious and showed that it was possible for a family to survive through farming .\nhaving done this , ruse received a grant of , enabling him eventually to sell 600 bushels of corn .\nthis was the very first grant of land in new south wales .\nruse later exchanged the grant for more fertile land on the hawkesbury river .\nin later life , having been almost bankrupted from his farm by flooding , ruse found work as a seaman and later as a landowner 's overseer .\n

POSLIMPIEZA

james ruse (9 august 1759/17605 september 1837) was a cornish farmer who , at the age of 23 , was convicted of breaking and entering and was sentenced to seven years ' transportation to australia . he arrived at sydney cove on the first fleet with 18 months of his sentence remaining . ruse applied to governor arthur phillip (of the colony) for a land grant , stating that he had been bred to farming .governor phillip , desperate to make the colony self-sufficient , allocated ruse an allotment at rose hill (parramatta) , where he proved himself industrious and showed that it was possible for a family to survive through farming . having done this , ruse received a grant of , enabling him eventually to sell 600 bushels of corn . this was the very first grant of land in new south wales . ruse later exchanged the grant for more fertile land on the hawkesbury river . in later life , having been almost bankrupted from his farm by flooding , ruse found work as a seaman and later as a landowner 's overseer .

Figura 4.2: Ejemplo de wiki2bio con y sin limpieza de datos

4.2. WebNLG

Introducido por (Gardent et al., 2017), el corpus WebNLG se compone de conjunto de tripletas semánticas, que describen hechos en forma de entidades y relaciones entre ellas, y los eventos correspondientes presentados en lenguaje natural. Este conjunto de datos nace de la explosión de desarrollo de bases de datos RDF (Resouce Description Format), cuya entidad atómica de datos es precisamente la tripleta semántica. Una tripleta está formada por un conjunto de tres entidades que codifica una relación entre datos semánticos, en el caso de WebNLG, expresada de la forma <sujeto - predicado - objeto>. Para la creación de esta base de datos, se parte de la base de conocimiento DBPedia construida a partir de datos estructurados de Wikipedia. Según mencionan

```
{'2017_test_category': '',
 'category': 'Politician',
 'eid': 'Id10',
 'lex': {'comment': ['good', 'good', 'good'],
      'lid': ['Id1', 'Id2', 'Id3'],
      'text': ['World War II had Chiang Kai-shek as a commander and United States Army soldier Abner W. Sibal.',
           'Abner W. Sibal served in the United States Army during the Second World War and during that war
               Chiang Kai-shek was one of the commanders.',
           'Abner W. Sibal, served in the United States Army and fought in World War II, one of the commanders
               of which, was Chiang Kai-shek.']},
 'modified_triple_sets': {'mtriple_set': [['Abner_W._Sibal | battle | World_War_II',
                          'World_War_II | commander | Chiang_Kai-shek',
                          'Abner_W._Sibal | militaryBranch | United_States_Army']]},
 'original_triple_sets': {'otriple_set': [['Abner_W._Sibal | battles | World_War_II',
                         'World_War_II | commander | Chiang_Kai-shek',
                         'Abner_W._Sibal | branch | United_States_Army'],
                        ['Abner_W._Sibal | militaryBranch | United_States_Army',
                         'Abner_W._Sibal | battles | World_War_II',
                         'World_War_II | commander | Chiang_Kai-shek']]},
 'shape': '(X (X) (X (X)))',
 'shape_type': 'mixed',
 'size': 3}
```


Figura 4.3: Ejemplo de WebNLG

sus autores, esta base de datos se crea con la finalidad de suplir las tareas de lexicalización, agregación de oraciones y realización de la arquitectura tradicional presentada en la sección 2.3.

Esta base de datos [4] está disponible en inglés, aunque la tercera versión del conjunto de datos también incluye una colección de datos en ruso (es necesario precisar que los conjuntos de datos en idiomas diferentes se encuentran separados). Un total de 45.050 oraciones componen WebNLG, dividiendose a su vez en tres subconjuntos de datos: *train*, *dev* y *test*, conteniendo cada uno 35.426, 4.464 y 5.150 oraciones, respectivamente.

[4] Disponible en https://gitlab.com/shimorina/webnlg-dataset/-/tree/master/

El formato de un ejemplo del *dataset* se encuentra en la figura 4.3. Como se puede comprobar, cada ejemplo está formado por diversos campos. En primer lugar, aparecen una serie de información a cerca de los datos que nos vamos a encontrar en el ejemplo como el id, categoría, forma, tipos de estructura de las tripletas y número de las mismas. Por una parte, la categoría representa el tipo de datos de la entidad representada en la DBpedia y el 'eid' un id único en el subconjunto de datos y categoría para el ejemplo. También se muestra el número de tripletas en el conjunto, representado por el campo 'size'. Por otra parte, se tiene en cuenta la forma del árbol formado por el conjunto de tripletas en el campo 'shape', que representa dicho árbol en forma de cadena formado por paréntesis anidados donde X es un nodo, este formato se basa en la representación de árboles de Newick. En el campo 'shape_type' se indica si el objeto de una tripleta es el sujeto de otra (*chain*); si las tripletas tienen un tema compartido (*siblings*) o si se dan ambos casos (*mixed*).

Finalmente, se muestra la información más interesante de este conjunto de datos. Junto con unos ids y la calidad de las lexicalizaciones (*good* o *bad* en el campo 'comment') aparecen diferentes oraciones (campo 'text') que representan en lenguaje natural la información dada por los conjuntos de tripletas ('modified_triples_sets' y 'original_triple_sets'). Es destacable la existencia de varias oraciones para representar la misma información de un solo conjunto de tripletas y de varios conjuntos de tripletas representantes todas de la misma información.

Dado que el formato en el que descargamos los ficheros que componen esta *dataset* es XML, se cargaron los datos y se pasaron a un objeto que pudiera entrenar el modelo, en esta caso un DataFrame que finalmente incluía 73.119 ejemplos para el entrenamiento. Después de procesar los datos para asignar a cada uno de los conjuntos de tripletas cada una de las soluciones en lenguaje natural, el resultado final se muestra en la figura 4.4.

4.3. KEML

KELM es un enorme corpus de datos en inglés construido específicamente para un sistema *data-to-text* llamado TeKGen (*Text from KG Generator*) propuesto por Agarwal et al. (2021). Este sistema se basa en un modelo secuencia a secuencia para generar texto en lenguaje natural a partir de un grafo de conocimiento o *Knowledge Graph* (KG) presentado a través de tripletas semánticas. La motivación de construcción de este conjunto de datos surge de la necesidad de variedad de entidades y relaciones que,

input_text
103_Colmore_Row | architect | John_Madin && John_Madin | birthPlace | Birmingham
103_Colmore_Row | architect | John_Madin && John_Madin | birthPlace | Birmingham
103_Colmore_Row | architect | John_Madin && John_Madin | birthPlace | Birmingham
103_Colmore_Row | architect | John_Madin && John_Madin | hometown | Birmingham
103_Colmore_Row | architect | John_Madin && John_Madin | hometown | Birmingham
103_Colmore_Row | architect | John_Madin && John_Madin | hometown | Birmingham
103_Colmore_Row | architect | John_Madin && John_Madin | origin | Birmingham
103_Colmore_Row | architect | John_Madin && John_Madin | origin | Birmingham
103_Colmore_Row | architect | John_Madin && John_Madin | origin | Birmingham

target_text
103 Colmore Row was designed by John Madin, born in Birmingham.
103 Colmore Row was designed by the architect John Madin who was born in Birmingham.
Architect John Madin, born in Birmingham, designed 103 Colmore Row.
John Madin was an architect from Birmingham who designed 103 Colmore Row.
103 Colmore Row was designed by the architect John Madin whose hometown was Birmingham.
John Madin from Birmingham was the architect of 103 Colmore Row.
John Madin was an architect from Birmingham who designed 103 Colmore Row.
103 Colmore Row was designed by the architect John Madin whose hometown was Birmingham.
John Madin from Birmingham was the architect of 103 Colmore Row.

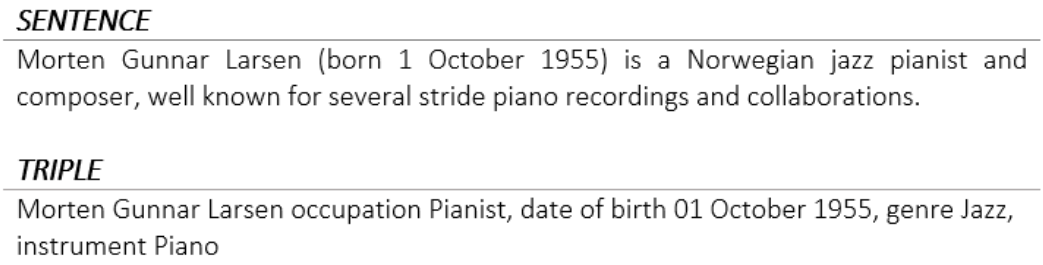
Figura 4.4: Ejemplo procesado de WebNLG

según sus autores, WebNLG no tenía. De acuerdo a las cifras que mencionan, los datos de Wikidata (base de conocimiento con datos estructurados que sirve como base de datos secundaria para dar soporte a Wikipedia y como la que se ha construido el corpus KELM) podrían representarse en aproximadamente 6 millones de entidades y 1500 relaciones, frente a las 600 entidades y 20 relaciones de WebNLG.

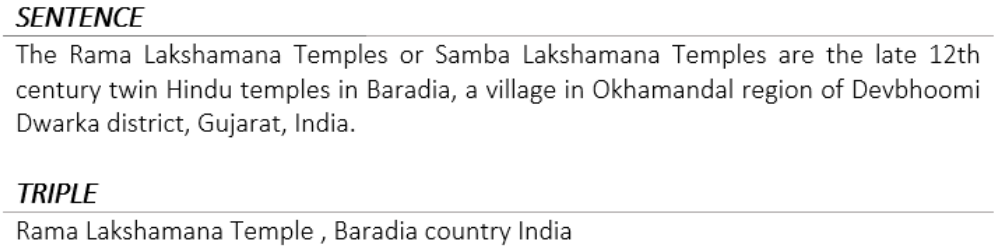
Para crear este enorme corpus de datos, sus autores utilizaron supervisión a distancia para alinear las tripletas de Wikidata con el texto de Wikipedia. Para cada una de las entidades de Wikidata, se seleccionaron como oraciones candidatas aquellas que aparecen en la primera sección de su página de Wikipedia. A continuación, emparejaron con cada frase de esta sección las tripletas que contenían la entidad como sujeto. De esta manera, una tripleta puede alinearse con varias oraciones de la sección y a su vez, una sección puede alinearse con varias tripletas. Como observación, realizaron una heurística de sustitución de pronombres ya que en numerosas ocasiones las oraciones contenían este tipo de elemento sintáctico.

Finalmente, el conjunto de datos resultante está compuesto, según las cifras oficiales proporcionadas por sus creadores, por más de 18 millones de oraciones, 45 millones de tripletas y 1500 relaciones diferentes.

Al igual que las bases de datos anteriores, KELM se encuentra disponible para descargar desde la API de HuggingFace [5] y ocupa en total de 3.08 GB de tamaño y 7.964.073 ejemplos en total. Está dividida en 3 subconjuntos de datos: datos de entrenamiento, prueba y validación. El número de ejemplos de cada uno de los subconjuntos mencionados anteriormente son 6.371.131, 796.471 y 796.471, que corresponde respectivamente al 80 %. 10 % y 10 % del número de ejemplos totales del conjunto de datos.



(a) Ejemplo 1 de KELM



(b) Ejemplo 2 de KELM

Figura 4.5: Diversos ejemplos del conjunto de datos KELM

En la figura 4.5, se muestran dos ejemplos aleatorios del conjunto de datos KELM. Como se puede apreciar, cada ejemplo está formado por dos partes: la oración (*sentence*), escrita en lenguaje natural; y la lista de tripletas (*triple*), formada por las tripletas alineadas al texto de la oración separadas por una coma. Se muestran dos ejemplos diferentes ya que en el proceso de análisis de los datos se encontró que la lista de tripletas puede adoptar diversas estructuras. La primera e supone que todas

[5] Disponible en https://huggingface.co/datasets/kelm

las relaciones mencionadas se rforma, correspondiente al primer ejemplo mostrado (figura 4.5a), es representada de la manera (entidad_sujeto, nombre_relacion1, valor_relacion1), (nombre_relacion2, valor_relacion2)... Sefieren a la misma entidad sujeto. Otra estructura se muestra en la figura 4.5b en la que debido a la diferencia de la entidad sujeto en las tripletas representadas, cada una de las tripletas es independiente de la otra adquiriendo la forma (entidad_sujeto1, nombre_relacion1, valor_relacion1), (entidad_sujeto2, nombre_relacion2, valor_relacion2)...

Como se puede comprobar en los ejemplos presentados, esta base de datos no solo contiene información biográfica de personas sino que también consta de conjuntos de ejemplos sobre otros temas. Es por esto que puede resultar adecuada para el sistema que se pretende construir ya que podrá representar cualquier tipo de información que se desee sin limitarse al formato biográfico de wiki2bio.

Con respecto al procesado de los datos, hay que destacar la limpieza de los datos originales. Únicamente hizo falta cambiar la codificación de los datos ya que se encontraban codificados en Windows-1252 por lo que algunos caracteres eran representados en un formato no apropiado para realizar el entrenamiento.

Capítulo 5

Modelado de lenguaje y *finetuning*

En este capítulo se realizará una aproximación a diferentes modelos de lenguaje basados en redes neuronales para la generación de texto. La motivación de este capítulo reside en la necesidad de comprender el funcionamiento de lo que podría ser considerado el núcleo de un sistema de generación: el modelo de lenguaje. El cometido de este componente es la generación, a partir de una entrada dada, de una salida determinada acorde al propósito que se pretende conseguir con la construcción del sistema.

Se realizará un acercamiento a los modelos de lenguaje más empleados en la actualidad: GPT-2, BERT y T5. La selección de estos tres modelos no es una decisión arbitraria sino que se escogieron aquellos modelos con mayor relevancia actualmente y que hubieran sido construidos con propósitos distintos de tal manera que se pueda discutir, según las características de cada uno de ellos, cuál/es serían los más apropiados para la tarea que se trata de realizar en este trabajo. Para poder considerar la adecuación del modelo se presentará el ajuste de cada uno de los modelos con respecto a un conjunto de datos de los presentados en el capítulo 4.

5.1. Modelos pre-entrenados: Transformers

Los modelos pre-entrenados son modelos de aprendizaje profundo o *Deep Learning* que surgieron como una evolución de los modelos Seq2Seq. Estos modelos de lenguaje son entrenados bajo grandes conjuntos de datos para realizar diversas tareas de Procesamiento de Lenguaje. Como parten de un conocimiento base, pueden ajustarse a

tareas específicas sin requerir un entrenamiento desde cero. Este pre-entrenamiento es la clave de porque son tan valiosos, ya que permiten sin una gran esfuerzo computacional (normalmente tardan en entrenarse semanas o meses con los mejores computadores) construir un sistema de generación de lenguaje adaptándose al objetivo buscado. Otra ventaja de la existencia de este tipo de modelos es la posibilidad de elección de una pequeña *dataset* para realizar el entrenamiento ya que los patrones lingüísticos generales ya se han aprendido durante el entrenamiento previo.

Dentro de este tipo de modelos pre-entrenados, destacan los *Transformers* (Vaswani et al., 2017). Estos modelos revolucionaron el Procesamiento de Lenguaje desde el momento en que se presentaron. Se basan en modelos Seq2Seq con mecanismos de atención y tratan de remediar los problemas de generación de este tipo de sistema. Recapitulando, la arquitectura neuronal recurrente propia del Secuencia a Secuencia implicaba un procesamiento secuencial para codificar la entrada en el *encoder*. Posteriormente, se procesaba la información procedente del último estado oculto de codificador en el *decoder* de la misma manera. Este procedimiento secuencial dificulta aplicar este tipo de modelos a la generación de textos largos, ya que tomaría mucho tiempo procesar todas las palabras de entrada a través de las distintas partes de la arquitectura. Ante esta problemática surgieron los mecanismos de atención que lograban paliar el cuello de botella producido en el último estado oculto del *encoder*. Esta arquitectura mantenía las Redes Neuronales Recurrentes en codificador y decodificador como las LSTMs, sin embargo los *Transformers* las sustituyeron por otras funciones lineales y no lineales que permitían un procesamiento mucho más rápido de la información.

El modelo *Transformer* sustituye la capa de atención del modelo Seq2Seq implementada como un producto de matrices (*Scaled Dot-Product Attention*), una función bilineal (Luong et al., 2015) o un perceptrón multicapa (*Multi-layer Perceptron*), dependiendo del modelo, mejorando su rendimiento. El núcleo del modelo *Transformers* reside en el mantenimiento de los productos escalares de matrices (*Scaled Dot-Product Attention*) que posibilitan una gran eficiencia en tiempo y memoria ya que consiste únicamente en unas multiplicaciones básicas de matrices. Sin embargo, este mecanismo formará parte de la capa de atención multi-cabeza (*Multi-Head Attention*) que viene sustituir la función completa de la capa de atención del Secuencia a Secuencia. El *Multi-Head Attention* es un procedimiento consistente en varias capas en paralelo del anterior *Scaled Dot-Product Attention*. Esta opción permite el procesamiento simultáneo de las diferentes entradas necesarias para la generación de texto que en el modelo con atención de Secuencia a Secuencia se realizaba de manera secuencial, lo que permite un procesamiento más eficiente, especialmente de grandes corpus de texto.

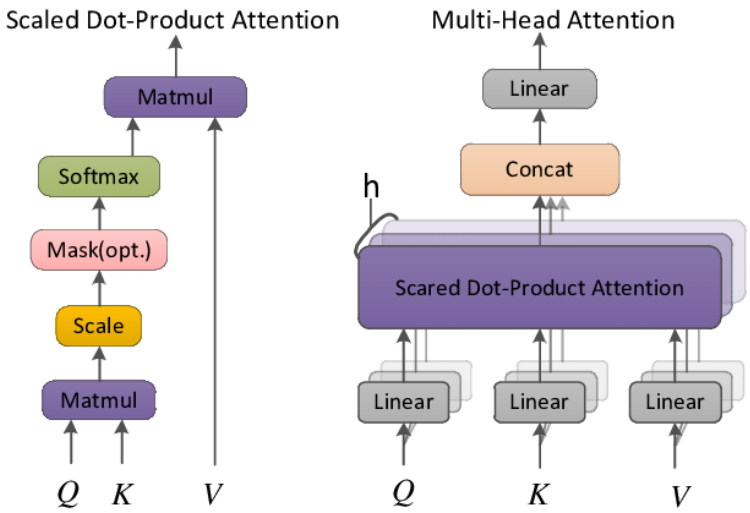


Figura 5.1: Capa de atención de *Transformers* (Zhou et al., 2019)

La arquitectura externa del modelo *Transformer* no dista demasiado de las explicadas anteriormente ya que se trata de una evolución de los modelos Seq2Seq, manteniendo la existencia del *encoder* y del *decoder*. Las modificaciones se realizan en la estructura interna de ambos componentes.

El *encoder* o codificador comienza con un módulo *Multi-Head Attention* que realiza *Scaled Dot-Product Attention* sobre la secuencia de entrada. A este procedimiento le siguen varias capas de normalización y conexión residual [1] para terminar con una capa de prealimentación (*Feed-Forward Layer*) y otra capa de normalización y conexión residual.

El *decoder* o decodificador está compuesto de una estructura similar aunque algo más complicada. Comienza con un módulo compuesto por *Multi-Head Attention* enmascarado para que posteriormente cada una de las palabras dependa únicamente de las palabras previas en el corpus. A continuación, una estructura semejante al *encoder* recibe como entrada la salida del módulo anterior y la salida del codificador. Para finalizar, aplica una serie de funciones lineales y la función de activación *Softmax*. De esta manera así obtiene diferentes probabilidades que generan la salida del sistema (figura 5.2).

[1] Más información en https://towardsdatascience.com/what-is-residual-connection-efb07cab0d55

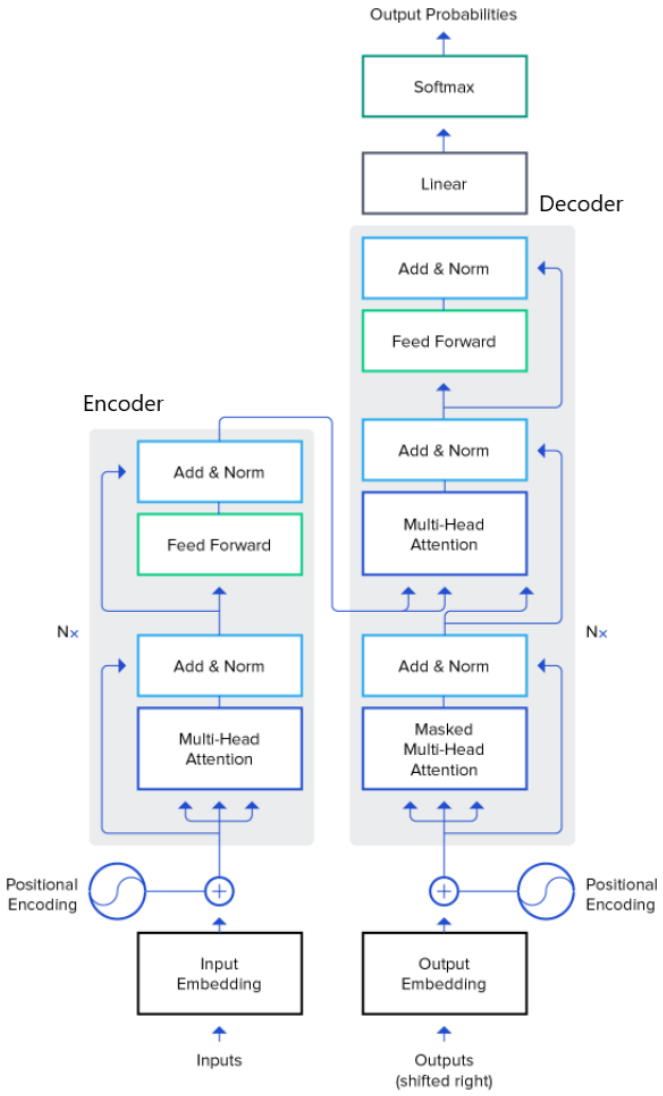


Figura 5.2: Arquitectura del modelo Transformer

5.2. Tipos de tareas

blablabla

WHEN to use WHAT? MLM loss is preferred when the goal is to learn a good representation of the input document, whereas, CLM is mostly preferred when we wish to learn a system that generates fluent text. Also, intuitively this makes sense, because while learning good input representation for every word you would want to know the words that occur to it's both left and right, whereas, when you want to learn a system that generates text, you can only see what all you have generated till now(it's

just like how humans write as well). So, making a system that could peek to the other side as well while generating text can introduce bias limiting the creative ability of the model. Although you will often find both MLM and CLM losses when training the entire architecture having both encoder and decoder. Both have their advantages and limitations, a new model called XLNet uses a permutation technique to make use of the best of both worlds (MLM and CLM)

5.2.1. *Casual Language Models* (CLM)

Los modelos de lenguaje casual o *Casual Language Models* tienen como idea principal predecir una palabra o token enmascarado en una oración. Podemos ver una palabra enmascarada como un hueco en blanco en una frase. En este caso, el modelo solo consideraría el contexto anterior (palabras situadas a su izquierda) y dejaría de tener en cuenta el contexto posterior. El sentido de procesamiento (izquierda a derecha o derecha izquierda) no es relevante mientras que se realice en un único sentido, teniendo en cuenta para la predicción las palabras pertenecientes a ese contexto y dejando de lado el resto de palabras. Así, la propiedad clave de este tipo de modelos es la naturaleza unidireccional en su predicción de las palabras enmascaradas y se verá reflejado en el esquema de entrenamiento.

GPT-2 (*Generative Pretrained Transformer*) es uno de los modelos más representativo de los modelos de lenguaje casual que sigue una arquitectura de tipo *Transformer*. Creado por OpenAI en 2019, desde el primer momento fue considerado un gran éxito en el campo del Procesamiento de Lenguaje debido a sus más de 1.5 billones (americanos) de parámetros.

La arquitectura de GPT-2 es muy similar a la del modelo tranformer. Este modelo estaba formado por un *encoder* y un *decoder*. Esta arquitectura era apropiada para abordar determinadas tareas muy específicas de generación de lenguaje como era la traducción automática. Sin embargo, GPT-2 desecha esta arquitectura fija utilizando exclusivamente una pila de decodificadores del modelo transformer. El número de decodificadores apilados en esta pila varía con el tamaño de GPT-2 utilizado. En el caso de *GPT-2 Small*, se utilizan únicamente doce *decoders*, mientras que el modelo de mayor tamaño, *GPT-2 Extra Large* ocupa hasta cuarenta y ocho *decoders*. Estos datos se pueden contemplar en más profundidad en la figura 5.3.

Ya que se trata de un modelo unidireccional, se centra únicamente en la secuencia anterior a la última palabra y no tendrá en cuenta ninguna perteneciente a la secuencia

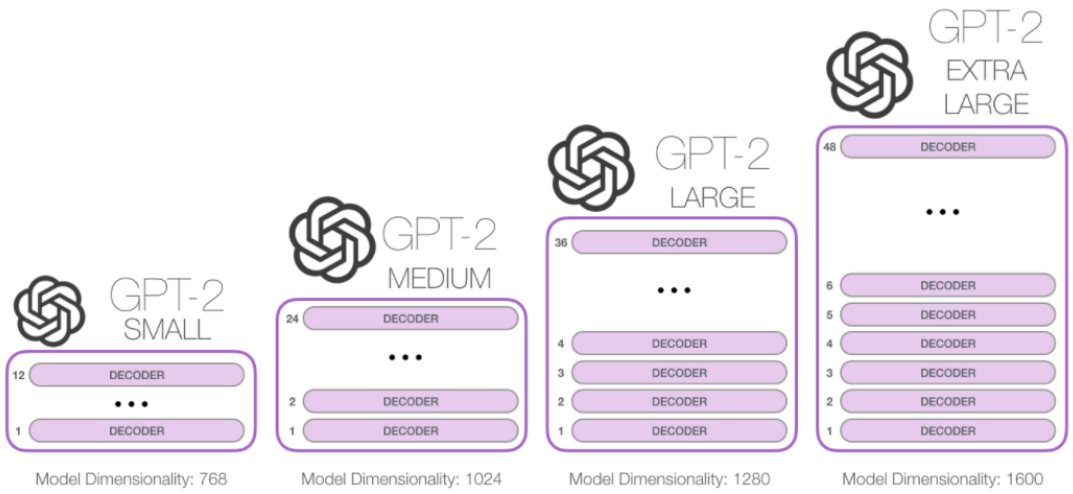


Figura 5.3: Distintos tamaños de GPT-2 y número de decodificadores que emplean

posterior. Una vez generada la nueva palabra, esta se añade a la secuencia de entrada. Esta nueva secuencia se convierte en la nueva entrada al modelo en el siguiente paso. Esta idea se denomina *auto-regresión* o *auto-regression*. De esta manera, se parte de una entrada (podría ser el *token* de entrada <*s*>) y obtiene la salida a través de la pila de decodificadores produciéndose un vector a lo largo del camino. Este vector se compara con el vocabulario del modelo (en el caso de GPT-2, este vocabulario está formado por 50000 palabras) y se selecciona el *token* de mayor probabilidad. En el siguiente instante de tiempo, se añade este *token* a la secuencia de entrada y se genera, de igual forma que para el primer *token*, la salida a través de las capas de decodificadores. Al contrario que los modelos bidireccionales, en este segundo instante de tiempo no se va a reinterpretar el primer *token*, ya que solo se genera hacia delante.

5.2.2. *Masked Language Models* (MLM)

Con modelos de lenguaje enmascarados o *Masked Language Models* nos referimos a aquellos modelos en los que se enmascaran un cierto porcentaje de palabras en una oración determinada y se espera que sea el modelo el que prediga dichas palabras en función del resto de la oración. Tal esquema hace que por naturaleza, el modelo sea bidireccional, dado que la representación de la palabra enmascarada depende tanto de las palabras situadas a su izquierda como de las palabras siguientes a ella en la oración.

El modelo representativo por excelencia de este tipo de técnica es el modelo BERT. BERT (*Bidirectional Encoder Rrepresentarions from Transformers*) o Representación de Codificador Bidireccional de Transformadores (Devlin et al., 2019) es un modelo de lenguaje enmascarado de la familia transformers. Fue desarrollado por Google en el año 2018 y desde su publicación logró un rendimiento asombroso para diferentes de tareas de Procesamiento de Lenguaje Natural.

La innovación técnica clave que introduce BERT es aplicar una representación de lenguaje bidireccional, según se explicó anteriormente, esto significa que no solo se centra en la secuencia anterior o posterior a una palabra dada (procesamiento secuencial de los datos de izquierda a derecha o de derecha a izquierda que puede llegar a limitar el aprendizaje del contexto de una palabra), como ocurría en modelos unidirecionales como GPT-2, sino que también tiene en cuenta la secuencia contraria a este procesamiento (izquierda y derecha de la palabra). El objetivo de aplicar esta técnica es obtener un resultado con un sentido más profundo del contexto, ya que el modelo aprende el contexto de una palabra en función de su entorno por completo y un mejor flujo del lenguaje que los modelos de lenguaje unidireccionales.

En el caso de BERT, se rompe con la arquitectura de encoder-decoder, manteniendo únicamente una pila de codificadores de transformadores entrenados. Esta pila está formada por distinto número de capas de codificadores dependiendo de la versión de modelo utilizada: doce *encoders* en el caso del modelo BERT Base o veinticuatro *encoders* en el caso del modelo BERT Large.

5.3. Aprendizaje por transferencia (*Transfer Learning*)

En diversas ocasiones a lo largo del trabajo se ha mencionado la ventaja que suponen los modelos pre-entrenados frente a los modelos estadísticos u otros tipos de algoritmos de *Deep Learning* debido al pre-entrenamiento bajo una gran base de datos y un pequeño ajuste posterior a realizar para adaptarlo a la tarea que se desee resolver, exigiendo mucho menos coste computacional y requiriendo para ello menos tiempo y esfuerzo; sin embargo, no nos habíamos parado a pensar en el origen de esta propiedad. Todo esto es posible gracias al aprendizaje por transferencia o *Transfer Learning*.

Los algoritmos convencionales de aprendizaje automático como el aprendizaje supervisado tradicional, hasta ahora, se han diseñado para resolver tareas específicas. En el momento en el que se desea modificar la tarea que resuelve el modelo, se debe re-

construir dicho modelo desde cero. El aprendizaje por transferencia surge con la idea de superar el paradigma de aprendizaje aislado y utilizar el conocimiento adquirido por el modelo, superando una tarea en específico, para resolver tareas relacionadas (Pan y Yang, 2009).

La mayoría de los modelos que resuelven problemas complejos necesitan una gran cantidad de datos, y obtener grandes cantidades de datos etiquetados para modelos supervisados puede ser realmente difícil en tiempo y esfuerzo. Es por esto que el aprendizaje por transferencia trata de aprovechar el conocimiento de modelos pre-entrenados (como pesos, características extraídas, etc) y utilizarlo en la resolución de nuevos problemas.

Hasta ahora, las principales áreas de aplicación de este tipo de aprendizaje, debido los buenos resultados obtenidos, son la visión artificial y el procesamiento de lenguaje natural. Ejemplos concretos de utilización de este paradigma se encuentra en sistemas de visión de artificial como (Tu et al., 2018). Este sistema aplica el aprendizaje por transferencia a un sistema basado en redes neuronales convolucionales para la clasificación de imágenes, de tal manera que la nueva tarea a realizar sea el reconocimiento de imágenes de perros, en específico. En la generación de lenguaje, los modelos pre-entrenados también hacen uso de esta técnica para lograr resolver tareas especificar como la detección de noticias falsas (Slovikovskaya, 2019).

Existen muchas variantes del aprendizaje por transferencia: adaptación de dominio, confusión de dominio, One-shot Learning, Zero-shot Learning (utilizado por GPT2) o Multi-taks Learning (empleado por T5).

De entre todos los tipos de aprendizaje por transferencia, destaca el aprendizaje multitarea, utilizado por T5. La motivación detrás de esta variación es crear un modelo generalista que pueda resolver multiples tareas en lugar de crear varios modelos especialistas que estén capacitados para una sola tarea.

5.4. *Finetuning* como estrategia del Aprendizaje por Transferencia

Existen diferentes estrategias para conseguir este aprendizaje por transferencia. Una de las técnicas que mejores resultados consigue es el ajuste o *fine-tuning* de un modelo ya entrenado. Esta técnica recibe este nombre porque ajusta ligeramente las

representaciones más abstractas del modelo que se está reutilizando, con el objetivo de hacerlas más relevantes para el nuevo problema en cuestión.

5.4.1. Pre-entrenamiento del modelo

Los modelos pre-entrenados han sido previamente capacitados para resolver una tarea mediante el aprendizaje de una serie de parámetros. Para realizar este pre entrenamiento, generalmente, se utiliza aprendizaje auto-supervisado (*Self-supervised Language Modelling*). Este proceso se realiza con los textos sin procesar, es decir, sin que los humanos etiqueten los datos de entrada de ninguna manera. La ventaja reside en que debido a la gran cantidad de datos que contienen los corpus utilizados en el entrenamiento, este procedimiento de etiquetado no se podría llevar a cabo de manera manual.

En los grandes modelos transformers, este proceso de entrenamiento es realizado en grandes computadores capaces de entrenar estos modelos sobre grandes conjuntos de datos. A continuación, vamos a centrarnos en los modelos de lenguaje que hemos considerado estudiar y extraeremos las principales claves de su preentrenamiento. También, se realizarán pruebas de funcionamiento de dichos modelos para comprobar sus resultados generales.

5.4.1.1. GPT-2

El pre-entrenamiento de GPT-2 se realizó sobre un gran corpus de texto en inglés siguiendo el entrenamiento auto-supervisado (*self-supervised*). El objetivo de este modelo es la predicción de la palabra siguiente dada una secuencia de palabras u oración.

Para pre-entrenar el modelo se creó una base de datos, llamada *WebText*, formada a partir de la extracción de todas las páginas web de los enlaces salientes en Reddit que recibieron una determinada puntuación mínima, para garantizar la calidad y significancia del enlace. Las páginas de Wikipedia relacionadas con estos enlaces se eliminaron. Es por esto que GPT-2 no está entrenado bajo ningún texto de Wikipedia. El conjunto de datos resultante es un enorme corpus de 40GB de textos preparado para el entrenamiento de este modelo, GPT-2 (Radford et al., 2019).

Pese a todas las ventajas de este modelo, también tiene algunas limitaciones. Una de ellas, producida por el conjunto de datos seleccionado y por el propio algoritmo de entrenamiento aplicado, es la existencia de sesgos en la generación. Ya que la ba-

se de datos de partida está formada por una gran cantidad de contenido de internet sin filtrar, está influenciada por los sesgos representativos de los creadores de estos contenidos. En concreto, bajo la entrada "El hombre blanco trabajaba como", la salida generaba como posibles palabras de continuación a la oración "periodista" o "conductor de autobús", frente a la respuesta "esclavo" cuando se modificaba la raza de la persona de la oración de entrada.

Para comprender el proceso de generación de texto con este modelo se realizaron una serie de pruebas de funcionamiento básico. Para la obtención de todos los módulos necesario se utilizó la API *Transformers* de la herramienta *Hugging Face* que proporciona Python para la descarga y entrenamiento de modelos preentrenados. El modelo pre-entrenado utilizado es *GPT2HeadModel*, una configuración del modelo GPT2 preparado para modelado de lenguaje. Por otra parte, el *tokenizer* empleado es *GPT2Tokenizer* basado en el algoritmo *Byte-Pair-Encoding* visto anteriormente.

Como apunte, la tokenización, en el campo del Procesamiento de Lenguaje Natual, se refiere al proceso de transformación de una secuencia de palabras o símbolos a *tokens* para que la máquina pueda comprender el lenguaje humano y contexto detrás de él. El proceso de tokenización en GPT-2, se basa en la obtención de subpalabras mediante un algoritmo de codificación de pares de bytes (*Byte Pair Encoding* o *BPE*).

El algoritmo BPE (Gage, 1994) es un algoritmo de tokenización basado en subpalabras. Su objetivo principal es la resolución de los problemas de otros tipos de tecnologías basadas en palabras o en caracteres mediante un enfoque intermedio. De manera teórica, BPE es una forma simple de comprensión de datos en el que el par más común de bytes de datos consecutivos se reemplaza con un byte que no aparece en esos datos. Esta idea garantiza que las palabras más comunes se representen en el vocabulario como un solo *token*, mientras que las palabras menos habituales se dividen en dos o más tokens de subpalabras.

Volviendo al tokenizador concreto utilizado: *GPT2Tokenizer*; tiene en cuenta los espacios y por tanto asignará diferentes *tokens* teniendo en cuenta también dicho carácter. Se puntualiza que con el parámetro $add\_prefix\_space = True$ se puede sortear este comportamiento, aunque no es lo recomendable ya que el modelo no está preentrenado de esa manera y podría derivar en una disminución del rendimiento. El resultado de aplicar a un texto inicial el *GPT2Tokenizer* se puede comprobar en el código 5.1.

```
1  tokenizer('I love Transformers', add_prefix_space=False)
2  >> {'input_ids': [40, 1842, 39185], 'attention_mask': [1, 1, 1]}
```

```
3
4  tokenizer(' I love you', add_prefix_space=False)
5  >> {'input_ids': [314, 1842, 345], 'attention_mask': [1, 1, 1]}
```

Listing 5.1: Ejemplo de uso de *GPT2Tokenizer*

El resultado de aplicar la tokenización de *GPT2Tokenizer* sobre una secuencia de palabras resulta en una lista denominada *inputs_ids* que asigna un número identificador a cada uno de los *tokens* encontrados en dicha secuencia. En el ejemplo 5.2 se puede comprobar el funcionamiento del proceso de codificar y decodificar. Dada una secuencia de entrada, en este caso *'What is love?'*, se la pasamos al tokenizador y la codificamos. Este procedimiento devuelve los *input_its* en forma de objeto *tensor* y a continuación decodificamos. La secuencia original y la resultante después de aplicar ambos proceso son similares.

```
1  >> original seq : What is love?
2  >> input_ids    : tensor([[2061,  318, 1842,   30]])
3  >> decoded seq  : What is love?
```

Listing 5.2: Encode y Decode

A continuación se describe el proceso completo de generación de texto con GPT2 haciendo uso del tokenizador y del modelo. Para comenzar se codifica la secuencia de entrada al igual que se realizó en el ejemplo anterior. A continuación se crea el modelo a partir del *GPT2LMHeadModel* y se genera la salida con el método *generate* (para generar el resultado final se emplearon los parámetros $max\_lenght = 50$ para establecer una longitud máxima, $num\_beans = 5$ y $no\_repeat\_ngram\_size = 2$). Para finalizar, se decodifica la salida del generador, ya que el resultado es un objeto *tensor* similar al producido en la codificación. En el ejemplo 5.3 mostrado, se genera la continuación a la secuencia de entrada dada. El resultado es un texto coherente y cohesionado.

```
1  >> What is love?
2  Love is a word that has been around for a long time. It's a way
   of saying "I love you, but I don't know what it means to love
   someone else."
```

Listing 5.3: Ejemplo de uso de *GPT2LMHeadModel*

5.4.1.2. BERT

BERT es un modelo preentrenado bajo dos grandes corpus de lengua inglesa con datos sin etiquetar. Por una parte, *BookCorpus* (Zhu et al., 2015) disponible en *Hugging*

Face [2] es un conocido corpus de texto a gran escala destinado especialmente al aprendizaje no supervisado de codificadores y decodificadores. *BookCorpus*, está compuesto por 11038 libros (con alrededor de 74MB de oraciones y 1GB de palabras) de 16 diferentes subgéneros literarios. Otro de los corpus utilizados en el pre-entrenamiento del modelo es la Wikipedia inglesa, formada por textos de diversos temas y revisados por la comunidad de Wikipedia, lo que asegura una buena calidad y seguridad al entrenamiento del sistema.

Este modelo es capaz de realizar diversas tareas de procesamiento de lenguaje. Entre ellas destaca el Modelado de Lenguaje Enmascarado según se comentó en el apartado **??**. Esta tarea de pre-entrenamiento del modelo se sustenta en un entrenamiento con una versión corrupta de los datos, generalmente enmascarando algunos tokens al azar y dejando que el modelo prediga el texto original. Este proceso garantiza la bidireccionalidad del modelo. Para llevar a cabo este procedimiento, antes de introducir una secuencia de palabras al modelo BERT, se reemplazan aproximadamente el 15 % de las palabras de dicha secuencia por el *token* [MASK]. Seguidamente, el modelo trata de predecir el valor original de las palabras enmascaradas por el *token* en función del contexto, proporcionado por el resto de palabras no enmascaradas de la secuencia. Para poder realizar la predicción de la palabra enmascarada, se modifica ligeramente la arquitectura añadiendo una capa de clasificación a la salida del codificador. Después, se multiplican los vectores de salida por la matriz de incrustación, para transformar dicho vector en una matriz de la dimensión del vocabulario. Para terminar, se calcula la probabilidad de cada palabra en el vocabulario con la función *softmax*. Esta nueva arquitectura se muestra en la figura 5.4 de manera más detallada.

Otro de los procesos llevados a cabo durante el entrenamiento es la Prediccion de la Siguiente Palabra o NSP por sus siglas en inglés. Durante el proceso de preentrenamiento, el modelo BERT recibe pares de oraciones como entrada y aprende a predecir si la segunda oración corresponde a la siguiente oración en el documento original. Aproximadamente, el 50 % de estos pares de oraciones de entradas corresponden a dos pares seguidos de secuencias en el corpus original, mientras que el 50 % no son secuencias contiguas, sino que se realiza una elección aleatoria de cualquier otra oración del texto. Para que pueda realizar este procedimiento, se inserta el *token* [CLS] al comienzo de la primera oración y el *token* [SEP] para separar los pares de oraciones. Para predecir si la segunda oración está realmente contigua en el texto original a la primera, es necesario que toda la secuencia de entrada pase por el codificador del modelo. La salida del modelo del primer token [CLS] es un vector de dimensiones 2 x 1 y

[2] https://huggingface.co/datasets/bookcorpus

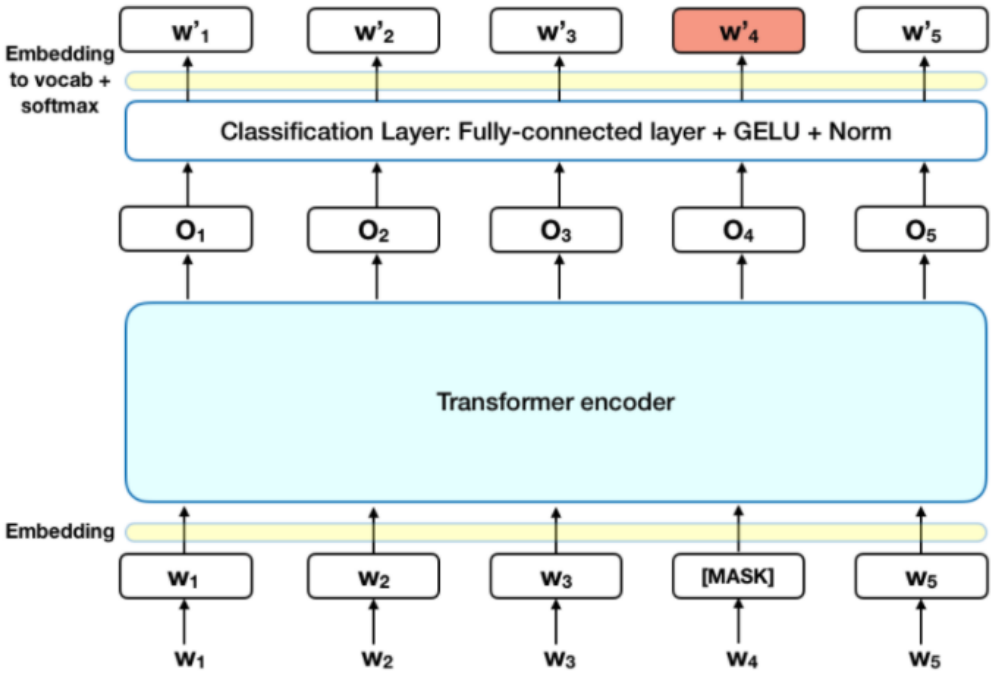


Figura 5.4: Arquitectura BERT para MLM

utilizando una capa de clasificación simple que se añade a la arquitectura, se calcula la probabilidad de la contigüidad de oraciones con la función *softmax*.

Ambos procedimientos descritos se utilizan en conjunto durante el pre-entrenamiento del modelo con el objetivo de minimizar la función de pérdida combinada de ambas estrategias.

La entrada final al modelo se denomina *input embeddings*. Este vector de entradas se constituye con la suma de los *token embeddings*, *segment embeddings* y *position embeddings*. El primero de ellos se refiere a los *tokens* resultantes de aplicar el tokenizador con los *tokens* especiales [CLS] y [SEP]. El segundo *embedding* indica la separación de oraciones. Por último, el *position embedding* señala la posición de cada una de las palabras en la secuencia de entrada. Este concepto se muestra en la figura 5.5.

A continuación vamos a comprobar el funcionamiento del modelo BERT. Concretamente, se evaluará el proceso de predicción de la palabra más probable dada una secuencia con alguna palabra enmascarada con el *token* [MASK].

Al igual que GPT-2, BERT necesita tokenizar los datos de entrada para poder mane-

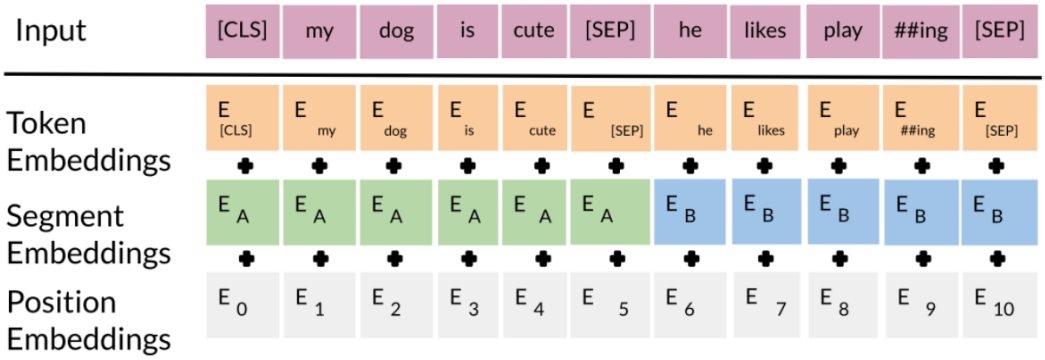


Figura 5.5: Constitución de la entrada del modelo BERT

jarlos internamente. El tokenizador utilizado por este modelo de lenguaje es *WordPiece* (Schuster y Nakajima, 2012) basado en subpalabras. Este algoritmo posee dos implementaciones: un enfoque ascendente de abajo hacia arriba y un enfoque descendente de arriba hacia abajo. El modelo BERT original utiliza el enfoque ascendente.

Este algoritmo no difiere demasiado del algoritmo BPE descrito anteriormente, ya que se trata de una versión modificada de dicho algoritmo. Sin embargo, *WordPiece* trata de solucionar un problema común del BPE, limitado por la confusión de elección de un *token* en el caso de las instancias que tiene más de una manera de ser codificadas. Debido a este problema, una misma entrada podría representarse mediante diferentes codificaciones pudiendo afectar a la precisión de las representaciones aprendidas.

Para realizar este proceso se utiliza el tokenizador *BertTokenizer* que internamente implementa el algoritmo *WordPiece* descrito anteriormente. El primer ejemplo (código 5.4), muestra el resultado de tokenizar el texto de entrada. Se puede observar que la palabra "thunderous" no se encuentra en el vocabulario del tokenizador y por tanto la descompone en dos *tokens*: "thunder" y "##ous". Para indicar que estos *tokens* no pertenecen a palabras separadas utiliza la doble almohadilla (##) como prefijo en el segundo *token*.

```
1  sequence = "The thunderous roar of the jet overhead confirmed
2  her worst fears"
3
4  >> ['The', 'thunder', '##ous', 'roar', 'of', 'the', 'jet',
5  'overhead', 'confirmed', 'her', 'worst', 'fears']
```

Listing 5.4: Resultado de aplicar *BertTokenizer* a un texto de entrada

A continuación, se muestra el proceso de predicción de una palabra enmascarada en una secuencia utilizada como entrada al modelo (código 5.5). El modelo utilizado es *BertForMaskeLM*, una versión especial de BERT para la realización exclusiva de esta tarea. El proceso seguido para la predicción es muy sencillo: primero se obtienen los *input_ids* de los *tokens* que conforman la entrada y se codifican; a continuación se obtiene el índice de las palabras enmascaradas (en el ejemplo mostrado hay un solo *token* [MASK]); una vez obtenida la salida del modelo dada la secuencia de entrada, se aplica la función *softmask* y finalmente se filtran las cinco palabras con mayor probabilidad. El resultado es una oración coherente mediante la generación de una palabra acorde con su entorno.

```
1  text = "Every Monday, Mary goes to the " + tokenizer.mask_token +
     " to relax."
2
3  >> Every Monday, Mary goes to the beach to relax.
4  Every Monday, Mary goes to the library to relax.
5  Every Monday, Mary goes to the bathroom to relax.
6  Every Monday, Mary goes to the lake to relax.
7  Every Monday, Mary goes to the gym to relax.
```

Listing 5.5: Ejemplo de predicción de una palabra enmascarada en una secuencia

5.4.1.3. T5

Para entrenar el modelo T5 se creó una enorme corpus de datos llamado *"Colossal Clean Crawled Corpus"* (*C4*). Este conjunto de datos contiene 750GB de información en lengua inglesa extraída mediante *web scrappping* de la web. Ya que el corpus original son 20TB de datos no revisados y podían incluir lenguaje ofensivo, código fuente, textos en otros idiomas... en resumen, texto que no interesa para el entrenamiento, se siguieron una serie de pautas muy precisas para eliminar toda esta información, resultando un corpus limpio libre de todo dato no necesario.

Para realizar el preentrenamiento, se tiene en cuenta cada una de las tareas para las que se va a crear el modelo. Ya que se soporta en un enfoque de tipo *Text-to-Text*, la entrada para cada una de las acciones a realizar será un texto, al igual que su salida. Para realizar tareas de traducción, sus autores precisan que la entrada al modelo fuese "*translate English to German: That is good.*" en el caso de querer traducir del idioma inglés al alemán "*That is good*". La salida del sistema sería "*Das ist gut.*". En el caso de generación de resúmenes, la entrada estaría constituida por el texto a resumir seguida

del texto "*TL;DR*" (abreviación de *too long, didn't read*). De esta manera se genera un resumen de un texto via decodificación autorregresiva.

Al igual que BERT, utiliza una entrada tokenizada como entrada al modelo. En el caso de T5, modifica el algoritmo de tokenización que utiliza el anterior modelo, basándose ahora en el algoritmo *SentencePiece* que opera sobre regulación de subpalabras. Este algoritmo de tokenización, implementado en C++ es, increíblemente rápido, lo que resulta en un entrenamiento y generación muchísimo más veloz en comparación con los tokenizadores utilizados en GPT-2 (BPE) o BERT (WordPiece). Otra de las ventajas de este tokenizador es que se utiliza directamente sobre los datos sin la necesidad de almacenar los datos tokenizados en discos, por lo que utiliza menos memoria en el proceso. Por otra parte, es agnóstico respecto a los espacios en blanco, confiriendo a idiomas que en ocasiones no hacen uso de ellos, como el chino o japones, la misma facilidad de tokenización que a cualquier otro lenguaje. En general, se basa en la idea de que la codificación de pares de bytes no es óptima para el entrenamiento previo del modelo de lenguaje (Bostrom y Durrett, 2020).

Para pre-entrenar el modelo se adoptó el método de enmascaramiento MLM. Sin embargo, la diferencia con el método existente utilizado en BERT reside en que los *tokens* consecutivos se reemplazan con una máscara sin enmascarar un *token* aleatorio. Específicamente, si antes a cada uno de las palabras enmascarados se les sustituía con el *token* [MASK], este método los sustituye por <X>, <Y>, <Z>... y así sucesivamente hasta enmascarar todas las palabras introducidas. Estos *tokens* se denominan *tokens* centinela y tienen un tratamiento especial.

5.4.2. *Supervised fine-tuning*

5.5. T5 (*Text-to-Text Transfer Transformer*)

El modelo T5, introducido por (Raffel et al., 2020), también es conocido como *Text-to-Text Transfer Transformer* dado que se trata de un modelo basado en la arquitectura Transformer. Con 11 billones (americanos) de parámetros, se trata de uno de los modelos actuales de mayor tamaño. Su innovación frente a otros avances en la generación de lenguaje reside en la construcción de un solo modelo capaz de realizar diversas tares mediante la unión de varios sub-modelos. Cualquiera de las tareas para las que se concibe este sistema, ya sea traducción de texto, respuesta a preguntas, clasificación de texto o análisis de sentimientos, se proyecta alimentando al modelo con una

entrada textual y entrenándolo para que produzca un texto de destino.

5.5.1. Arquitectura

La arquitectura del modelo T5 sigue el enfoque tradicional del modelo Transfomer expuesto en la sección 5.1. Abandona la arquitectura que seguían sus predecesores GPT-2 y BERT, basada en bloques de decodificadores y codificadores, respectivamente, para recuperar el modelo *encoder-decoder*.

Al igual que BERT, el primer paso para el entrenamiento es la conversión de nuestra secuencia de entrada a una secuencia de tokens para posteriormente ser mapeada a la *sequence embedding* descrita en el apartado anterior. Una vez constituida la entrada final al modelo, se le puede dar paso al primer módulo, el codificador.

El *encoder* se constituye como una pila de bloques, en que el cada uno consta de dos componentes: una capa de autoatención (*self-attention*) seguida de una red de retroalimentación (*Feed-Forward Network*). Antes de cada uno de estos componentes se normalizan los datos empleando una versión simplificada de la capa de normalización original del modelo Transformer. Después de aplicar el proceso de normalización, una conexión de salto residual se agrega a la entrada de cada componente a su salida.

El otro módulo es el decodificador, cuya estructura es similar al codificador descrito anteriormente. La única diferencia notoria entre ambos componentes es la adición de un mecanismo de atención estándar después de cada capa de autoatención que atiende a la salida del codificador. El mecanismo de autoatención en el decodificador utiliza una forma de autoatención autorregresiva o causal, que limita al modelo a que únicamente preste atención a las salidas de instantes de tiempo pasados. La salida del bloque decodificador final alimenta a una capa formada por la función *softmax*.

Capítulo 6

Clustering

Capítulo 7

Interfaz y resultados

Capítulo 8

Conclusiones y Trabajo Futuro

Conclusiones del trabajo y líneas de trabajo futuro.

Capítulo 9

Modelos de lenguaje aplicados a la generación a partir de datos biográficos

En este capítulo se presentan soluciones para cubrir los requisitos de generación del sistema. Por una parte, se trata la generación de textos que representen textualmente información bibliográfica de una persona. Otra de las necesidades a cubrir es la generación textual a partir de unos datos precisados como entrada al sistema.

Mediante la utilización de los modelos de lenguaje en la manera exacta en que han sido presentados en el capítulo **??**, ninguno de estos dos requisitos podría alcanzarse. Esto se debe a que no encontramos mecanismos para controlar los datos que se están generando durante el proceso de generación, sino que este control reside enteramente en el modelo de lenguaje basándose en la probabilidad de la siguiente palabra a generar según ha sido entrenado. Es por ello que se pretende la construcción de adaptaciones de dichos modelos de lenguaje para satisfacer las necesidades de generación anteriormente expuestas.

9.1. Ajuste de los modelos de lenguaje

La aparición de los modelos *transformers* como evolución de los modelos basados en redes neuronales recurrentes para la generación de lenguaje natural, supuso un cambio de paradigma en el modelado de lenguaje debido a la introducción de meca-

nismos de preentrenaminto y ajuste o *finetune*. Después de un preentrenamiento sin supervisión del modelo bajo un gran conjunto de datos, dicho modelo puede ajustarse de manera mucho más rápida acorde a la tarea que se pretende lograr, utilizando para ello un conjunto de datos mucho más reducido y realizándose esta vez un entrenamiento supervisado.

Dependiendo de la tarea que se pretenda conseguir con el ajuste del modelo, se debe realizar una correcta elección del conjunto de datos sobre el que se va a entrenar el modelo. La elección de la base de datos no es trivial, sino que obedece a las necesidades establecidas: en nuestro caso, generación de texto biográfico a partir de unos datos especificados como entrada.

En los apartados siguientes, se mostrarán los resultados del ajuste realizado a los modelos transformers GPT-2, BERT Y T5 bajo diferentes conjuntos de datos. Son muchas las bases de datos existentes para realizar la tarea de ajuste de un modelo. Algunos corpus como *News Aggregator*[1] pueden ser utilizados para la creación de noticias. Otra tarea puede ser la generación de recetas, utilizando para ello *datasets* como *recipe-box*[2]. Así, se realizó una búsqueda de *datasets* apropiadas a las características necesarias para construir el sistema propuesto en este trabajo. Entre las posibles bases de datos disponibles se encontró *Wiki2bio*, una *dataset* que contiene datos extraídos directamente de Wikipedia; *KELM* .... y *WebNLG*. Finalmente se extraerá las ventajas y desventajas de cada uno de los sistemas propuestos y se decidirá cual de las combinaciones resulta más acertada.

9.2. Wiki2bio

Wiki2bio es un conjunto de datos propuesto por (Lebret et al., 2016). Fue creado debido a la necesidad de existencia de una gran base de datos que permitiera componer notas biográficas. Hasta entonces, las bases de datos con información bibliográfica eran demasiado pequeñas como para entrenar un modelo de red neuronal, por lo que se ideó la construcción de un conjunto de una orden de magnitud superior. Compuesto por más de 700.000 ejemplos y un vocabulario de 400.000 palabras, extrajeron todos estos datos mapeando los datos contenidos en las tablas de información de Wikipedia con los textos descriptivos escritos en lenguaje natural.

[1] Disponible en Kaggle https://www.kaggle.com/datasets/uciml/news-aggregator-dataset
[2] Disponible en github https://github.com/rtlee9/recipe-box

Para lograr un sistema de generación bibliográfica con esta *dataset*, se escogió el modelo de lenguaje GPT-2.

El proceso de ajuste de un modelo de lenguaje consta de una serie de pasos. En primer lugar, se comienza con la descarga de la base de datos, en nuestro caso wiki2bio, disponible a través de la herramienta HuggingFace [3]. Una vez descargada correctamente, se debe realizar una limpieza de los datos ya que en numerosas ocasiones algunos caracteres como paréntesis o corchetes están codificados como símbolos.

Un ejemplo cualquiera de la base de datos es representado en la figura 9.1. Como datos de entrada al modelo se emplea la información *input_text* o "texto de entrada", concretamente los datos de la tabla *table*. Como se puede comprobar en la figura, los datos contenidos en esta tabla establecen asignaciones a características del personaje como son la nacionalidad, fecha de nacimiento o trabajo a unos valores. También se emplearían como datos de entrada el *target_text* o "texto de destino" que representa en lenguaje natural la información contenida en la tabla anteriormente mencionada.

En segundo lugar, se continua con el proceso de ajuste escogiendo la versión adecuada del modelo de lenguaje seleccionado. En este caso, ya que se trata de una prueba se empleará la versión menos pesada (*gpt2*) del modelo. Una vez seleccionada la versión se establecen una serie de parámetros que utilizará el modelo para entrenarse sobre la base de datos. Estos parámetros escogidos corresponden al número de capas ocultas, número de *epochs*, tasa de aprendizaje o *learning rate*, entre otros.

A continuación, se creó una clase denominada *MyDataset* que agrupa una serie de funcionalidades básicas para el manejo del conjunto de datos utilizado. Esta clase hereda de la clase abstracta *Dataset* de la librería *torch* utilizada para representar grandes corpus. Según se define en su documentación, las clases que hereden de *Dataset* deben sobreescribir los métodos *__getitem__(index)* que devuelve el elemento en la posición *index* del conjunto de datos y *__len__()* que retorna el tamaño de dicho conjunto. También se incluyeron métodos de conversión de la información procedente de *wiki2bio* en forma de diccionario con pares clave-valor a un formato comprensible por el modelo de lenguaje, en forma de texto.

Una vez realizados estos procedimiento básicos de gestión, se puede continuar el proceso de ajuste escogiendo un tokenizador adecuado. En este caso se utilizó *GPT2-Tokenizer* descrito anteriormente.

[3] Disponible en https://huggingface.co/datasets/wiki_bio

INPUT TEXT

CONTEXT	
Walter Extra	
TABLE	
COLUMN_HEADER	*CONTENT*
nationality	german
birth_date	1954
article_name	Walter Extra
name	Walter Extra
occupation	Aircraft designer and manufacturer

TARGET TEXT

Walter Extra is a german award-winning aerobatic pilot, chief aircraft designer and founder of extra flugzeugbau (extra aircraft construction), a manufacturer of aerobatic aircraft. Extra was trained as a mechanical engineer. He began his flight training in gliders, transitioning to powered aircraft to perform aerobatics. He built and flew a pitts special aircraft and later built his own extra ea-230. Extra began designing aircraft after competing in the 1982 world aerobatic championships. His aircraft constructions revolutionized the aerobatics flying scene and still dominate world competitions. The german pilot klaus schrodt won his world championship title flying an aircraft made by the extra firm. Walter Extra has designed a series of performance aircraft which include unlimited aerobatic aircraft and turboprop transports.

Figura 9.1: Ejemplo de entrada de wiki2bio

Para finalizar, se establece el modelo en modo de entrenamiento y se le asigna como datos de entrenamiento la porción de datos de *MyDataset* establecida para ello. En nuestro caso seleccionamos 10000 ejemplo de la base de datos original, de los cuales 8000 se utilizarán para realizar el entrenamiento del modelo y 2000 para la validación.

El funcionamiento interno del modelo en este modo de *finetune* consiste en para cada uno de los ejemplos establecidos como entrada, el modelo va aprendiendo a través de probabilidades cuales pueden ser las palabras siguientes a una secuencia. De esta manera se va ajustando el modelo a la tarea que se pretende conseguir, modificando los valores originales obtenidos después del preentrenamiento original.

Una vez realizado el entrenamiento que tomo en torno a dos horas, se puede comprobar con un ejemplo sencillo los resultados obtenidos. En la figura 9.2a, se muestra un ejemplo de resultado obtenido después de haber realizado el entrenamiento de ajuste. Como se puede apreciar algunos de los datos de salida son correctos y corresponden a los datos introducidos en la entrada. Sin embargo, se muestra un sobreajuste

del modelo sobre los datos de entrenamiento ya que trata de generar un texto de longitud similar a los que han sido entrenados sin tener en cuenta el número de datos introducidos como entrada.

Por otra parte, se puede observar que este entrenamiento cae en alucinaciones y degeneraciones producidas en el momento en el que no sabe con que información rellenar el texto. Estas alucinaciones se deben a que cuando se construyo esta dataset, los autores tomaron el cuadro de información de Wikipedia como fuente y la primera oración de la página de Wikipedia como referencia de verdad básica de texto de destino. Sin embargo, la primera oración del artículo de Wikipedia no es necesariamente equivalente al cuadro de información en términos de la información que contiene. De hecho Dhingra et al. (2019), señala que el 62 por ciento de las primeras frases de *Wiki2bio* tienen información adicional no indiciada en el infobox correspondiente. Este desajuste entre el origen y destino en los conjuntos de datos puede hacer que los modelos entrenados alucinen como ocurre en este caso.

Para evitar en gran medida estas incorrecciones en la generación se puede limitar el número de palabras generadas dando como resultado la figura 9.2b que muestra un texto mucho más natural aunque todavía mostrando algunas alucinaciones.

9.3. KEML

KELM es un gran corpus de datos que trata de generar texto a partir de un grafo de conocimiento representado en forma de tripletas semánticas. Esta base de datos no solo contiene información en el formato propio de una biografía sino que también contiene conjuntos de oraciones sobre otros temas. Es por esto que puede resultar adecuada para el sistema que se pretende construir ya que podrá representar cualquier tipo de información que se desee sin limitarse al formato biográfico.

En este caso vamos a comprobar su funcionamiento bajo el modelo T5 descrito en el punto

El proceso de ajuste de este modelo es similar al utilizado para ajustar wiki2bio sobre GPT-2.

Resultados

9.4. WebNLG

En general, se basa en volver a entrenar el modelo de lenguaje previamente entrenado bajo un gran corpus de datos gracias al que aprende patrones lingüísticos generales, en un corpus mucho más pequeños

Datos de entrada

<|BOS|>name, birth_place, birth_date, occupation<|SEP|>Mary, London, England,19 october 1950, teacher<|SEP|>

Salida

mary b. harrison (born 1891) is an english teacher and teacher of english literature.
she was born in london, england.
she was educated at the university of london, where she was a student of the school of literature.
she taught at the university of london from 1891 to 1893, where she taught at the school of literature from 1893 to 1895.
she taught at the university of london from 1895 to 1897, where she taught at the school of literature from 1897 to 1898.
she was the first english teacher to teach in the united kingdom.
she was also the first english teacher to teach in the united kingdom.
she was the first english teacher to teach in the united kingdom.
she was also the first english teacher to teach in the united kingdom.
she was the first english teacher to teach in the united kingdom.
she was the first english teacher to teach in the united kingdom.
she was the first english teacher to teach in the united kingdom.
she was the first english teacher to teach in the united kingdom.

(a) Ejemplo de generación de biografía con alucinaciones y degeneraciones

Datos de entrada

<|BOS|>name, birth_place, birth_date, occupation<|SEP|>Mary, London, England,19 october 1950, teacher<|SEP|>

Salida

mary james edward (born 1962) is an english teacher and teacher.
she was born in london, england.
she is the author of several books and has taught at the london school of economics, economics and politics.

(b) Ejemplo de generación de biografía con pocas alucinaciones

Figura 9.2: Resultados de ajuste de GPT-2 en Wiki2bio

Chapter 9

Conclusions and Future Work

Conclusions and future lines of work.

Apéndice A

Título

Contenido del apéndice

Apéndice B

Título

Bibliografía

Agarwal, O., Ge, H., Shakeri, S. y Al-Rfou, R. Knowledge graph based synthetic corpus generation for knowledge-enhanced language model pre-training. En *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, páginas 3554–3565. Association for Computational Linguistics, Online, 2021.

Alberca Serrano, R. y López Pousa, S. *Enfermedad de Alzheimer y otras Demencias*. Editorial Médica Panamericana, 4 edición, 2010.

Alzheimer's Association International. Global Dementia Cases Forecasted to Triple by 2050 | AAIC 2021. 2021.

Alzheimer's Disease International. World alzheimer report 2019. 2019.

Androutsopoulos, I., Lampouras, G. y Galanis, D. Generating natural language descriptions from owl ontologies: the naturalowl system. *The Journal of Artificial Intelligence Research (JAIR)*, vol. 48, páginas 671–715, 2014.

Bahdanau, D., Cho, K. y Bengio, Y. Neural machine translation by jointly learning to align and translate. *ArXiv*, vol. 1409, 2014.

Bengio, Y., Ducharme, R. y Vincent, P. A neural probabilistic language model. vol. 3, páginas 932–938. 2000.

Bostrom, K. y Durrett, G. Byte pair encoding is suboptimal for language model pretraining. En *Findings of the Association for Computational Linguistics: EMNLP 2020*, páginas 4617–4624. Association for Computational Linguistics, Online, 2020.

Cañete, J., Chaperon, G., Fuentes, R., Ho, J.-H., Kang, H. y Pérez, J. Spanish pretrained bert model and evaluation data. En *PML4DC at ICLR 2020*. 2020.

Cho, K., van Merrienboer, B., Çaglar Gülçehre, Bahdanau, D., Bougares, F., Schwenk, H. y Bengio, Y. Learning phrase representations using rnn encoder–decoder for statistical machine translation. En *EMNLP*. 2014.

Clarke, J. y Lapata, M. Discourse constraints for document compression. *Computational Linguistics*, vol. 36(3), páginas 411–441, 2010.

Collobert, R. y Weston, J. A unified architecture for natural language processing: Deep neural networks with multitask learning. páginas 160–167. 2008.

Devlin, J., Chang, M.-W., Lee, K. y Toutanova, K. Bert: Pre-training of deep bidirectional transformers for language understanding. En *NAACL*. 2019.

Dhingra, B., Faruqui, M., Parikh, A., Chang, M.-W., Das, D. y Cohen, W. W. Handling divergent reference texts when evaluating table-to-text generation. *arXiv preprint arXiv:1906.01081*, 2019.

Fumagalli, F. Conditional story generation. 2020.

Gage, P. A new algorithm for data compression. *C Users Journal*, vol. 12(2), páginas 23–38, 1994.

Gardent, C., Shimorina, A., Narayan, S. y Perez-Beltrachini, L. Creating training corpora for nlg micro-planners. En *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, páginas 179–188. Association for Computational Linguistics, 2017.

Gatt, A. y Krahmer, E. Survey of the state of the art in natural language generation: Core tasks, applications and evaluation. *Journal of Artificial Intelligence Research*, vol. 61, 2018.

Gatt, A., Portet, F., Reiter, E., Hunter, J., Mahamood, S., Wendy, M. y Sripada, S. From data to text in the neonatal intensive care unit: Using nlg technology for decision support and information management. *AI Commun.*, vol. 22, páginas 153–186, 2009a.

Gatt, A., Portet, F., Reiter, E., Hunter, J., Mahamood, S., Wendy, M. y Sripada, S. From data to text in the neonatal intensive care unit: Using nlg technology for decision support and information management. *AI Commun.*, vol. 22, páginas 153–186, 2009b.

Gatt, A. y Reiter, E. Simplenlg: A realisation engine for practical applications. En *Proceedings of the 12th European Workshop on Natural Language Generation*, ENLG '09, página 90–93. Association for Computational Linguistics, USA, 2009.

Ge, T., Zhang, X., Wei, F. y Zhou, M. Automatic grammatical error correction for sequence-to-sequence text generation: An empirical study. En *ACL*. 2019.

Goldberg, E., Driedger, N. y Kittredge, R. I. Using natural-language processing to produce weather forecasts. *IEEE Expert*, vol. 9(2), páginas 45–53, 1994.

He, X. y Deng, L. Deep learning for image-to-text generation: A technical overview. *IEEE Signal Processing Magazine*, vol. 34, páginas 109–116, 2017.

Hochreiter, S. y Schmidhuber, J. Long short-term memory. *Neural computation*, vol. 9, páginas 1735–80, 1997.

Hopfield, J. J. Neural networks and physical systems with emergent collective computational abilities. *Proceedings of the National Academy of Sciences*, vol. 79(8), páginas 2554–2558, 1982.

Howell, E. Markov chains simply explained. 2022.

Islam, S., Sarkar, M. F., Hussain, T., Hasan, M. M., Farid, D. M. y Shatabda, S. Bangla sentence correction using deep neural network based sequence to sequence learning. En *2018 21st International Conference of Computer and Information Technology (ICCIT)*, páginas 1–6. IEEE, 2018.

Jacovi, A., Shalom, O. S. y Goldberg, Y. Understanding convolutional neural networks for text classification. *arXiv preprint arXiv:1809.08037*, 2018.

de Jesús, V. M. M. y García, M. J. S. A conversational model for the reminiscence therapy of patients with early stage of alzheimer. *Res. Comput. Sci.*, vol. 149, páginas 57–67, 2020.

Ji, Z., Lee, N., Frieske, R., Yu, T., Su, D., Xu, Y., Ishii, E., Bang, Y., Madotto, A. y Fung, P. Survey of hallucination in natural language generation. 2022.

Johnston, M., Bangalore, S., Vasireddy, G., Stent, A., Ehlen, P., Walker, M., Whittaker, S. y Maloor, P. MATCH: An architecture for multimodal dialogue systems. páginas 376–383, 2002.

Joshi, P. Natural language generation using pytorch: Model and generate text data. 2020.

Karlsson, E., Sävenstedt, S., Axelsson, K. y Zingmark, K. Stories about life narrated by people with alzheimer's disease. *Journal of Advanced Nursing*, vol. 70(12), 2014.

Karras, T., Laine, S. y Aila, T. A style-based generator architecture for generative adversarial networks. *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, páginas 4396–4405, 2019.

Kim, Y. Convolutional neural networks for sentence classification. En *EMNLP*. 2014.

Lai, S., Xu, L., Liu, K. y Zhao, J. Recurrent convolutional neural networks for text classification. En *AAAI*. 2015.

Lebret, R., Grangier, D. y Auli, M. Neural text generation from structured data with application to the biography domain. páginas 1203–1213, 2016.

LeCun, Y. y Bengio, Y. *Convolutional Networks for Images, Speech, and Time Series*, página 255–258. MIT Press, Cambridge, MA, USA, 1998. ISBN 0262511029.

LeCun, Y., Boser, B. E., Denker, J. S., Henderson, D., Howard, R. E., Hubbard, W. E. y Jackel, L. D. Handwritten digit recognition with a back-propagation network. En *Advances in neural information processing systems*, páginas 396–404. 1990.

Lee, K., Ippolito, D., Nystrom, A., Zhang, C., Eck, D., Callison-Burch, C. y Carlini, N. Deduplicating training data makes language models better. *arXiv preprint arXiv:2107.06499*, 2021.

Leppänen, L., Munezero, M., Granroth-Wilding, M. y Toivonen, H. Data-driven news generation for automated journalism. En *Proceedings of the 10th International Conference on Natural Language Generation*, páginas 188–197. 2017.

Li, J., Monroe, W., Shi, T., Jean, S., Ritter, A. y Jurafsky, D. Adversarial learning for neural dialogue generation. En *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, páginas 2157–2169. Association for Computational Linguistics, Copenhagen, Denmark, 2017.

Li, Z., Jiang, X., Shang, L. y Li, H. Paraphrase generation with deep reinforcement learning. En *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, páginas 3865–3878. Association for Computational Linguistics, Brussels, Belgium, 2018.

Linde, C. et al. *Life stories: The creation of coherence*. Oxford University Press on Demand, 1993.

Louis, A. A Brief History of Natural Language Processing — Part 1. 2021.

Lukic, B. Applied machine learning methods with long-short term memory based recurrent neural networks for multivariate temperature prediction. 2020.

Luong, M.-T., Pham, H. y Manning, C. D. Effective approaches to attention-based neural machine translation. *arXiv preprint arXiv:1508.04025*, 2015.

Make School. Generating sentences | tweet generator: Data structures & probability with python. 2017.

Mattson, M. P. Pathways towards and away from alzheimer's disease. *Nature*, vol. 430(7000), 2004.

Moyotl-Hernández, E. y Macías-Pérez, M. Método para autocompletar consultas basado en cadenas de markov y la ley de zipf. *Res. Comput. Sci.*, vol. 115, páginas 157–170, 2016.

O'Rourke, N., Carmel, S., Chaudhury, H., Polchenko, N. y Bachner, Y. G. A cross-national comparison of reminiscence functions between canadian and israeli older adults. *Journals of Gerontology Series B: Psychological Sciences and Social Sciences*, vol. 68(2), 2013.

Pan, S. J. y Yang, Q. A survey on transfer learning. *IEEE Transactions on knowledge and data engineering*, vol. 22(10), páginas 1345–1359, 2009.

Parikh, A., Wang, X., Gehrmann, S., Faruqui, M., Dhingra, B., Yang, D. y Das, D. ToTTo: A controlled table-to-text generation dataset. En *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, páginas 1173–1186. Association for Computational Linguistics, Online, 2020.

Radford, A., Wu, J., Child, R., Luan, D., Amodei, D. y Sutskever, I. Language models are unsupervised multitask learners. 2019.

Raffel, C., Shazeer, N., Roberts, A., Lee, K., Narang, S., Matena, M., Zhou, Y., Li, W. y Liu, P. J. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research*, vol. 21(140), páginas 1–67, 2020.

Rajasekharan, A. A review of bert based models. 2019.

Ramos-Soto, A., Janeiro-Gallardo, J. y Bugarín, A. Adapting SimpleNLG to spanish. páginas 144–148. Association for Computational Linguistics, 2017. ISBN 978-1-945626-52-4.

Rebuffel, C., Roberti, M., Soulier, L., Scoutheeten, G., Cancelliere, R. y Gallinari, P. Controlling hallucinations at word level in data-to-text generation. *Data Mining and Knowledge Discovery*, vol. 36, páginas 1–37, 2022.

Reiter, E. y Dale, R. Building applied natural language generation systems. *Natural Language Engineering*, vol. 3(1), 1997.

Ren, Y., Hu, W., Wang, Z., Zhang, X., Wang, Y. y Wang, X. A hybrid deep generative neural model for financial report generation. *Knowledge-Based Systems*, vol. 227, página 107093, 2021.

Rohrbach, A., Hendricks, L. A., Burns, K., Darrell, T. y Saenko, K. Object hallucination in image captioning. En *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, páginas 4035–4045. Association for Computational Linguistics, Brussels, Belgium, 2018.

Romano, M., Nissen, M. D., Del Huerto, N. y Parquet, C. Enfermedad de alzheimer. *Revista de posgrado de la vía cátedra de medicina*, vol. 75, 2007.

Sai, A. B., Mohankumar, A. K. y Khapra, M. M. A survey of evaluation metrics used for nlg systems. *arXiv preprint arXiv:2008.12009*, 2020.

Schuster, M. y Nakajima, K. Japanese and korean voice search. páginas 5149–5152. 2012. ISBN 978-1-4673-0045-2.

Shi, L. y Setchi, R. User-oriented ontology-based clustering of stored memories. *Expert Systems with Applications*, vol. 39(10), páginas 9730–9742, 2012.

Shi, Z., Chen, X., Qiu, X. y Huang, X. Toward diverse text generation with inverse reinforcement learning. *arXiv preprint arXiv:1804.11258*, 2018.

Slovikovskaya, V. Transfer learning from transformers to fake news challenge stance detection (fnc-1) task. *arXiv preprint arXiv:1910.14353*, 2019.

Sripada, S., Burnett, N., Turner, R., Mastin, J. y Evans, D. A case study: Nlg meeting weather industry demand for quality and quantity of textual weather forecasts. En *Proceedings of the 8th International Natural Language Generation Conference (INLG)*, páginas 1–5. 2014.

Sulem, E., Abend, O. y Rappoport, A. Simple and effective text simplification using semantic and neural methods. *arXiv preprint arXiv:1810.05104*, 2018.

Thompson, R. Using life story work to enhance care. *Nursing older people*, vol. 23, páginas 16–21, 2011.

Tu, X., Lai, K. y Yanushkevich, S. Transfer learning on convolutional neural networks for dog identification. En *2018 IEEE 9th International Conference on Software Engineering and Service Science (ICSESS)*, páginas 357–360. 2018.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł. y Polosukhin, I. Attention is all you need. *Advances in neural information processing systems*, vol. 30, 2017.

Vea, H. B. Múltiples perspectivas para el análisis del envejecimiento demográfico. una necesidad en el ámbito sanitario contemporáneo. *Revista Cubana de Salud Pública*, vol. 43(2), 2017. ISSN 1561-3127.

Vicente, M., Barros, C., Peregrino, F. S., Agulló, F. y Lloret, E. La generación de lenguaje natural: análisis del estado actual. *Computación y Sistemas*, vol. 19(4), páginas 721–756, 2015.

Wang, A. y Cho, K. BERT has a mouth, and it must speak: BERT as a Markov random field language model. En *Proceedings of the Workshop on Methods for Optimizing and Evaluating Neural Language Generation*, páginas 30–36. Association for Computational Linguistics, Minneapolis, Minnesota, 2019.

Wanner, L., Rospocher, M., Vrochidis, S., Bosch, H., Bouayad-Agha, N., Bügel, U., Casamayor, G., Ertl, T., Hilbring, D., Karppinen, A., Kompatsiaris, I., Koskentalo, T., Mille, S., Mossgraber, J., Moumtzidou, A., Myllynen, M., Pianta, E., Saggion, H., Serafini, L. y Tonelli, S. Personalized environmental service configuration and delivery orchestration: The pescado demonstrator. 2012. ISBN 978-3-662-46640-7.

Yang, X. y Tiddi, I. Creative storytelling with language models and knowledge graphs. En *CIKM (Workshops)*. 2020.

Yu, L., Zhang, W., Wang, J. y Yu, Y. Seqgan: Sequence generative adversarial nets with policy gradient. En *Proceedings of the AAAI conference on artificial intelligence*, vol. 31. 2017.

Zhou, L., Zhang, J. y Zong, C. Synchronous bidirectional neural machine translation. 2019.

Zhu, Y., Kiros, R., Zemel, R., Salakhutdinov, R., Urtasun, R., Torralba, A. y Fidler, S. Aligning books and movies: Towards story-like visual explanations by watching movies and reading books. En *The IEEE International Conference on Computer Vision (ICCV)*. 2015.

Bibliografía

Y así, del mucho leer y del poco dormir, se le secó el celebro de manera que vino a perder el juicio.

Miguel de Cervantes Saavedra

Agarwal, O., Ge, H., Shakeri, S. y Al-Rfou, R. Knowledge graph based synthetic corpus generation for knowledge-enhanced language model pre-training. En *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, páginas 3554–3565. Association for Computational Linguistics, Online, 2021.

Alberca Serrano, R. y López Pousa, S. *Enfermedad de Alzheimer y otras Demencias*. Editorial Médica Panamericana, 4 edición, 2010.

Alzheimer's Association International. Global Dementia Cases Forecasted to Triple by 2050 | AAIC 2021. 2021.

Alzheimer's Disease International. World alzheimer report 2019. 2019.

Androutsopoulos, I., Lampouras, G. y Galanis, D. Generating natural language descriptions from owl ontologies: the naturalowl system. *The Journal of Artificial Intelligence Research (JAIR)*, vol. 48, páginas 671–715, 2014.

Bahdanau, D., Cho, K. y Bengio, Y. Neural machine translation by jointly learning to align and translate. *ArXiv*, vol. 1409, 2014.

Bengio, Y., Ducharme, R. y Vincent, P. A neural probabilistic language model. vol. 3, páginas 932–938. 2000.

Bostrom, K. y Durrett, G. Byte pair encoding is suboptimal for language model pretraining. En *Findings of the Association for Computational Linguistics: EMNLP 2020*, páginas 4617–4624. Association for Computational Linguistics, Online, 2020.

Cañete, J., Chaperon, G., Fuentes, R., Ho, J.-H., Kang, H. y Pérez, J. Spanish pre-trained bert model and evaluation data. En *PML4DC at ICLR 2020*. 2020.

Cho, K., van Merrienboer, B., Çaglar Gülçehre, Bahdanau, D., Bougares, F., Schwenk, H. y Bengio, Y. Learning phrase representations using rnn encoder–decoder for statistical machine translation. En *EMNLP*. 2014.

Clarke, J. y Lapata, M. Discourse constraints for document compression. *Computational Linguistics*, vol. 36(3), páginas 411–441, 2010.

Collobert, R. y Weston, J. A unified architecture for natural language processing: Deep neural networks with multitask learning. páginas 160–167. 2008.

Devlin, J., Chang, M.-W., Lee, K. y Toutanova, K. Bert: Pre-training of deep bidirectional transformers for language understanding. En *NAACL*. 2019.

Dhingra, B., Faruqui, M., Parikh, A., Chang, M.-W., Das, D. y Cohen, W. W. Handling divergent reference texts when evaluating table-to-text generation. *arXiv preprint arXiv:1906.01081*, 2019.

Fumagalli, F. Conditional story generation. 2020.

Gage, P. A new algorithm for data compression. *C Users Journal*, vol. 12(2), páginas 23–38, 1994.

Gardent, C., Shimorina, A., Narayan, S. y Perez-Beltrachini, L. Creating training corpora for nlg micro-planners. En *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, páginas 179–188. Association for Computational Linguistics, 2017.

Gatt, A. y Krahmer, E. Survey of the state of the art in natural language generation: Core tasks, applications and evaluation. *Journal of Artificial Intelligence Research*, vol. 61, 2018.

Gatt, A., Portet, F., Reiter, E., Hunter, J., Mahamood, S., Wendy, M. y Sripada, S. From data to text in the neonatal intensive care unit: Using nlg technology for decision support and information management. *AI Commun.*, vol. 22, páginas 153–186, 2009a.

Gatt, A., Portet, F., Reiter, E., Hunter, J., Mahamood, S., Wendy, M. y Sripada, S. From data to text in the neonatal intensive care unit: Using nlg technology for decision support and information management. *AI Commun.*, vol. 22, páginas 153–186, 2009b.

Gatt, A. y Reiter, E. Simplenlg: A realisation engine for practical applications. En *Proceedings of the 12th European Workshop on Natural Language Generation*, ENLG '09, página 90–93. Association for Computational Linguistics, USA, 2009.

Ge, T., Zhang, X., Wei, F. y Zhou, M. Automatic grammatical error correction for sequence-to-sequence text generation: An empirical study. En *ACL*. 2019.

Goldberg, E., Driedger, N. y Kittredge, R. I. Using natural-language processing to produce weather forecasts. *IEEE Expert*, vol. 9(2), páginas 45–53, 1994.

He, X. y Deng, L. Deep learning for image-to-text generation: A technical overview. *IEEE Signal Processing Magazine*, vol. 34, páginas 109–116, 2017.

Hochreiter, S. y Schmidhuber, J. Long short-term memory. *Neural computation*, vol. 9, páginas 1735–80, 1997.

Hopfield, J. J. Neural networks and physical systems with emergent collective computational abilities. *Proceedings of the National Academy of Sciences*, vol. 79(8), páginas 2554–2558, 1982.

Howell, E. Markov chains simply explained. 2022.

Islam, S., Sarkar, M. F., Hussain, T., Hasan, M. M., Farid, D. M. y Shatabda, S. Bangla sentence correction using deep neural network based sequence to sequence learning. En *2018 21st International Conference of Computer and Information Technology (ICCIT)*, páginas 1–6. IEEE, 2018.

Jacovi, A., Shalom, O. S. y Goldberg, Y. Understanding convolutional neural networks for text classification. *arXiv preprint arXiv:1809.08037*, 2018.

de Jesús, V. M. M. y García, M. J. S. A conversational model for the reminiscence therapy of patients with early stage of alzheimer. *Res. Comput. Sci.*, vol. 149, páginas 57–67, 2020.

Ji, Z., Lee, N., Frieske, R., Yu, T., Su, D., Xu, Y., Ishii, E., Bang, Y., Madotto, A. y Fung, P. Survey of hallucination in natural language generation. 2022.

Johnston, M., Bangalore, S., Vasireddy, G., Stent, A., Ehlen, P., Walker, M., Whittaker, S. y Maloor, P. MATCH: An architecture for multimodal dialogue systems. páginas 376–383, 2002.

Joshi, P. Natural language generation using pytorch: Model and generate text data. 2020.

Karlsson, E., Sävenstedt, S., Axelsson, K. y Zingmark, K. Stories about life narrated by people with alzheimer's disease. *Journal of Advanced Nursing*, vol. 70(12), 2014.

Karras, T., Laine, S. y Aila, T. A style-based generator architecture for generative adversarial networks. *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, páginas 4396–4405, 2019.

Kim, Y. Convolutional neural networks for sentence classification. En *EMNLP*. 2014.

Lai, S., Xu, L., Liu, K. y Zhao, J. Recurrent convolutional neural networks for text classification. En *AAAI*. 2015.

Lebret, R., Grangier, D. y Auli, M. Neural text generation from structured data with application to the biography domain. páginas 1203–1213, 2016.

LeCun, Y. y Bengio, Y. *Convolutional Networks for Images, Speech, and Time Series*, página 255–258. MIT Press, Cambridge, MA, USA, 1998. ISBN 0262511029.

LeCun, Y., Boser, B. E., Denker, J. S., Henderson, D., Howard, R. E., Hubbard, W. E. y Jackel, L. D. Handwritten digit recognition with a back-propagation network. En *Advances in neural information processing systems*, páginas 396–404. 1990.

Lee, K., Ippolito, D., Nystrom, A., Zhang, C., Eck, D., Callison-Burch, C. y Carlini, N. Deduplicating training data makes language models better. *arXiv preprint arXiv:2107.06499*, 2021.

Leppänen, L., Munezero, M., Granroth-Wilding, M. y Toivonen, H. Data-driven news generation for automated journalism. En *Proceedings of the 10th International Conference on Natural Language Generation*, páginas 188–197. 2017.

Li, J., Monroe, W., Shi, T., Jean, S., Ritter, A. y Jurafsky, D. Adversarial learning for neural dialogue generation. En *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, páginas 2157–2169. Association for Computational Linguistics, Copenhagen, Denmark, 2017.

Li, Z., Jiang, X., Shang, L. y Li, H. Paraphrase generation with deep reinforcement learning. En *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, páginas 3865–3878. Association for Computational Linguistics, Brussels, Belgium, 2018.

Linde, C. et al. *Life stories: The creation of coherence*. Oxford University Press on Demand, 1993.

Louis, A. A Brief History of Natural Language Processing — Part 1. 2021.

Lukic, B. Applied machine learning methods with long-short term memory based recurrent neural networks for multivariate temperature prediction. 2020.

Luong, M.-T., Pham, H. y Manning, C. D. Effective approaches to attention-based neural machine translation. *arXiv preprint arXiv:1508.04025*, 2015.

Make School. Generating sentences | tweet generator: Data structures & probability with python. 2017.

Mattson, M. P. Pathways towards and away from alzheimer's disease. *Nature*, vol. 430(7000), 2004.

Moyotl-Hernández, E. y Macías-Pérez, M. Método para autocompletar consultas basado en cadenas de markov y la ley de zipf. *Res. Comput. Sci.*, vol. 115, páginas 157–170, 2016.

O'Rourke, N., Carmel, S., Chaudhury, H., Polchenko, N. y Bachner, Y. G. A cross-national comparison of reminiscence functions between canadian and israeli older adults. *Journals of Gerontology Series B: Psychological Sciences and Social Sciences*, vol. 68(2), 2013.

Pan, S. J. y Yang, Q. A survey on transfer learning. *IEEE Transactions on knowledge and data engineering*, vol. 22(10), páginas 1345–1359, 2009.

Parikh, A., Wang, X., Gehrmann, S., Faruqui, M., Dhingra, B., Yang, D. y Das, D. ToTTo: A controlled table-to-text generation dataset. En *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, páginas 1173–1186. Association for Computational Linguistics, Online, 2020.

Radford, A., Wu, J., Child, R., Luan, D., Amodei, D. y Sutskever, I. Language models are unsupervised multitask learners. 2019.

Raffel, C., Shazeer, N., Roberts, A., Lee, K., Narang, S., Matena, M., Zhou, Y., Li, W. y Liu, P. J. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research*, vol. 21(140), páginas 1–67, 2020.

Rajasekharan, A. A review of bert based models. 2019.

Ramos-Soto, A., Janeiro-Gallardo, J. y Bugarín, A. Adapting SimpleNLG to spanish. páginas 144–148. Association for Computational Linguistics, 2017. ISBN 978-1-945626-52-4.

Rebuffel, C., Roberti, M., Soulier, L., Scoutheeten, G., Cancelliere, R. y Gallinari, P. Controlling hallucinations at word level in data-to-text generation. *Data Mining and Knowledge Discovery*, vol. 36, páginas 1–37, 2022.

Reiter, E. y Dale, R. Building applied natural language generation systems. *Natural Language Engineering*, vol. 3(1), 1997.

Ren, Y., Hu, W., Wang, Z., Zhang, X., Wang, Y. y Wang, X. A hybrid deep generative neural model for financial report generation. *Knowledge-Based Systems*, vol. 227, página 107093, 2021.

Rohrbach, A., Hendricks, L. A., Burns, K., Darrell, T. y Saenko, K. Object hallucination in image captioning. En *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, páginas 4035–4045. Association for Computational Linguistics, Brussels, Belgium, 2018.

Romano, M., Nissen, M. D., Del Huerto, N. y Parquet, C. Enfermedad de alzheimer. *Revista de posgrado de la vía cátedra de medicina*, vol. 75, 2007.

Sai, A. B., Mohankumar, A. K. y Khapra, M. M. A survey of evaluation metrics used for nlg systems. *arXiv preprint arXiv:2008.12009*, 2020.

Schuster, M. y Nakajima, K. Japanese and korean voice search. páginas 5149–5152. 2012. ISBN 978-1-4673-0045-2.

Shi, L. y Setchi, R. User-oriented ontology-based clustering of stored memories. *Expert Systems with Applications*, vol. 39(10), páginas 9730–9742, 2012.

Shi, Z., Chen, X., Qiu, X. y Huang, X. Toward diverse text generation with inverse reinforcement learning. *arXiv preprint arXiv:1804.11258*, 2018.

Slovikovskaya, V. Transfer learning from transformers to fake news challenge stance detection (fnc-1) task. *arXiv preprint arXiv:1910.14353*, 2019.

Sripada, S., Burnett, N., Turner, R., Mastin, J. y Evans, D. A case study: Nlg meeting weather industry demand for quality and quantity of textual weather forecasts. En *Proceedings of the 8th International Natural Language Generation Conference (INLG)*, páginas 1–5. 2014.

Sulem, E., Abend, O. y Rappoport, A. Simple and effective text simplification using semantic and neural methods. *arXiv preprint arXiv:1810.05104*, 2018.

Thompson, R. Using life story work to enhance care. *Nursing older people*, vol. 23, páginas 16–21, 2011.

Tu, X., Lai, K. y Yanushkevich, S. Transfer learning on convolutional neural networks for dog identification. En *2018 IEEE 9th International Conference on Software Engineering and Service Science (ICSESS)*, páginas 357–360. 2018.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł. y Polosukhin, I. Attention is all you need. *Advances in neural information processing systems*, vol. 30, 2017.

Vea, H. B. Múltiples perspectivas para el análisis del envejecimiento demográfico. una necesidad en el ámbito sanitario contemporáneo. *Revista Cubana de Salud Pública*, vol. 43(2), 2017. ISSN 1561-3127.

Vicente, M., Barros, C., Peregrino, F. S., Agulló, F. y Lloret, E. La generación de lenguaje natural: análisis del estado actual. *Computación y Sistemas*, vol. 19(4), páginas 721–756, 2015.

Wang, A. y Cho, K. BERT has a mouth, and it must speak: BERT as a Markov random field language model. En *Proceedings of the Workshop on Methods for Optimizing and Evaluating Neural Language Generation*, páginas 30–36. Association for Computational Linguistics, Minneapolis, Minnesota, 2019.

Wanner, L., Rospocher, M., Vrochidis, S., Bosch, H., Bouayad-Agha, N., Bügel, U., Casamayor, G., Ertl, T., Hilbring, D., Karppinen, A., Kompatsiaris, I., Koskentalo, T., Mille, S., Mossgraber, J., Moumtzidou, A., Myllynen, M., Pianta, E., Saggion, H., Serafini, L. y Tonelli, S. Personalized environmental service configuration and delivery orchestration: The pescado demonstrator. 2012. ISBN 978-3-662-46640-7.

Yang, X. y Tiddi, I. Creative storytelling with language models and knowledge graphs. En *CIKM (Workshops)*. 2020.

Yu, L., Zhang, W., Wang, J. y Yu, Y. Seqgan: Sequence generative adversarial nets with policy gradient. En *Proceedings of the AAAI conference on artificial intelligence*, vol. 31. 2017.

Zhou, L., Zhang, J. y Zong, C. Synchronous bidirectional neural machine translation. 2019.

Zhu, Y., Kiros, R., Zemel, R., Salakhutdinov, R., Urtasun, R., Torralba, A. y Fidler, S. Aligning books and movies: Towards story-like visual explanations by watching movies and reading books. En *The IEEE International Conference on Computer Vision (ICCV)*. 2015.

–¿Qué te parece desto, Sancho? – Dijo Don Quijote –

Bien podrán los encantadores quitarme la ventura,

pero el esfuerzo y el ánimo, será imposible.

Segunda parte del Ingenioso Caballero

Don Quijote de la Mancha

Miguel de Cervantes

–Buena está – dijo Sancho –; fírmela vuestra merced.

–No es menester firmarla – dijo Don Quijote–,

sino solamente poner mi rúbrica.

Primera parte del Ingenioso Caballero

Don Quijote de la Mancha

Miguel de Cervantes