Instituto Tecnológico y de Estudios Superiores de Monterrey

Campus Querétaro

TC3007C. Inteligencia artificial avanzada para la ciencia de datos II

Módulo 2:

"Implementación de un Modelo de Deep Learning"

Evidencia presentada por el estudiante:

Javier Suarez Duran A01707380

Profesores:

Benjamín Valdés Aguirre
José Antonio Cantoral Ceballos
Carlos Alberto Dorantes Dosamantes
Ismael Solis Moreno
Eduardo Daniel Juárez Pineda

Fecha de entrega:

Domingo 5 de Noviembre de 2023

Resumen — En este reporte se documenta la implementación completa de un modelo de deep learning con el cual poder clasificar sentimientos a través del procesamiento del lenguaje natural. El programa con el modelo utiliza el lenguaje de programación "Python" y puede ser encontrado en el siguiente repositorio de GitHub: https://github.com/Javier1257/TC3007C_DL_Model.git

Palabras clave — *modelo, implementación, desempeño, deep learning, GitHub.*

I. Introducción

El modelo a ser implementado es una red neuronal recurrente cuya función es determinar qué tipo de sentimiento se puede captar en un texto definido, para el cual se está utilizando el conjunto de datos de "Twitter Sentiment Analysis" que se puede encontrar en la página Kaggle; más detalladamente el propósito con este set de datos será que clasifique entre textos que reflejen emociones positivas, negativas e indiferentes de las que no lo son. Además, se pondrá a prueba el desempeño del modelo y con base en los resultados se tomarán las medidas necesarias para tratar de mejorar su eficacia en la solución del problema.

II. Descripción del modelo

Para comenzar en este caso se trata de una red neuronal recurrente; el modelo en total cuenta con cinco capas, una capa de embedding como entrada, no entrenable cargada con los pesos de "glove.6B.100d", una capa de salida y tres capas internas, de las cuales la segunda es una red neuronal recurrente "GRU" (Gated Recurrent Unit) y el resto constituyen el clasificador formado por las tres capas de neuronas restantes; profundizando con el modelo empezamos con la capa de embedding la cual se decidió por definir como no entrenable al suministrarle la información dentro de un archivo de embeddings como es el de "glove.6B.100d" para simplificar esta primera parte del modelo y no tener la necesidad de entrenar la capa desde cero así como aprovechar él amplió diccionario que en este caso el archivo ofrece para el aprendizaje del modelo; continuando la siguiente capa es la "GRU" cuyo dimensión es de 128 elementos y utiliza la tangente hiperbólica (tanh) como función de activación, además de también utilizar un bias y regularización L2 con un valor de 0.00001; luego le sigue una capa de "Dropout" con un valor del 0.3 y finalmente la información llega al clasificador, la primer capa del clasificador se forma por 32 neuronas, un bias y relu como función de activación, luego las siguientes dos capas muestran la misma estructura, 16 neuronas, bias y relu también como función de activación, y por último para terminar el proceso la capa de salida cuenta con 3 neuronas y softmax como función de activación dado que el modelo debe de ser capaz de predecir entre 3 clases distintas. Por consiguiente la función de perdida dentro del modelo es "sparse categorical crossentropy", además que también usa un optimizador Adam con un learning rate de 0.001; en este caso el set de datos se conforma por dos archivos ".csv", uno para entrenamiento y uno para pruebas, el set de entrenamiento se dividió en 20% para un set de validación y el resto para el entrenamiento del modelo, este duro 20 épocas y contó con una función de "early stopping" la cual se centraba en supervisar que la perdida del set de validación mejorar

continuamente o de lo contrario el entrenamiento se detuviera, además de que al hacerlo esta función obtenía los mejores pesos calculados y los definía como los valore definitivos en el modelo; y por último hace falta mencionar que se utilizaron "Batches" de 64.

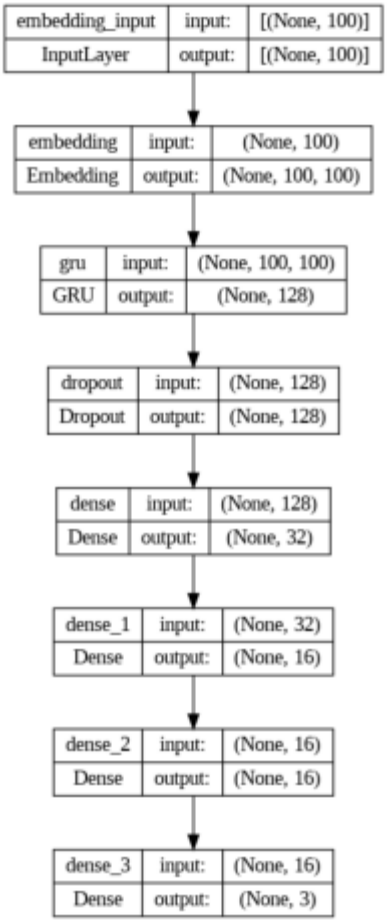
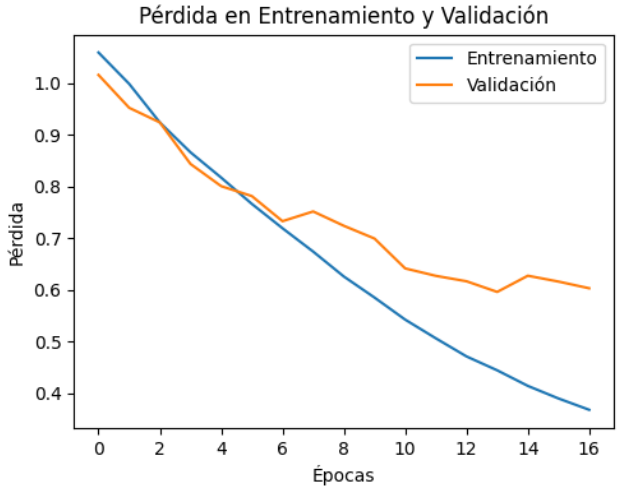


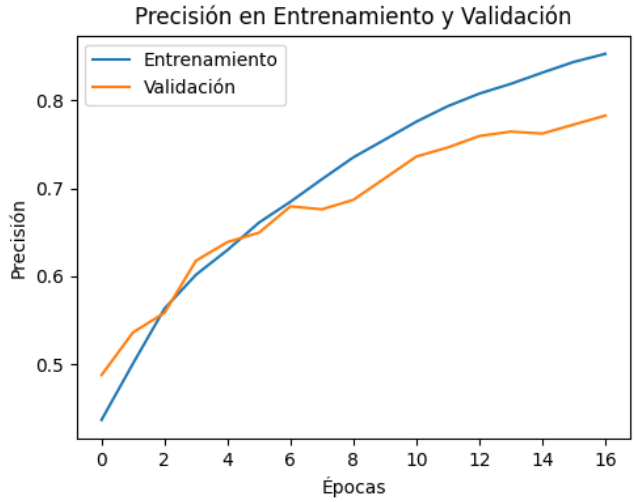
Diagrama del modelo generado por código

III. Análisis del Modelo

Después del Entrenamiento del modelo se obtuvieron las siguientes gráficas con respecto al desempeño del modelo durante este:



Gráfica de Perdida generada por código



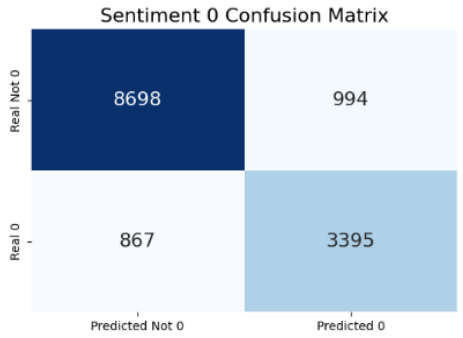
Gráfica de Precisión generada por código

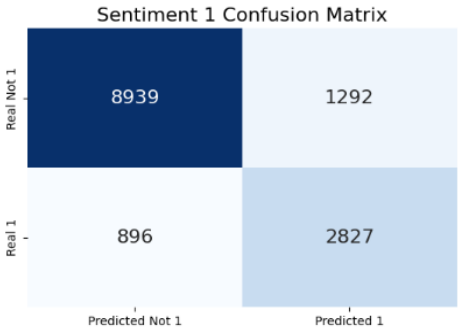
En las imágenes se puede apreciar que el modelo detuvo el entrenamiento en la época 16 y que alcanzo una perdida de 0.36 en el set de entrenamiento y 0.6 en el de validación; además de que la precisión en el set de entrenamiento es de 0.82 y en el set de validación de 0.78; esta información ya indican que el modelo está un poco sesgado por lo que significa que este tiene "Overfitting", puesto en otras palabras, el modelo no generalizo muy bien la información y trato de aprender el set de datos en lugar de buscar y entender los patrones detrás de ellos.

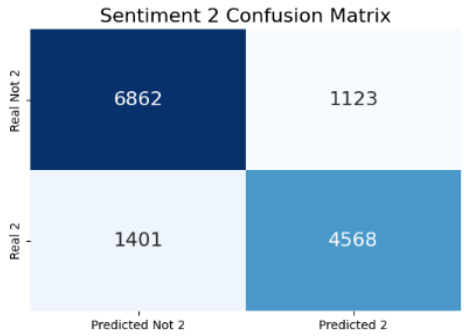
También nos podemos dar cuenta que tiende a hacer más clasificaciones de una clase en particular con la siguiente evidencia:

- Sentiment_0: Emoción Positiva
- Sentiment_1: Emoción Negativa
- Sentiment_2: Emoción Indiferente

Set de validación:

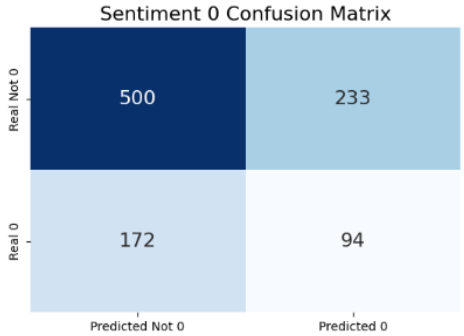


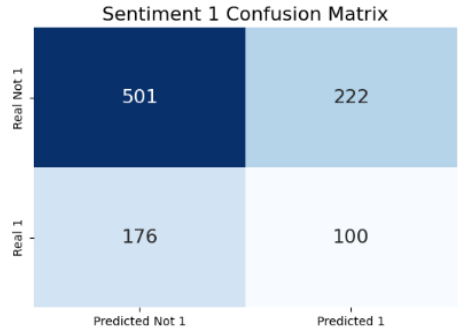


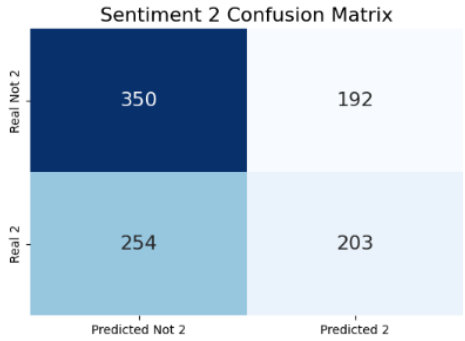


Gráficas generadas por código

Set de pruebas:







Gráficas generadas por código

Las matrices de confusión son utilizadas para visualizar el comportamiento del modelo con respecto a sus predicciones y podemos inferir tanto de los resultados del set de validación como del de pruebas que la clase “Sentiment_2” relacionada con una emoción indiferente es la clase que más se predice. Por último se obtuvieron las siguientes métricas al evaluar el modelo con respecto a cada set de datos como es el de validación y el de pruebas.

Métricas de evaluación del set de validación:

```
Weighted Precision:  0.7627287902491199
Weighted Recall:  0.7732549806507095
Weighted F1 Score:  0.7672577982454449

Macro Precision:  0.7541757462121863
Macro Recall:  0.7737318555231262
Macro F1 Score:  0.7631344506067147

Accuracy:  0.7362046724953418
```

Métricas calculadas en código

Métricas de evaluación del set de pruebas:

```
Weighted Precision:  0.39743984850631386
Weighted Recall:  0.3973973973973974
Weighted F1 Score:  0.3948053208636382

Macro Precision:  0.37064827684813234
Macro Recall:  0.3866345373788704
Macro F1 Score:  0.37600200753451213

Accuracy:  0.3433433433433433
```

Métricas calculadas en código

En este caso las hubo dos tipos de evaluaciones, las que llevan la etiqueta “Weighted” se enfocan en medir el rendimiento del modelo con base en las clases clasificadas sin importar si están equilibradas o no, y las que llevan la etiqueta “Macro” castigan este desequilibrio midiendo que tan bien el modelo es capaz de predecir las diferentes clases sin tomar en cuenta cuál predomine en el set de datos. Y aquí se vuelve más visible lo antes mencionado y es que el modelo no logra captar del todo los patrones detrás de la información, por consiguiente no logra realizar buenas predicciones al obtener puntajes tan bajos en las métricas; es claro que el set de validación utilizado durante el entrenamiento llega a ser clasificado decentemente con esa media del 75% de aciertos en todas las métricas, pero el rendimiento obtenido en el set de pruebas es deficiente, por debajo del 40% en todas las métricas calculadas, dando a entender que el modelo no está funcionando adecuadamente, por lo tanto, necesita de un buen refinamiento para poder clasificar adecuadamente las emociones en los textos probados.

IV. Refinamiento

Durante este proceso se probaron múltiples arquitecturas, además de eliminar el set de validación y balancear las clases dentro del mismo para que todas y cada una de ellas contengan el mismo número de instancias , como resultado de estas implementaciones el modelo llegó a los siguientes resultados.

Métricas de evaluación del set de entrenamiento:

```
Weighted Precision:  0.9789002650759488
Weighted Recall:  0.9743790713066499
Weighted F1 Score:  0.9764883576901295

Macro Precision:  0.9789002650759487
Macro Recall:  0.9743790713066499
Macro F1 Score:  0.9764883576901294

Accuracy:  0.9743790713066499
```

Métricas calculadas en código

Métricas de evaluación del set de pruebas:

```
Weighted Precision:  0.38009005852454647
Weighted Recall:  0.33633633633633636
Weighted F1 Score:  0.34194039209168736

Macro Precision:  0.3561719411859177
Macro Recall:  0.34632817336526234
Macro F1 Score:  0.3362050559196284

Accuracy:  0.33633633633633636
```

Métricas calculadas en código

Lo que nos indica que el modelo sigue sin poder terminar de comprender adecuadamente los datos y clasificar satisfactoriamente los textos; sin embargo, algo positivo en comparación es que se llegaron a prácticamente los mismos resultados con una arquitectura más simple la cual se conforma una capa de embedding como entrada, esta no es entrenable y utiliza "glove.6b.300d" como sus pesos; luego la siguiente capa es una GRU con bias y un tamaño de 128, regularización L2 de 0.0001 y tangente hiperbólica como función de activación; por último la capa de salida se conforma de 3 neuronas simples y softmax como función de activación. El modelo se entrenó con 0.001 de learning rate y optimizado Adam, "sparse categorical crossentropy" como función de pérdida y Batches de 128; todo esto durante 20 épocas en las que se monitoreaba la precisión en pos de detenerlo antes en caso de no haber una mejora significativa entre las épocas de entrenamiento.

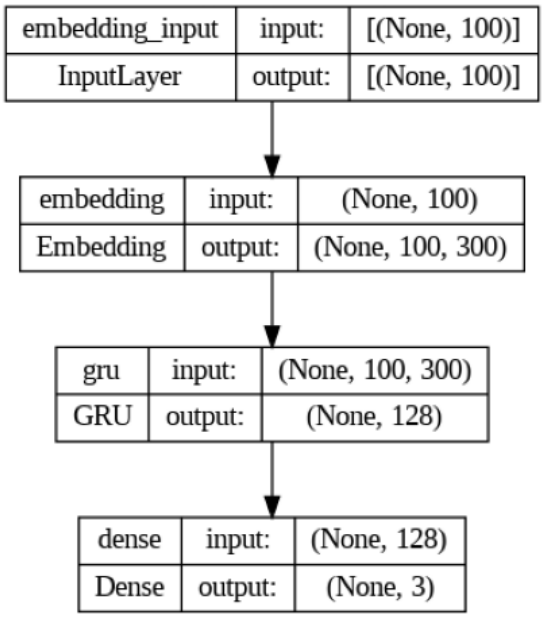


Diagrama del modelo generado por código

V. Conclusión

En conclusión, el modelo no logra resolver satisfactoriamente el problema por lo que se sugiere explorar nuevas técnicas de pre procesado para lograr que el modelo pueda generalizar de una mejor manera los datos utilizados; por otro lado, es positivo conocer que un modelo más simple y menos complejo logra llegar a los mismos resultados que la versión original por lo que durante futuras implementaciones se puede retomar como punto de partida para así lograr un modelo que se ajuste perfectamente al problema sin necesidad de aumentar demasiado la complejidad para obtener un modelo más óptimo de procesar.

VI. Anexos

GitHub del Modelo de Deep Learning Implementado:
https://github.com/Javier1257/TC3007C_DL_Model.git

Set de Datos utilizado para la solución del problema:
https://www.kaggle.com/datasets/jp797498e/twitter-entity-sentiment-analysis/data