UNIVERSITAT POLITÈCNICA DE VALÈNCIA

MIARFID Official Master's Degree in Artificial Intelligence, Pattern Recognition and Digital Imaging

Master Universitario en Inteligencia Artificial, Reconocimiento de Formas e Imagen Digital
Universidad Politécnica de Valencia

Transcripción Automática de Fechas con Whisper

TRABAJO TECNOLOGÍAS DEL LENGUAJE HUMANO

Máster Universitario en Inteligencia Artificial, Reconocimiento de Formas e Imagen Digital

Gómez Corral, Miquel

CAPÍTULO 1
Introducción

Este trabajo aborda el uso de modelos basados en transformers para la transcripción automática de fechas en formato textual. El modelo base a partir del cual se han construido las arquitecturas es Whisper, un modelo de reconocimiento automático del habla.

Todo el código relacionado con la implementación del modelo, ha sido provisto como material de la asignatura. El trabajo se ha basado en hacer uso de este a partir de modificaciones a nivel de preprocesamiento, tokenización, entrenamiento y evaluación.

Hardware utilizado

Los experimentos se han realizado en mi máquina personal: GPU NVIDIA RTX 5070 (12 GB VRAM), procesador AMD Ryzen 9 9000X, 64 GB de RAM y Ubuntu 24.04 LTS.

Todos los modelos han podido ser entrenados y evaluados sin problemas.

1.1 Cambios principales

Para facilitar el uso del código base, se han realizado una serie de modificaciones.

- En los datasets en inglés, se ha encontrado la que frase `'tuesday thank you.'` aparece siempre con el punto al final. Por eliminar redundancias y mejorar la calidad de los datos, se ha decidido eliminar todos los signos de puntuación en el preprocesamiento.
- Se han 'parametrizado' los principales elementos del código como las rutas, los tokens especiales (PAD, SOS, EOS) o algunos parámetros para facilitar cambios entre los ejercicios

1.2 Notebooks entregados

Se han entregados 4 notebooks. A continuación la lista de los nombres con una breve descripción del contenido de cada uno.

- `ejercicio-1-es.ipynb`: Ejercicio 1 en español.
- `ejercicio-1-en.ipynb`: Ejercicio 1 en inglés.
- `ejercicio-2.ipynb`: Ejercicio 2 (multilingüe con instrucciones).
- `ejercicio-2-extra.ipynb`: Ejercicio 2 con implementación Top-K y Top-P en inferencia.
- `ejercicio-3.ipynb`: Ejercicio 3 (llamada a funciones).

CAPÍTULO 2
Ejercicios

2.1 Ejercicio 1: Modelos monolingües

En este ejercicio se ha creado la estructura principal de los Notebooks seguida por el resto de ejercicios.

Como se ha mencionado, se han parametrizado las principales variables del código. Se ha detectado el problema con los signos de puntuación en inglés y se ha estructurado todo el procesamiento como se indica en el ejercicio.

A la hora de entrenar los modelos, parecía que el modelo en español necesitamos algunos epochs más para entrenar. Tras 15 epochs para el modelo en español y 10 para el modelo en inglés, se han obtenido los siguientes resultados:

Idioma	**WER %**
Español	1.36
Inglés	1.21

Tabla 2.1: Resultados en transcripción automática de fechas monolingüe: WER Inglés y Español.

2.2 Ejercicio 2: Modelo multilingüe

Partiendo del código del ejercicio 1, se ha implementado un modelo multilingüe que recibe instrucciones para identificar el idioma de la entrada. De nuevo, se han limpiado los signos de puntuación en inglés, pero ahora, se han juntado los datasets de ambos idiomas para crear un dataset combinado con todas las combinaciones: Transcripción en español, Transcripción en inglés, Traducción del Inglés al Español y Traducción del Español al Inglés.

Se han añadido los tokens especiales para identificar el idioma de la entrada y salida, y se ha hecho que el dataset de entrenamiento ponga antes de cada frase el token correspondiente a la instrucción. Luego, para la generación, se añade el token de instrucción antes de entrar al bucle, justo después del token SOS.

Con esto, se han obtenido los siguientes resultados tras 10 epochs de entrenamiento:

Idioma	**WER %**
Bilingüe inglés y español	4.20

Tabla 2.2: Resultados en transcripción automática de fechas monolingüe: WER Inglés y Español.

Vemos que el rendimiento empeora de forma significative respecto a los modelos monolingües. Esto es algo esperado, ya que el modleo tiene que aprender a hacer 4 tareas diferentes (aun que parecidas) en lugar de una. Aún así el resultado es muy bueno.

2.3 Ejercicio 3: Llamadas a funciones

En este ejercicio de nuevo, parte del notebook creado en el ejercicio 1, y se ha hecho el mismo preprocesamiento que en los ejercicios anteriores.

Ahora, el tokenizador tiene más entrada, ya que tiene que manejar las palabras y las llamadas a funciones. Ya que la cantidad de opciones para llamar a funciones no era muy grande, se ha decidido que cada llamada a función, con su parámetro, sea un token único. Esto nos deja con tokens como:

next_day:

- "next_day('friday')"
- "next_day('monday')"
- "next_day('thursday')"
- "next_day('tuesday')"
- "next_day('wednesday')"

relative_day:

- relative_day(+1)"
- relative_day(+2)"
- relative_day(+3)"

Luego, se ha hecho que el modelo aprenda a dar como salida el texto y estos tokens. Con el modelo entrenado, se han procesado sus predicciones para ejecutar las llamadas a funciones y obtener la fecha final.

Los resultados son muy buenos, equivocándose unicamente en 10 de 1000 ejemplos. or lo tanto, el error medio es de un 1.0 %.

idioma	Error %
Bilingüe Funciones	1.00

Tabla 2.3: Resultados en transcripción automática de fechas monolingüe: WER Inglés y Español.

2.4 Ejercicio Extra: Top-K y Top-P

Como ejercicios extra, se ha decidido hacer los dos primeros: implemntar Top-K y Top-P en la generación, y hacer un modelo multilingüe con instrucciones.

Se ha partido del ejercicios dos para hacer estas modificaciones, ya que era el que más margen de mejora presentaba. Se ha vuelto a entrenar el modelo durante 10 epochs para tener una línea base con decodificación greedy, y se ha conseguido un WER de 3.87 %, ligeramente mejor al original.

La implementación estas estrategias de decodificación, pasa por cambiar el código del generate.

1. **Top-K:** Se parte de los logits, se ordenan de mayor a menor y se mantienen los K primeros. El resto de logits se descartan y se hace una softmax con los K logits restantes para obtener las probabilidades. Finalmente, se hace el muestreo a partir de estas probabilidades.
2. **Top-P:** Se parte de los logits, se aplica softmax para obtener las probabilidades, se ordenan de mayor a menor y se calcula la suma acumulada. Se mantienen los tokens hasta que la suma acumulada supere el umbral P. El resto de logits se ponen descartan y se hace una softmax y muestreo como en el caso anterior.

Para ver el impacto de estas técnicas, se ha probado a decodificar el test con diferentes valores de K y P. La línea base con decodificación greedy obtuvo un WER de 3.87 %. Los resultados con Top-K y Top-P son los siguientes:

K	WER %
1	3.87
2	4.53
3	4.23
4	4.44
5	4.60
6	4.72
7	4.39
8	4.79
9	4.23
10	4.72

Tabla 2.4: Resultados con Top-K.

P (%)	WER %
10.0	3.87
20.0	3.87
30.0	4.51
40.0	10.51
50.0	34.32
60.0	540.25
70.0	589.45
80.0	576.49
90.0	538.15
100.0	4.86

Tabla 2.5: Resultados con Top-P.

Cómo vemos. la estrategia ganadora sigue siendo la greedy. El Top-k=1 y top-p con poca masa de probabilidad se comportan de la misma forma que el greedy porque al menos ha de elegir un token, pero luego al aumentar la massa se vuelve un método nefasto. El top-k muestra más consistencia a lo largo de las ejecuciones, aun que no mejora al método greedy.

La razón por la que esto puede ser así, es el simple hecho de que no hay tantos tokens como para que estrategias de este estilo valgan la pena. El modelo no tiende a confundir tokens (darles logits similares), por lo que usando la estrategia por defecto se consiguen los mejores resultados.

2.5 Resumen resultados

A continuación se muestran los resultados WER obtenidos en los diferentes ejercicios realizados:

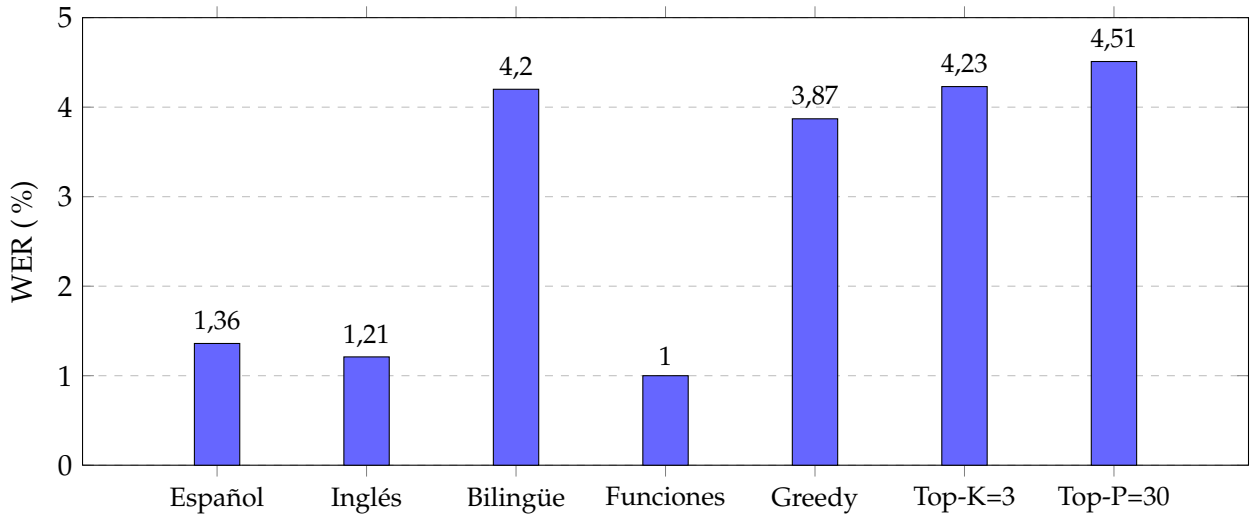

Figura 2.1: Comparación de WER obtenido en cada ejercicio. Español e Inglés corresponden al Ejercicio 1, Bilingüe al Ejercicio 2, Funciones al Ejercicio 3, y Greedy Extra al ejercicio extra con decodificación greedy.