Part Of Speech (POS) tagging utilizando Transformers

Hugo Albert Bonet
Lingüística Computacional
Máster Universitario en Inteligencia Artificial, Reconocimiento de Formas e Imagen Digital.
Universitat Politècnica de València, España
halbbon@etsinf.upv.es

***Resumen*—En este trabajo se desarrolla el entrenamiento de varios modelos basados en la arquitectura Transformers para abordar el problema de *Part Of Speech tagging* en un corpus en español reducido a 66 etiquetas distintas. En concreto, se comparan MarIA y BETO con los modelos clásicos de TnT y HMM. Adicionalmente, se evalúa el interés de realizar fine-tuning de LLMs, en concreto LLaMa en su segunda versión, para esta misma tarea.**

***Palabras clave*—Procesamiento de Lenguaje Natural, Part Of Speech, Transformers, BETO, MarIA, RoBERTa, BERT, Modelos Ocultos de Markov, Trigramas**

I. Descripción del problema

El problema de *Part Of Speech (POS) tagging* consiste en el etiquetado morfosintáctico de secuencias de palabras. Dada una frase se pretende averiguar cuál es la secuencia de categorías gramaticales que con mayor probabilidad se asocia a esta secuencia.

El problema de *POS tagging* se puede afrontar desde varios frentes desde el punto de vista del aprendizaje automático. Tradicionalmente se han empleado Modelos Ocultos de Markov (HMM) o modelos de N-gramas [1] para la predicción de la probabilidad del siguiente token (o etiqueta). Sin embargo, el estado del arte actual se basa en el aprendizaje profundo, especialmente mediante la arquitectura de redes neuronales llamada Transformers [2].

En trabajos previos se ha evaluado la actuación de los modelos de HMM y Trigrams'n'Tags (TnT) frente al corpus *cess-esp* del toolkit NLTK, con una reducción a 66 etiquetas. En este trabajo, se evaluarán modelos de Transformers ante el mismo corpus para compararlos con los resultados previos obtenidos.

Para obtener una comparación de resultados fiable, se realizará el mismo tratamiento previo de los datos. Este preproceso consiste en transformar la anotación de las etiquetas originales (289 etiquetas) a un conjunto reducido (66 etiquetas). Para para conseguirlo se siguieron los siguientes criterios: todas las etiquetas serán de longitud igual a 2 por defecto, salvo los verbos (v) y los signos de puntuación (F) que pueden ser de tres. También pueden existir etiquetas de longitud unitaria. En el conjunto transformado también se deben eliminar anotaciones de la forma: (u'*0*', u'sn').

II. Proceso de entrenamiento

Para realizar el entrenamiento de los Transformers se ha empleado el entorno de entrenamiento proporcionado por la librería de HuggingFace. Este entorno proporciona una interfaz que facilita el entrenamiento de Transformers, conexión con un repositorio en línea, e integración con librerías comunes para el entrenamiento de redes neuronales como Tensorflow o PyTorch.

Los experimentos de este trabajo incluyen tanto el entrenamiento mediante la abstracción de la librería *transformers* de HuggingFace como su integración con PyTorch para los modelos de MarIA y BETO, adaptaciones al castellano de RoBERTa y BERT respectivamente.

MarIA, o *roberta-large-bne* [3], es un modelo basado en Transformers, concretamente en RoBERTa [4], entrenado para reemplazar la palabra enmascarada en un corpus en español de más de 570 GB extraído de páginas web por la Biblioteca Nacional de España entre 2009 y 2019. Esto permite obtener embeddings contextuales de cada una de las palabras de la frase para comprender qué etiqueta corresponde a cada una de ellas.

Además, se han desarrollado dos experimentos adicionales. El primero consiste en combinar los resultados de MarIA utilizando el optimizador AdamW y el nuevo optimizador AdEMAMix [5], desarrollado en 2024 por Apple. El segundo trata de realizar un proceso de fine-tuning al Large-Language-Model (LLM) *llama-2-7b-bnb-4bit* [6] mediante la librería de Unsloth, que permite reentrenar LLMs de una forma menos demandante mediante procesos de cuantización, adaptaciones de bajo rango (LoRA) [7], escalado (RoPE Scaling) [8]...

Para la evaluación de los modelos se emplearán las medidas de Accuracy, para la comparación con trabajos previos, y F1-score debido a la distribución desbalanceada de las etiquetas, mostradas en la Figura 1.

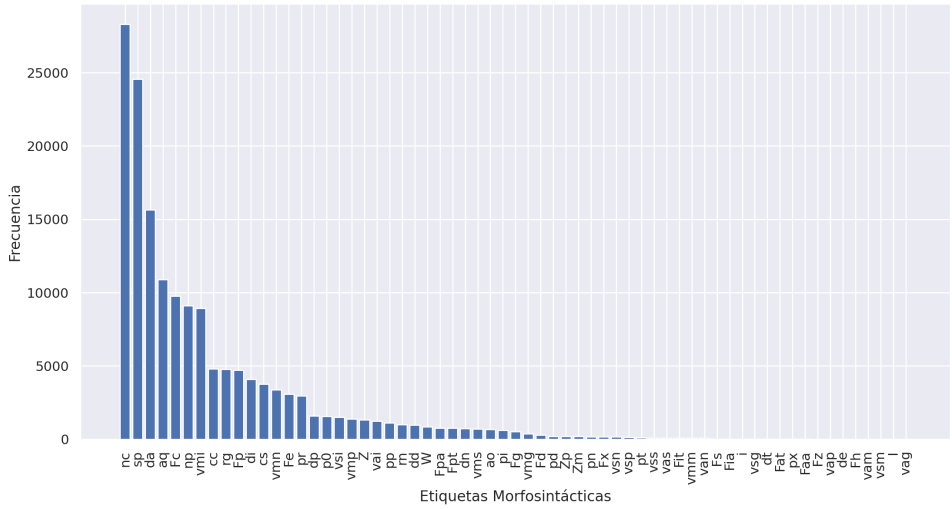

Fig. 1: Distribución de frecuencias de las etiquetas

A. *MarIA*

Como se ha mencionado, MarIA ha sido entrenada en tres condiciones distintas: mediante el Trainer de HuggingFace, mediante un bucle personalizado de PyTorch y empleando el optimizador AdEMAMix. En la Figura 2 se muestra la evolución de las métricas de accuracy y F1-score en las tres épocas de fine-tuning.

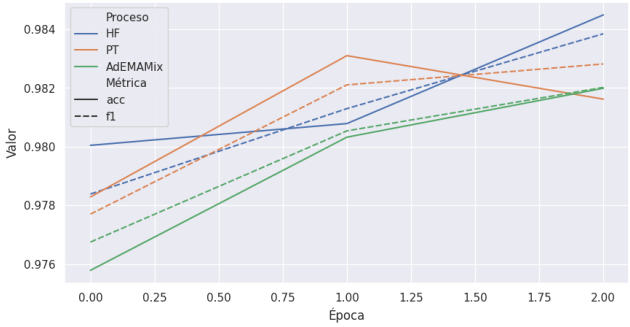

Fig. 2: Evolución de métricas en el entrenamiento de MarIA

Se puede observar cómo las tres configuraciones obtienen resultados por encima del 98% de acierto para ambas métricas, lo que muestra que además de ser efectivo consigue una predicción balanceada. El modelo que peores resultados obtiene a lo largo del aprendizaje es el que emplea el optimizador AdEMAMix, mientras que el mejor finalmente es el que utiliza el Trainer de HuggingFace. El modelo entrenado con PyTorch reduce su accuracy en la última época para conseguir un mejor F1-score.

Los resultados obtenidos en la partición de test se muestran en la Tabla II.

Precisión	Cobertura	Accuracy	F1-Score
0.981	0.981	0.981	0.985

TABLA I: Métricas de MarIA en test

B. *BETO*

Como se ha mencionado, BETO ha sido entrenado en dos condiciones distintas, después de haber observado la ausencia de mejoras tras usar AdEMAMix: mediante el Trainer de HuggingFace y mediante un bucle personalizado de PyTorch. En la Figura 3 se muestra la evolución de las métricas de accuracy y F1-score en las tres épocas de fine-tuning.

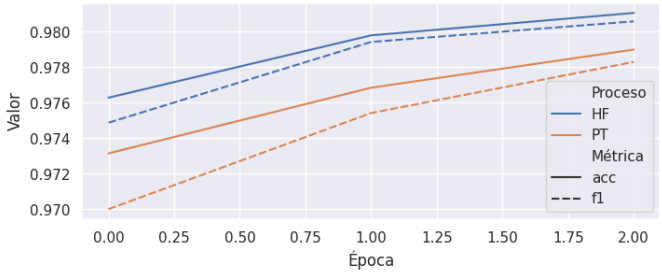

Fig. 3: Evolución de métricas en el entrenamiento de BETO

De nuevo el modelo entrenado con el Trainer de HuggingFace obtiene mejores resultados. Esta configuración obtiene resultados por encima del 98% de acierto para ambas métricas, mientras que el entrenado con PyTorch no alcanza este valor, a pesar de acercarse significativamente. Ambos obtienen valores inferiores para el F1-Score que para la Accuracy a lo largo de todo el entrenamiento.

Los resultados obtenidos en la partición de test se muestran en la Tabla II.

Precisión	Cobertura	Accuracy	F1-Score
0.978	0.976	0.977	0.979

TABLA II: Métricas de BETO en test

C. LLaMa

El modelo *llama-2-7b-bnb-4bit* ha sido seleccionado como representante open-source de los LLMs para experimentar y evaluar su eficacia en esta tarea. Para entrenarlo, se ha empleado la librería Unsloth, que proporciona procesos de cuantización, adaptaciones de bajo rango (LoRA), escalado (RoPE Scaling) o aprendizaje por refuerzo. El prompt utilizado en el entrenamiento es el siguiente:

"""Debajo encontrarás una oración en castellano. De la siguiente lista de etiquetas, devuelve cuál es la etiqueta más adecuada para cada palabra en la oración.

Etiquetas: {}

Oración: {}

Respuesta: {}"""

En la Figura 4 se muestra la evolución de la pérdida en las 60 iteraciones de fine-tuning.

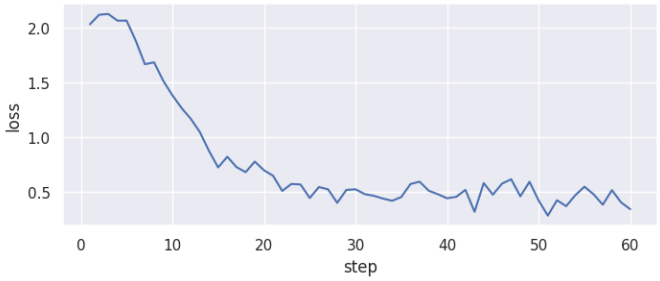

Fig. 4: Evolución de la pérdida en el entrenamiento de LLaMa

Observamos una mejora de la pérdida hasta alrededor de 0.4, donde se estanca. A continuación se muestra un ejemplo de respuesta:

["¡s¿ Debajo encontrarás una oración en castellano. De la siguiente lista de etiquetas, devuelve cuál es la etiqueta más adecuada para cada palabra en la oración.

Etiquetas: ['vsp', 'Zm', 'vai', 'Fpt', 'rg', 'Fia', 'di', 'de', 'Y', 'pi', 'Fz', 'vsm', 'cs', 'pt', 'vmp', 'Fe', 'pr', 'van', 'px', 'vmi', 'Fp', 'vsi', 'vag', 'Fd', 'sp', 'cc', 'Zp', 'dn', 'aq', 'vms', 'pn', 'Faa', 'vam', 'vsg', 'vmm', 'vsn', 'W', 'ao', 'Fat', 'vmn', 'dd', 'Fs', 'vas', 'vap', 'X', 'Fit', 'vss', 'dt', 'I', 'p0', 'dp', 'Fpa', 'np', 'Fg', 'da', 'i', 'pp', 'pd', 'rn', 'Z', 'Fh', 'pe', 'Fc', 'Fx', 'vmg', 'nc']

Oración: Mi compañero Hugo está interesado en tu oferta.

Respuesta: ['rg', 'Fc', 'vmi', 'da', 'nc', 'sp', 'nc', 'Fc', 'vmi', 'da', 'nc', 'sp', 'nc', 'Fp']¡/s¿"]

Observamos que entiende la premisa, se adapta al formato y selecciona etiquetas de la batería de opciones. Sin embargo, no es capaz de adaptar la longitud de la lista resultante al tamaño de la frase.

III. Comparación de resultados

Para comparar resultados con modelos previos, como son HMM y TnT, se empleará la métrica del accuracy debido a que fueron evaluados con esta medida. TnT ha sido evaluado por sí solo y combinado con un etiquetador basado en sufijos de 3 letras como modelo suavizador (proceso *AffixTagger*. La Figura 5 muestra la comparativa de todos los modelos (excepto LLaMa, dado que no produjo una salida que fuese útil para comparar los resultados).

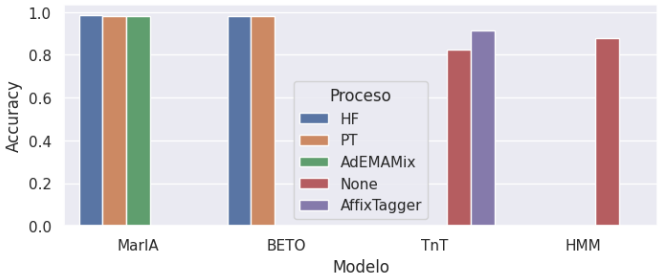

Fig. 5: Comparativa de modelos clásicos y transformers

Se puede observar como los modelos basados en Transformers obtienen resultados superiores a los modelos clásicos con cualquiera de los métodos con los que han sido entrenados, demostrando su superioridad y respaldando su reputación en el campo del Procesamiento de Lenguaje Natural (PLN).

IV. Conclusiones

Se ha podido observar en el transcurso de este trabajo que los modelos basados en Transformers presentan un rendimiento significativamente superior a los modelos clásicos. No obstante, su demanda de recursos es también muy superior, llegando MarIA a necesitar 18 minutos y 23 segundos en una máquina con una GPU T4 para ser entrenada a través de 3 épocas, con una utilización prácticamente constante del 100% de los recursos que ofrece la GPU como se muestra en la Figura 6.

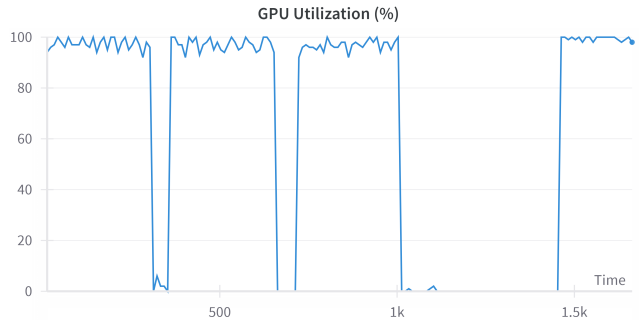

Fig. 6: Porcentaje de utilización de la GPU en el entrenamiento

Por tanto, en caso de no contar con modelos previamente entrenados que sean capaces de adaptarse a la tarea en pocas iteraciones, los modelos clásicos son una mejor solución para un corpus de menores dimensiones.

Además, el uso de LLMs para tareas tan específicas en las que es necesario amoldarse a unas restricciones muy rígidas no ha resultado eficaz. Tal vez modelos como GPT-o1 sean capaces de conseguirlo, pero el entorno open-source no está preparado todavía para ello.

BIBLIOGRAFÍA

[1] Hasan, F. M., UzZaman, N., & Khan, M. (2007). Comparison of different POS Tagging Techniques (N-Gram, HMM and Brill's tagger) for Bangla. In Advances and innovations in systems, computing sciences and software engineering (pp. 121-126). Springer Netherlands.

[2] Vaswani, A. (2017). Attention is all you need. Advances in Neural Information Processing Systems.

[3] https://huggingface.co/PlanTL-GOB-ES/roberta-large-bne

[4] Liu, Y. (2019). Roberta: A robustly optimized bert pretraining approach. arXiv preprint arXiv:1907.11692.

[5] Pagliardini, M., Ablin, P., & Grangier, D. (2024). The ademamix optimizer: Better, faster, older. arXiv preprint arXiv:2409.03137.

[6] Touvron, H., Martin, L., Stone, K., Albert, P., Almahairi, A., Babaei, Y., ... & Scialom, T. (2023). Llama 2: Open foundation and fine-tuned chat models. arXiv preprint arXiv:2307.09288.

[7] Hu, E. J., Shen, Y., Wallis, P., Allen-Zhu, Z., Li, Y., Wang, S., ... & Chen, W. (2021). Lora: Low-rank adaptation of large language models. arXiv preprint arXiv:2106.09685.

[8] Liu, X., Yan, H., Zhang, S., An, C., Qiu, X., & Lin, D. (2023). Scaling laws of rope-based extrapolation. arXiv preprint arXiv:2310.05209.