1-ELECCIÓN DE DATASETS

A. Dataset para Traducción Automática: TED2020

-Origen: El dataset proviene del repositorio abierto OPUS (opus.nlpl.eu), una de las mayores colecciones de texto traducido de la web.

-Cantidad de datos: El conjunto contiene un total de 416.846 pares de frases alineadas inglés-español. Tras la división, utilizamos 330.308 registros para entrenamiento y 82.577 para validación/test

-Tipología: Se trata de transcripciones de TED Talks. A diferencia de los textos extraídos de redes sociales o subtítulos de películas, este texto se caracteriza por un registro formal, culto y bien estructurado. Los temas son diversos (tecnología, ciencia, psicología, deporte,cultura), lo que enriquece el vocabulario del modelo.

-Justificación de la elección: He elegido TED2020 principalmente por la calidad de su alineamiento. Al ser transcripciones profesionales, las frases en inglés y español suelen corresponderse casi perfectamente, lo que facilita el aprendizaje del modelo Encoder-Decoder. Además, su tamaño es ideal: suficientemente grande para ser representativo, pero manejable dentro de las limitaciones de memoria de el Colab.

B. Dataset para Detección de Entidades (NER): CoNLL-2003

-Origen: He utilizado la versión estándar del dataset CoNLL-2003 para la tarea de reconocimiento de entidades en inglés, descargada desde el repositorio de referencia synalp/NER.

-Cantidad de datos: El corpus está pre-dividido en tres conjuntos: Entrenamiento (eng.train con aprox. 14k frases), Validación (eng.testa con aprox. 3k frases) y Prueba (eng.testb con aprox. 3k frases).

-Tipología: El texto proviene de noticias de la agencia Reuters. Está anotado manualmente con cuatro tipos de entidades: PER (Personas), ORG (Organizaciones), LOC (Localizaciones) y MISC (Miscelánea).

-Justificación de la elección: He elegido el dataset CoNLL-2003 porque es el estándar de referencia en tareas de NER. Es un conjunto de datos muy utilizado en investigación, con un etiquetado fiable y consistente, lo que lo hace perfecto para entrenar modelos supervisados como los de *spaCy*. Además, trabaja con noticias reales, por lo que incluye entidades conocidas (personas, países, organizaciones…). Esto encaja muy bien con la segunda parte de la práctica, donde necesitamos hacer enlazado de entidades con Wikidata, ya que muchas de estas entidades aparecen registradas allí.

2-Problemas encontrados:

A. Dataset para Traducción Automática: TED2020

Mientras trabajaba con el dataset TED2020 me encontré con varios obstáculos. Aunque el texto estaba bastante limpio, había muchísima variación en la longitud de las frases: algunas muy cortas y otras larguísimas, lo que complicaba bastante el entrenamiento.

El mayor problema, sin embargo, no fue el dataset sino Colab. Al entrenar el modelo, la GPU no aguantaba las secuencias largas y la memoria se saturaba continuamente. También detecté algo de ruido tipográfico, como signos de puntuación o mayúsculas que, aunque parecen inofensivos, hacen que el vocabulario crezca sin aportar realmente información útil.

Limpieza y preprocesamiento realizados:

Para poder trabajar con el dataset de forma más eficiente, apliqué varios pasos de limpieza. Primero, normalicé el texto pasándolo todo a minúsculas, para evitar duplicar tokens como “Hello” y “hello”.

Después, eliminé la puntuación y otros caracteres innecesarios con expresiones regulares, dejando un texto lo más limpio posible.

Una vez preparado, tokenicé todo y apliqué padding para que las secuencias tuvieran la misma longitud.

Dado que las limitaciones de hardware no me dejaban avanzar, tuve que recurrir a un truncamiento bastante agresivo. Tras analizar los histogramas y ver que el 95% de las frases tenían menos de 40 palabras, me vi obligado a fijar un MAX_LENGTH de solo 4 tokens y reducir las UNITS del modelo a 200.

Esto permitió que el modelo entrenara sin explotar la memoria… pero obviamente a costa de perder casi todo el contexto semántico real de las frases.

B. Dataset para Detección de Entidades (NER): CoNLL-2003

Problemas encontrados:

Al trabajar con CoNLL-2003 me encontré con varios inconvenientes. El primero fue el formato original: los datos vienen en columnas, como es típico en CoNLL, pero ese formato no es compatible directamente con spaCy v3, que necesita archivos binarios .spacy.

Otro problema fue la sobrescritura de archivos. La herramienta spacy convert generaba siempre un archivo llamado eng.spacy, sin importar si estaba convirtiendo el train, dev o test. Eso hacía que cada conversión borrara la anterior.

Limpieza y preprocesamiento realizados:

Para solucionarlo, primero preparé un script que convertía los .txt del dataset al formato .spacy usando el conversor de NER de spaCy.

Además, automatizé el renombrado de los archivos convertidos para que no se machacaran entre sí: por ejemplo, eng.train pasaba a train.spacy, eng.testa a dev.spacy, etc. Por último, generé un config.cfg adaptado a mi entorno, ajustando hiperparámetros como la tasa de aprendizaje y el optimizador para asegurar que el modelo funcionara bien utilizando la GPU.

2-TRADUCCIÓN AUTOMÁTICA

1-La elección de los parámetros del modelo Encoder–Decoder estuvo fuertemente condicionada tanto por las características del corpus TED como por las limitaciones computacionales de Google Colab. En primer lugar, en lo referente a la longitud máxima de las secuencias, aunque un análisis previo del corpus indica que muchas frases contienen entre 15 y 20 palabras, en la práctica no fue posible trabajar con secuencias tan largas. Por este motivo, se comenzó con un valor de 8 tokens y posteriormente se realizaron pruebas con 12 tokens. Esta reducción fue una decisión puramente técnica, ya que secuencias más extensas provocaban problemas de memoria en la GPU. Aunque este ajuste supone perder parte del contexto en oraciones complejas, permitió que el entrenamiento fuera estable y pudiera completarse sin interrupciones.

En cuanto a las unidades de memoria, que representan la capacidad del modelo para retener información contextual en su estado oculto, se fijó un valor de 512 unidades. Esta elección se realizó con el objetivo de dotar al modelo de suficiente capacidad para capturar relaciones semánticas complejas, superando configuraciones más sencillas de 128 o 256 unidades. Dado que la GPU disponible lo permitía, este valor ofrecía un buen compromiso entre expresividad del modelo y coste computacional. Respecto a la dimensión de los embeddings, se optó por dos enfoques distintos: en el modelo entrenado desde cero se utilizó una dimensión de 200, un valor común en este tipo de tareas que permite aprender representaciones densas sin incrementar excesivamente el número de parámetros; en cambio, al emplear embeddings preentrenados, la dimensión quedó determinada por los vectores GloVe utilizados, que cuentan con 300 dimensiones.

2-El impacto del tipo de embedding empleado fue especialmente relevante en los resultados obtenidos. Cuando se utilizaron embeddings personalizados, el modelo partía de pesos completamente aleatorios, lo que implicaba aprender al mismo tiempo el significado de las palabras y la tarea de traducción. Esto se tradujo en un proceso de aprendizaje más lento y en un rendimiento inicial bajo, con precisiones reducidas y oraciones generadas poco coherentes desde el punto de vista gramatical. En cambio, al incorporar embeddings preentrenados GloVe y mantenerlos congelados durante el entrenamiento, el modelo se benefició de conocimiento semántico previo sobre las relaciones entre palabras. Gracias a ello, la convergencia fue mucho más rápida y el modelo pudo centrarse principalmente en aprender la correspondencia sintáctica entre ambos idiomas, lo que se reflejó en mejores métricas de precisión y traducciones con un mayor sentido semántico.

3-DETECCIÓN DE NER Y NEL

1-El entrenamiento del modelo de Reconocimiento de Entidades Nombradas (NER) ha dado resultados muy positivos. A lo largo del proceso de entrenamiento se observa una convergencia rápida y estable, alcanzando el mejor rendimiento en torno al paso 4000. En ese punto, el modelo logra un F-Score del 91,24 % sobre el conjunto de validación.

Este valor es especialmente relevante teniendo en cuenta la arquitectura utilizada (tok2vec

junto con el componente NER basado en los vectores *en_core_web_lg*). Superar de forma consistente el 90 % de precisión indica que el modelo ha aprendido patrones generales del lenguaje y no se ha limitado a memorizar los datos de entrenamiento, lo que sugiere una buena capacidad de generalización.

Estos resultados deben interpretarse teniendo en cuenta el origen de los datos. El corpus CoNLL-2003 está formado por noticias de la agencia Reuters de los años 90, lo que introduce un claro sesgo de dominio en el modelo entrenado.

Por un lado, este sesgo resulta beneficioso en contextos similares al corpus original. El modelo muestra un excelente rendimiento en textos periodísticos, especialmente al identificar personas, organizaciones y localizaciones relacionadas con ámbitos políticos y económicos. Esto explica en gran medida los buenos resultados obtenidos durante la evaluación.

Por otro lado, este mismo sesgo supone una limitación importante. Si el modelo se aplicara a textos de otros dominios, como artículos médicos, publicaciones científicas o lenguaje informal de redes sociales, su rendimiento disminuiría considerablemente. La razón es que estos estilos de escritura y su terminología no están representados en el corpus de entrenamiento, lo que reduce la capacidad del modelo para reconocer correctamente las entidades.

E	#	LOSS TOK2VEC	LOSS NER	ENTS_F	ENTS_P	ENTS_R	SCORE
0	0	0.00	128.50	0.49	0.82	0.35	0.00
0	200	830.00	9397.02	66.15	67.44	64.91	0.66
0	400	1582.68	6002.51	74.60	75.57	73.66	0.75
0	600	109.43	2692.11	76.96	77.93	76.02	0.77
0	800	75.62	1888.92	83.83	85.21	82.50	0.84
1	1000	104.24	1848.67	85.50	86.71	84.33	0.86
1	1200	118.58	1294.51	83.35	83.05	83.64	0.83
1	1400	127.98	1469.36	86.05	85.83	86.27	0.86
1	1600	279.07	1667.71	85.89	87.53	84.32	0.86
2	1800	148.82	1469.87	87.64	87.49	87.80	0.88
2	2000	172.75	1091.71	87.46	88.02	86.91	0.87
2	2200	291.67	1465.29	88.02	88.36	87.68	0.88
3	2400	271.04	1385.08	89.26	89.80	88.72	0.89
3	2600	309.77	1298.22	89.83	90.25	89.41	0.90
4	2800	285.98	1191.90	89.65	89.98	89.31	0.90
4	3000	455.57	1520.97	90.03	89.93	90.14	0.90
5	3200	576.19	1397.69	90.03	90.49	89.57	0.90
6	3400	2400.15	1286.32	89.91	90.00	89.82	0.90
7	3600	427.48	917.76	90.46	90.43	90.49	0.90
8	3800	469.36	911.70	90.38	90.05	90.73	0.90
9	4000	447.63	880.80	91.24	91.05	91.43	0.91
9	4200	519.68	701.99	90.02	90.06	89.97	0.90
10	4400	375.67	564.29	90.89	90.88	90.90	0.91
11	4600	658.21	640.14	90.70	91.09	90.32	0.91
12	4800	694.11	656.83	90.59	90.65	90.53	0.91
13	5000	731.83	600.31	90.92	91.51	90.34	0.91
13	5200	490.01	433.87	90.76	90.69	90.83	0.91
14	5400	602.53	442.90	90.09	90.35	89.84	0.90
15	5600	774.08	490.73	90.45	90.57	90.34	0.90

2-La integración del sistema de Enlace de Entidades (NEL) mediante la API de Wikidata permite ampliar las capacidades del modelo más allá del simple reconocimiento. Gracias a este enfoque, las entidades detectadas pueden vincularse a un identificador único dentro de una base de conocimiento.

En las pruebas realizadas con un texto relacionado con el Premio Nobel, el sistema ha demostrado un funcionamiento correcto al enlazar distintos tipos de entidades, incluyendo personas, lugares y organizaciones. Casos como la identificación de László Krasznahorkai, la ciudad de Barcelona o editoriales específicas confirman que el sistema es capaz de conectar el texto con conocimiento estructurado de forma fiable, siempre que la entidad exista en Wikidata.

El principal reto detectado en esta fase es la ambigüedad semántica. La implementación actual realiza una búsqueda directa en Wikidata y selecciona el primer resultado disponible, asumiendo que este es el más adecuado.

Este enfoque funciona en muchos casos sencillos, pero puede fallar en situaciones ambiguas. Un ejemplo claro es el término “Barcelona”, que puede referirse tanto a una ciudad como a un club de fútbol. Aunque en los experimentos realizados el sistema ha seleccionado la entidad correcta, en un contexto deportivo esta decisión podría ser incorrecta, lo que afectaría a la calidad global del sistema.

Para resolver este problema en aplicaciones reales, sería necesario incorporar mecanismos de desambiguación de entidades. Estos métodos tendrían en cuenta el contexto en el que aparece la entidad, analizando las palabras que la rodean y comparándolas con la información asociada a los distintos candidatos en Wikidata. De esta forma, el sistema podría seleccionar la entidad que mejor se ajuste al significado del texto, mejorando así la precisión del enlace.