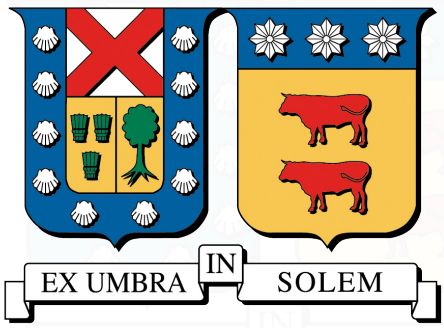
UNIVERSIDAD TÉCNICA FEDERICO SANTA MARÍA
DEPARTAMENTO DE ELECTRÓNICA
VALPARAISO - CHILE



MEJORAR LA PRECISIÓN DE CLASIFICACIÓN EN UN MODELO DE DEEP LEARNING

CRISTIAN MATÍAS NAVARRO GONZÁLEZ

MEMORIA PARA OPTAR AL TÍTULO DE
INGENIERO CIVIL ELECTRÓNICO

PROFESOR GUÍA	:	SR. WERNER CREIXELL F.
PROFESOR CORREFERENTE	:	SR. MAURICIO ARAYA L.

DICIEMBRE 2022

AGRADECIMIENTOS

Quiero dedicar esta sección principalmente a mi padres, Cristian Navarro y Ximena González que durante toda mi vida y periodo educativo es de quienes más he aprendido, ellos me brindaron las herramientas y el apoyo más que necesario en todo momento para salir adelante. Por esto y más, muchas gracias por su entrega, su amor y su apoyo incondicional, los amo mucho!.

A mis hermanos que me apoyaron, visitaron y me dieron el espacio para mis estudios.

A mi tío que me brindó la oportunidad de trabajar con él y adquirir gran conocimiento en mi práctica industrial y así formarme en distintas áreas de una empresa.

Agradezco también a mi pareja y su familia que en este último tiempo también me han apoyado y me han dado el espacio sin compromiso para avanzar en mi trabajo y estudios.

A mis amigos que hicieron de este paso por la universidad una gran aventura. Por todas las tardes y noches de estudio en la universidad, por las horas de juegos y por momentos que no se olvidaran jamás, gracias de todo corazón.

Finalmente, agradezco a mi profesor guía Werner Creixell que con su guía, conocimiento y entusiasmo me motivó en todo momento a crecer y ser un profesional.

Índice de Contenidos

Índice de Tablas

Índice de Figuras

RESUMEN

Recientemente, el desarrollo en el área de la Inteligencia Artificial, específicamente en el área de Deep Learning ha avanzado de manera explosiva, impactando varios campos de la ingeniería muy rápidamente.

En este trabajo de título se utilizara el potencial del Deep Learning para una clasificación multiclase, específicamente una clasificación de texto. El objetivo de este trabajo es tomar nombres de productos y clasificarlos en sus categorías correspondientes para sustituir el proceso de categorización que posee una empresa actualmente. Para esto, entre las posibles alternativas de solución, se escoge la opción de definir y entrenar una red neuronal para realizar la clasificación de productos. Finalmente se utilizarán distintas métricas para medir el desempeño de la red construida y compararla con el modelo que actualmente posee la empresa para la categorización de productos.

ABSTRACT

Recently, the development in the area of Artificial Intelligence, specifically in the area of Deep Learning has advanced explosively, impacting several fields of engineering very quickly.

In this title work, the potential of Deep Learning for a multiclass classification, specifically a text classification, will be used. The objective of this work is to take product names and classify them in their corresponding categories to replace the categorization process that a company currently has. For this, among the possible solution alternatives, the option of defining and training a neural network to carry out the classification of products is chosen. Finally, different metrics will be used to measure the performance of the built network and compare it with the model that the company currently has for the categorization of products.

1 | Introducción

El auge de la inteligencia artificial y el aprendizaje supervisado, ha permitido ser un gran apoyo para empresas que deben realizar actividades de clasificación de información manualmente, ya que este proceso manual genera grandes gastos en tiempo de trabajo, equipos para los trabajadores, sueldo del personal, entre otros. Todos estos gastos son evitables al trabajar con Deep Learning, un caso específico que está realizando este cambio en su trabajo es Almacén Gurú.

Almacén Gurú es una empresa que entre varias funciones, genera informes para proveedores de productos, como lo son, Coca - Cola , Watts, entre otros. En estos informes se incluye información acerca de: productos vendidos, zonas de mayor venta, etc. Para generar estos informes, se extrae la información de las facturas de los proveedores desde el servicio de impuestos internos. Estas facturas contienen los nombres de los productos vendidos, sin embargo estos nombres no siempre son lo suficientemente claros para entender cual fue el producto que se vendió o sus características. Es por esto que existe un equipo de categorización manual encargado de desglosar el nombre de los productos y clasificarlos en distintas características como lo son, categoría, tipo de variedad, envase, entre otros.

Esta empresa actualmente desea automatizar el proceso de clasificación referente a la categoría de los productos, puesto que es la información más utilizada.

En la figura 1 se muestra una porción de la información de los productos presente en la base de datos de la empresa. Se puede ver que la columna “KeyItem” (que corresponde al producto vendido en la factura) es en algunas ocasiones poco clara respecto a las características del producto y sin embargo distintas descripciones pertenecen a una misma categoría.

KeyItem	Categoria	tipo_variedad	mililitros	envase	contenido_gr
16376686-aquarius-limonada-pera	aguas envasadas	aguas saborizadas	500	botella de plastico	
16376686-cachantun-500ml-sin-gas	aguas envasadas	aguas minerales	500	botella de plastico	
16376686-vital-16lts	aguas envasadas	aguas minerales	1600	botella de plastico	
16376686-vital-16lts-con-gas	aguas envasadas	aguas minerales	1600	botella de plastico	
16376686-vital-16lts-con-sin-gas	aguas envasadas	aguas minerales	1600	botella de plastico	
16376686-vital-16lts-sin-gas	aguas envasadas	aguas minerales	1600	botella de plastico	
16376686-vital-600cc	aguas envasadas	aguas minerales	600	botella de plastico	
16376686-vital-600cc-con-gas	aguas envasadas	aguas minerales	600	botella de plastico	
16376686-vital-600cc-con-sin-gas	aguas envasadas	aguas minerales	600	botella de plastico	
16376686-vital-600cc-sin-gas	aguas envasadas	aguas minerales	600	botella de plastico	
17106458-recargas-de-agua-purifica	aguas envasadas	aguas purificadas	otros	bidon recarga	
25710378-aceite-economico	aceites comestibles				
5081274-aloe-vera	aguas envasadas	aguas de aloe vera	500	botella de plastico	
5081274-cachantun	aguas envasadas	aguas minerales	500	botella de plastico	
5081274-cachantun-de-medio	aguas envasadas	aguas minerales	500	botella de plastico	

Figura 1.1: Ejemplo varios nombres para un mismo producto.

Así entonces, en este proyecto de título se aprovechará la capacidad de clasificación multiclase de los modelos de Deep Learning para clasificar el texto referente a los productos en sus distintas categorías. Para esto, se medirá la precisión actual de un modelo de categorización que posee la empresa y se comparará con un modelo propuesto por el estudiante para determinar una mejor solución de clasificación.

2 | Trabajo a desarrollar y resultados esperados

2.1. Etapas del trabajo

El trabajo por desarrollar a lo largo del proyecto de titulación puede descomponerse en los siguientes pasos:

2.1.1. Extracción y Pre – Procesado

El primer paso sera extraer la información de los nombres de los productos y categorías desde la base de datos de la empresa. Luego será necesario transformar los datos a codificación One-Hot para preparar la entrada para el modelo de Deep Learning de la empresa.

2.1.2. Aplicación de modelos existentes

Se aplicará el modelo de Deep Learning presente actualmente en la empresa sobre los datos extraídos y pre procesados en la etapa anterior.

2.1.3. Determinar precisión de clasificación

Una vez se tenga la predicción del modelo de Deep Learning, se aplicarán distintas métricas para estimar el desempeño del algoritmo actual, tales como precisión, matriz de confusión, entre otros.

2.1.4. Proponer un modelo de Deep Learning

Finalmente, se buscará una arquitectura que permita reemplazar el modelo actual de la empresa en busca de una mejor precisión a la hora de clasificar los productos.

2.2. Resultados esperados

Como resultado esperado, se tiene principalmente mejorar la precisión en la clasificación de productos de la empresa. Así sería posible remover en gran medida la supervisión humana del proceso de clasificación, solo sería necesaria su presencia para subir el set de datos a clasificar, mejorando así significativamente en temas de costos y tiempo la clasificación de información.

3 | Estado del Arte

El objetivo de este trabajo de título es mejorar la precisión de clasificación de texto de nombres de productos. Para este proceso, se utilizará principalmente modelos de Deep Learning y distintas métricas para medir la precisión de la red. Es por esto que a continuación se detallan los temas principales a utilizar en este proyecto junto con trabajos de externos que pueden ser útiles para el desarrollo del mismo.

3.1. Redes Neuronales

Una red neuronal es un modelo inspirado en una parte del cerebro humano, específicamente en como se procesa la información. La red neuronal funciona con un número elevado de unidades de procesamiento interconectadas llamadas neuronas o nodos, estos nodos se organizan en capas. En una red neuronal se pueden distinguir a grandes rasgos 3 capas, tal como se observa en la figura 3.1 una capa de entrada, con nodos que representan los campos de entrada, una o varias capas ocultas y una capa de salida con nodos que representan los campos de salida.

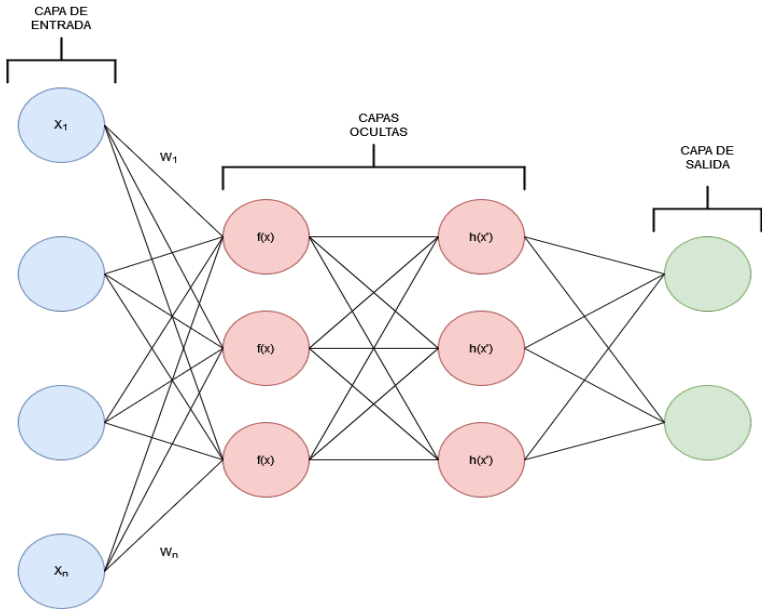


Figura 3.1: Estructura red neuronal.

Cada nodo se conecta a otro y tiene un peso y un umbral asociados. Si la salida de cualquier nodo individual está por encima del valor del umbral especificado, dicho nodo se activa, enviando datos a la siguiente capa de la red. De lo contrario, no se pasan datos a la siguiente capa de la red.

Se puede pensar cada nodo como un modelo de regresión lineal, compuesto de datos de entrada, ponderaciones, un umbral y una salida. Así entonces, la fórmula asociada a cada nodo sería de la siguiente manera:

$$f(x) = b + \sum_i w_i x_i \tag{3.1}$$

$$b = umbral \quad w_i = pesos \quad x_i = entradas \tag{3.2}$$

Podemos interpretar que cada peso w_i, representa la influencia de la entrada en la salida, ya que multiplica a x_i. Mientras el término b (umbral) controla qué tan predispuesto está el nodo a disparar un 1 o un 0 independiente de los pesos. En el caso de regresión f(X) nos da el resultado predicho, a partir de un vector de entrada X. Mientras que en el caso de la clasificación, la clase predicha está dada por:

$$clasificacion = \begin{cases} 1 & Si \quad f(X) > 0 \\ 0 & Si \quad f(X) <= 0 \end{cases} \tag{3.3}$$

Existe un paso adicional para que los nodos se comporten realmente como neuronas, una función de activación. La función de activación utiliza la suma ponderada de la ecuación 3.1 y la transforma una vez más como salida. Existen muchas funciones de activación, pero para entender el proceso de una red neuronal describiremos solamente una en detalle, la función sigmoide, para mayor información sobre otras funciones, revise la información presente en (1). La función sigmoide se define como:

$$\sigma(z) = \frac{1}{1 + e^{-z}} \tag{3.4}$$

$$z = f(x) = b + \sum_i w_i x_i$$

Podemos ver en la figura 3.2 que la función sigmoide actúa como una especie de función "aplastadora", comprimiendo la salida a un rango de 0 a 1.

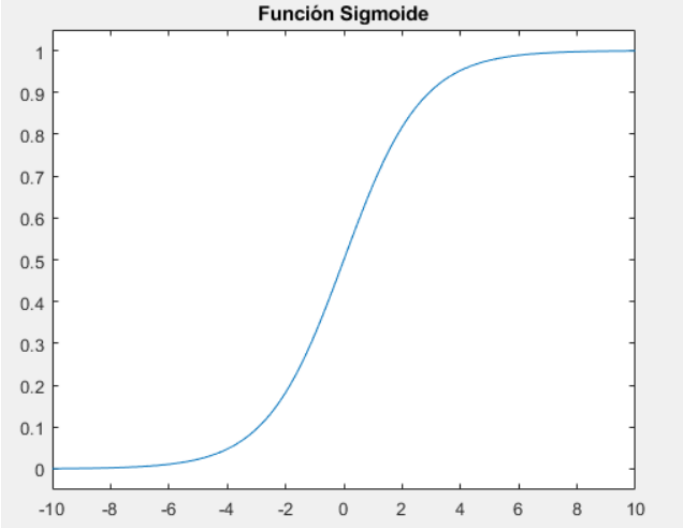


Figura 3.2: Función Sigmoide.

Así entonces la red esta lista para genera predicciones. La red aprende examinando los registros individuales, generando una predicción para cada registro y realizando ajustes a las ponderaciones cuando realiza una predicción incorrecta. Aquí es donde aparece la función de coste.(2)

La función de coste trata de determinar el error entre el valor estimado y el valor real, con el fin de optimizar los parámetros de la red neuronal. Una función de coste común es el error cuadrático medio (MSE), la cual es de la siguiente manera:

$$MSE = \frac{1}{2m}\sum_{i=1}^{m}(y' - y)^2 \tag{3.5}$$

En última instancia, el objetivo es minimizar nuestra función de coste para asegurar la corrección de las ponderaciones y obtener mejores predicciones. A medida que el modelo ajusta sus ponderaciones y umbrales, utiliza la función de coste y el descenso del gradiente para alcanzar el punto de convergencia, o el mínimo local.

Esto quiere decir que queremos encontrar los parámetros w_i que minimicen la función de coste, ya que entre menor sea la diferencia entre valor predicho y valor real, quiere decir que el modelo aproxima mejor la salida . Así entonces, se aplica el método del gradiente para actualizar los pesos de la siguiente forma:

$$w_i' = w_i - k\nabla J(w_i) \tag{3.6}$$

3.2. Embeddings

3.2.1. Word embeddings

Los word embedding es un técnica que representa palabras como vectores de números reales. Dicha representación agrupa palabras que son semánticamente y sintácticamente similares. Por ejemplo esperamos que las palabras “delfín y foca” se encuentren cerca, pero “Paris” y “delfín” no se encuentren cerca ya que no existe una fuerte relación entre ellas. Esto provoca que palabras semánticamente similares, tengan vectores similares.

Para cada par de palabras, se calcula la distancia entre los vectores asociados a cada una. La medida más utilizada para esto es la similitud coseno, la cual da una medida de similitud entre dos vectores (no nulos) con el que se evalúa el valor del coseno del ángulo comprendido entre ellos. Esta función trigonométrica proporciona un valor igual a 1 si el ángulo comprendido es cero, es decir, ambas palabras son idénticas. Si los vectores fuesen ortogonales el coseno se anularía, lo cual significaría que ambas palabras no cuentan con ninguna relación entre ellas. En caso de que ambos vectores apunten en sentidos opuestos el resultado sería -1, ambas palabras tendrían significados contrarios. El valor de esta métrica se encuentra por lo tanto entre -1 y 1. Esto quiere decir, que para calcular la similitud entre dos palabras, utilizamos el ángulo que forman sus vectores como medida. La similitud coseno se expresa de la siguiente manera:

$$cos(x, y) = \frac{x \cdot y}{\|x\| \cdot \|y\|} \tag{3.7}$$

En el Cuadro 3.1 se muestra un ejemplo de varios vectores de palabras, es posible ver que el valor numérico en cada dimensión captura la cercanía de la asociación de la palabra a dicho significado. Su objetivo es cuantificar y categorizar similitudes semánticas entre elementos lingüísticos. (3)

	Categoria1	Categoria 2	Categoria 3
Coca-Cola	0.45	0.23	0.92
Agua	0.54	0.12	0.98
Belmont	0.45	0.73	0.21
Jamón	0.10	0.89	0.25

Tabla 3.1: Ejemplo vectores de palabras

Si bien, el ejemplo presentado posee valores aleatorios, es una buena representación de lo que esta técnica permitiría hacer en este proyecto. Donde con los nombres de los productos y las categorías a la que pertenecen se puede desarrollar el word embedding para obtener los vectores de palabras con el valor numérico que represente la categoría más probable a la que pertenece cierto producto.

3.2.2. Embedding Layer

Una buena definición de la capa de embedding es *"The Embedding layer is best understood as a dictionary that maps integer indices (which stand for specific words) to dense vectors. It takes integers as input, it looks up these integers in an internal dictionary, and it returns the associated vectors..."*(4)

La capa de embedding toma con entrada un tensor 2D de enteros y devuelve un tensor 3D de flotantes. Cuando se inicializa la capa de embedding, tiene pesos aleatorios como cualquier otra capa. Durante el entrenamiento, los vectores de palabras se ajustan gradualmente con el backpropagation estructurando el espacio de embedding en algo que el modelo pueda explotar. Es decir después de entrenar esta capa, se puede decir que el modelo "aprende" la relación entre las palabras y puede predecir mejor el "contexto" al que pertenecen.

Considerando que la capa de embedding devuelve un tensor 3D, es necesario aplicar una capa flatten a su salida para conectar una red que permita clasificar la información. Es por esto que la estructura ideal a desarrollar en este proyecto tendría la siguiente estructura:

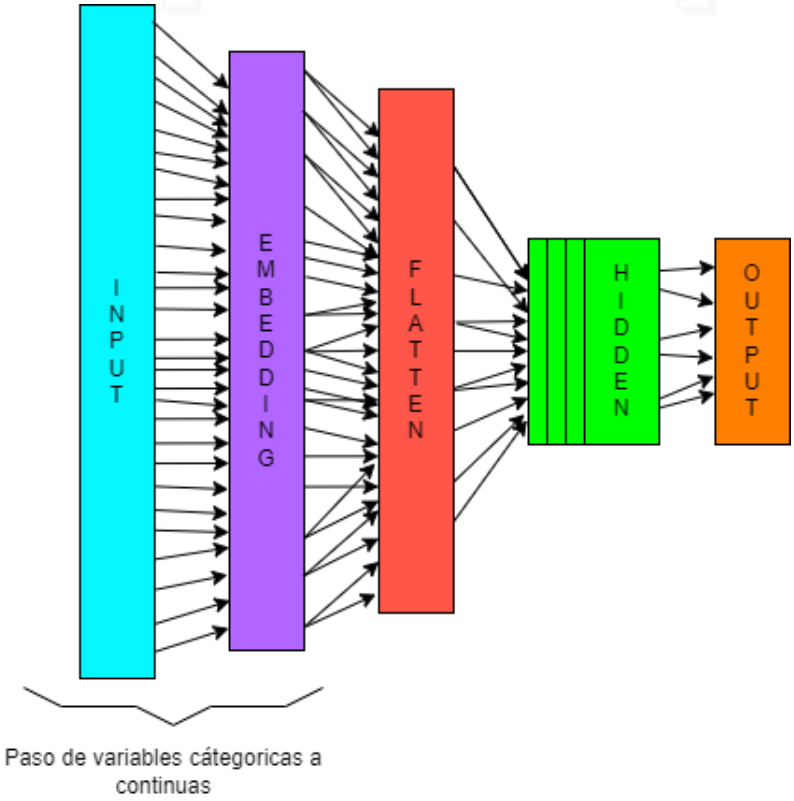


Figura 3.3: Estructura red clasificadora con capa de embedding.

3.3. Métricas de evaluación

Existen diversas métricas para evaluar cualquier modelo de deep learning, a continuación definiré unos conceptos necesarios para comprenderlas y luego se presentarán las métricas mas conocidas (5):

Cuando un valor es predicho por una red neuronal, pertenece a uno de los siguientes casos:

- Verdadero positivo (TP): Cuando la clase predicha y la real son la misma.
- Verdadero negativo(TN): Cuando la clase predicha y la real son la misma y no son esta.
- Falso positivo(FP): Cuando se predice una clase pero en realidad pertenece a otra clase.
- Falso negativo(FN): Cuando se predice que no pertenece a una clase cuando realmente si es esa clase.

Exactitud (Accuracy)

La exactitud, se refiere a cuán cerca del valor real se encuentra el valor predicho. Esto se calcula como se muestra en la ecuación 3.8.

$$Exactitud = \frac{TP + TN}{TP + TN + FP + FN} \tag{3.8}$$

Precisión

La precisión representa el porcentaje de muestras que se han identificado como una clase. Esto se calcula como se muestra en la ecuación 3.9.

$$Precision = \frac{TP}{TP + FP} \tag{3.9}$$

Para entender de mejor manera estos conceptos, se presenta la Figura 3.4, donde la imagen izquierda hace referencia a un ejemplo de alta exactitud con baja precisión. Mientras que en la parte derecha hay un ejemplo de alta precisión con baja exactitud.

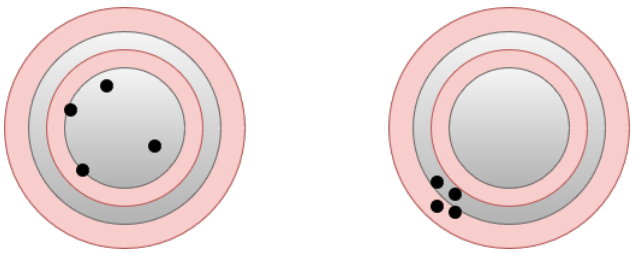


Figura 3.4: Ejemplo Alta exactitud, baja precisión y alta precisión, baja exactitud.

Sensibilidad (Recall)

La sensibilidad se define como la tasa de muestras bien etiquetadas entre el número de muestras relevantes. Esto se calcula como se muestra en la ecuación 3.10

$$Sensibilidad = \frac{TP}{TP + FN} \tag{3.10}$$

Con esta métrica se intenta simular el porcentaje de muestras que ha logrado predecir correctamente un modelo por cada etiqueta, por lo que es una métrica muy utilizada en clasificación multi-clase.

F1-Score

El F1-Score es otra métrica muy empleada porque nos resume la precisión y la sensibilidad en una sola métrica. El valor F1 asume que nos importa de igual forma la precisión y la sensibilidad. Esto se calcula como se muestra en la ecuación 3.11

$$F1 - Score = \frac{2 * sensibilidad * Precision}{sensibilidad + Precision} \tag{3.11}$$

Con estas últimas métricas podemos obtener cuatro casos posibles por clase:

- Alta precisión y alta sensibilidad: El modelo escogido maneja perfectamente esa clase.
- Alta precisión y baja sensibilidad: El modelo escogido no detecta la clase muy bien, pero cuando lo hace es altamente confiable.
- Baja precisión y alta sensibilidad: El modelo escogido detecta bien la clase, pero también incluye muestras de la otra clase.
- Baja precisión y baja sensibilidad: El modelo escogido no logra clasificar la clase correctamente

Matriz de confusión

La matriz de confusión, permite representar de manera visual el comportamiento de un sistema. En ella, el eje de abscisas representa la clase real y el de ordenadas representa la clase predicha por el modelo. En esta matriz, se pueden encontrar los valores de los positivos, negativos, falsos positivos y falsos negativos.

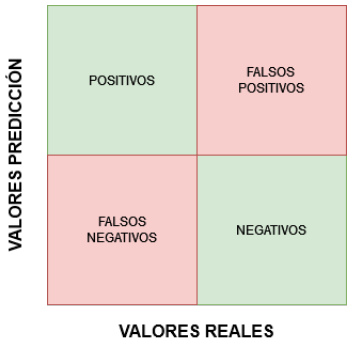


Figura 3.5: Estructura Matriz de confusión.

3.4. Análisis de trabajos desarrollados por otros autores

3.4.1. Word embeddings

Efficient Estimation of Word Representations in Vector Space

Word2Vec (6) se trata de un modelo predictivo de generación de word embeddings desarrollado por un equipo de investigadores dirigido por Tomas Mikolov en Google. Implementa dos modelos neuronales: CBOW y Skip-gram. En el primero, dado el contexto de la palabra objetivo, intenta predecirla. En el segundo, dada la palabra intenta predecir el contexto. Para los propósitos de este trabajo de titulo (clasificar un producto según su nombre de factura), podría ser útil el modelo CBOW, por lo que solo se explicará este.

CBOW (6) es arquitectura es muy similar a una red neuronal. Esencialmente intenta predecir una palabra objetivo a partir de una lista de palabras de contexto. Por ejemplo, dada la frase "Que tengas un gran día", elegiremos que nuestra palabra objetivo sea ”un” y nuestras palabras de contexto sean ["Que", "tengas", "gran","día"]. Lo que hará este modelo es tomar las representaciones distribuidas de las palabras de contexto para tratar de predecir la palabra objetivo. Así entonces, considerando las palabras de contexto como la entrada, se puede representar la arquitectura CBOW de la siguiente manera:

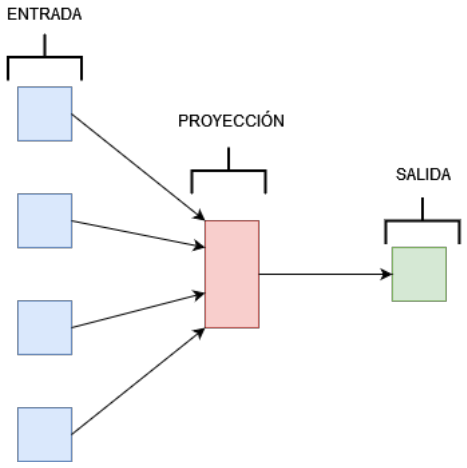


Figura 3.6: Arquitectura CBOW.

GloVe: Global Vectors for Word Representation

GloVe (7) significa Vectores Globales para la representación de palabras. Es un algoritmo de aprendizaje no supervisado desarrollado por investigadores de la Universidad de Stanford con el objetivo de generar incrustaciones de palabras agregando matrices globales de co-ocurrencia a partir de palabras de un corpus dado.

La idea básica detrás de la incrustación de palabras GloVe es derivar la relación entre las palabras de las estadísticas. A diferencia de la matriz de ocurrencia, la matriz de co-ocurrencia dice con qué frecuencia un par de palabras en particular ocurre juntas. Cada valor en la matriz de co-ocurrencia representa un par de palabras que ocurren juntas.

Para comprender la idea, se presenta el siguiente ejemplo:

Considerando que se tienen las siguientes frases:

- EL GATO TIENE MUCHA HAMBRE
- EL GATO TIENE QUE ESTAR JUGANDO

El objetivo de la matriz GLOVE es relacionar cada palabra con el corpus completo, por lo que se obtendría lo siguiente:

	EL	GATO	TIENE	MUCHA	HAMBRE	QUE	ESTAR	JUGANDO
EL	0							
GATO	2	0						
TIENE	1	2	0					
MUCHA	1	1	0	0				
HAMBRE	1	1	1	0	0			
QUE	1	1	1	0	0	0		
ESTAR	1	1	1	0	0	1	0	
JUGANDO	1	1	1	0	0	1	1	0

Tabla 3.2: Ejemplo matriz GLOVE

A partir de la matriz es posible obtener la puntuación que indica la cantidad de veces que esta palabra se relaciona con otra y así predecir el contexto de esta.

4 | Alternativas posibles de solución del trabajo

4.1. Posibles alternativas

4.1.1. Alternativa 1: Mejorar Red Actual

Como primera alternativa, se encuentra la posibilidad de mejorar el rendimiento de la red actual que ocupa la empresa.

Para esto, el primero paso será medir el desempeño actual de la red. La métrica más importante corresponde a la matriz de confusión, ya que con esta se pueden obtener otras métricas como el Recall y la precisión a partir de los resultados de verdaderos positivos, falsos negativos entre otros, valores que se pueden extraer a partir de la matriz de confusión como se vio en el capitulo 3.

El siguiente paso, previo al entrenamiento de la red, es contar con datos categorizados correctamente para poder entrenar a la red. Esta información se encuentra disponible por parte de la empresa en una base de datos en Amazon Web Services a la que el estudiante ya cuenta con credenciales para extraer la información necesaria.

Así entonces, para entrenar la red se utilizará un porcentaje de los datos como set de entrenamiento y el porcentaje restante como set de prueba (se estima 70 % del dataset para entrenamiento y un 30 % para pruebas).

Finalmente se vuelven a medir las métricas para comparar los resultados obtenidos pre y post entrenamiento.

4.1.2. Alternativa 2: Embeddings + Redes Neuronales

Como segunda alternativa se encuentra el uso de embeddings y redes neuronales para desarrollar un nuevo sistema para la clasificación de los productos.

El primer paso de esta alternativa corresponde a la extracción y pre-procesamiento de los datos. Ya sea filtrando el texto, eliminando caracteres extraños, etc. Luego, se realizaría la tokenización de datos considerando una cierta cantidad de palabras para un diccionario de palabras y así pasar todos los nombres de productos a una codificación one-hot.

Esta codificación sirve de entrada para una capa de embedding y así buscar asociaciones entre las palabras para que el modelo tenga la capacidad de aprender que palabras se suelen utilizar juntas

y poder predecir mejor los nombres de productos.

Una ve se tenga esto, será posible aplicar redes neuronales.

En el modelo a diseñar, la dimensión de la primera capa correspondería con la cantidad de palabras únicas . Posteriormente, vendría una capa flatten para conectar las capas ocultas. En estas capas los datos pasarán por una función de activación Linear.La función Linear deja los valores introducidos tal y como entran. Así no se perdería la información entregada por la capa de embedding.

Por otra parte, en la capa de salida, existe una función Softmax. La función Softmax transforma las salidas a una representación en forma de probabilidades, de tal manera que el sumatorio de todas las probabilidades de las salidas de 1.

$$f(z)_j = \frac{e^{z_j}}{\sum_{k=1}^{K} e^{z_k}} \tag{4.1}$$

Además, como función de pérdida, se utilizará 'Categorical Cross Entropy' (8), esta función suele ser adecuada en modelos de redes cuya salida representa una probabilidad, justo como lo es en este caso donde se hace una clasificación multiclase con función de activación Softmax.

Finalmente, el optimizador a utilizar corresponde a ADAM (9). La función del optimizador es calcular el gradiente de la función de coste por cada peso de la red. Como se quiere minimizar el error, se modificará cada peso en la dirección negativa del gradiente.

4.2. Ventajas y desventajas de las alternativas posibles

4.2.1. Alternativa 1

La primera alternativa se basa en analizar y utilizar el modelo ya desarrollado por la empresa y así ocupar la capacidad de entrenamiento de redes neuronales para mejorar su desempeño.

- En temas de dificultad, esta alternativa resulta tentadora dado que presenta una baja dificultad considerando que el modelo ya se encuentra desarrollado por la empresa.
- En temas de esfuerzo, esta alternativa requiere menos esfuerzo que la alternativa 2 dado que la información manual para entrenar el modelo se encuentra disponible en la base de datos de la empresa. Y las métricas de evaluación son cálculos conocidos.
- En cuanto a la efectividad de esta alternativa, es sabido que las redes neuronales son una excelente opción para la resolución de todo tipo de problemas, lo que incluye este caso como un problema de clasificación multiclase, por lo que a largo plazo y con entrenamiento esta alternativa es bastante prometedora.

4.2.2. Alternativa 2

La segunda alternativa se basa en hacer un pre-procesado de los datos y aplicar un modelo de redes neuronales con una capa de embedding.

- En temas de dificultad, esta alternativa presenta una mayor dificultad que la alternativa 1 puesto que requiere transformar la información presente en la base de datos y posteriormente armar un modelo de redes neuronales.
- En temas de esfuerzo, esta alternativa requiere un esfuerzo mayor que la alternativa 1 dado que cubre trabajo necesario(como el armado de la red) que en la alternativa 1 ya está resuelto.
- En cuanto a la efectividad de esta alternativa, dado que al investigar y estructurar personalmente una red neuronal con una capa de embedding, se estima que puede llegar a superar la alternativa anterior.

4.3. Características de comparación considerando ponderaciones de importancia

A partir de las alternativas descritas, se consideran las siguientes características para determinar la mejor alternativa:

- Dificultad: Hace referencia a que tantos problemas se pueden presentar en el desarrollo de alguna solución. Ya sea por falta de conocimiento, experiencia o información. Es una característica considerada no importante al momento de tomar una decisión, puesto que con organización del trabajo e investigación profunda, ambas alternativas son viables.
- Esfuerzo: Hace referencia al tiempo empleado y cantidad de trabajo que deberá aplicar el estudiante para llevar a cabo la alternativa. Es una característica importante pero no determinista.
- Efectividad: Hace referencia a los posibles resultados prometedores de cada alternativa. Es una característica importante y determinista del rumbo a tomar en este trabajo de título.

Finalmente se presenta la siguiente tabla comparativa de las alternativas a partir de las características descritas anteriormente.

	Dificultad	Esfuerzo	Efectividad
Alternativa 1	Baja	Medio	Alta
Alternativa 2	Media-Alta	Medio-Alto	Alta

Tabla 4.1: Características comparación

Si bien la alternativa 1 pareciera ser la más simple y con buenos resultados, no se descarta la efectividad que se puede conseguir la segunda alternativa al momento de aplicarla a este problema. Así entonces es que la alternativa 2 parece ser la mejor para resolver este problema.

5 | Propuesta de solución

A partir de las características de cada alternativa presentadas en el capítulo anterior, es posible notar que la alternativa 1 posee una dificultad baja, requiere menos esfuerzo y sin embargo puede alcanzar una efectividad alta. Por su parte la alternativa 2 posee mayor dificultad y requiere mayor esfuerzo al tratarse de un modelo que debe ser desarrollado por cuenta propia y se considera que al tratarse de una red neuronal con una capa de embedding también alcanza una efectividad alta y posiblemente mayor a la que posea la alternativa 1.

A partir de esto es que se decide desarrollar la alternativa 2, ya que, si bien la alternativa 1 es mas sencilla, la alternativa 2 permite al estudiante obtener mayor conocimiento y experiencia en el uso de redes neuronales, lo cual permitiría generar una solución que compita a nivel de precisión con la red actual de la empresa.

Finalmente, tomando en cuenta la investigación realizada en el capítulo 3, una red neuronal con una capa de embedding permite desarrollar modelos más simples estructuralmente y a la vez obtener buenos valores de precisión en clasificación multiclase, por lo que se procederá con la alternativa 2 para obtener un modelo simple y potente en la clasificación de nombres de productos.

6 | Desarrollo de la solución

6.1. Desarrollo de las etapas planteadas

6.1.1. Extracción y Pre – Procesado

Tal como se dijo en el capitulo 2, el primer paso es extraer la información de los productos y sus categorías desde la base de datos de la empresa. En la tabla 6.1 y 6.2 se deja una vista previa de la información descargada:

KeyItem	idCategoría
08245332-126-panchito-alfajor-800-g-20-und	2
08245332-1311-alfajor-bon-o-bon-leche-40-und	2
08245332-139112-alfajores-maicena-x-12	2
08245332-1242-alfajor-premium-720-gr	2
...	...

Tabla 6.1: Vista previa productos

idCategoría	Categoría
2	alfajores
3	bizcochos
4	caramelos envasados
5	chocolates
6	galletas
...	...

Tabla 6.2: Vista previa categorías

Es importante destacar que se consta de 149 categorías y 678.165 productos a clasificar en este trabajo.

El siguiente paso es segmentar los datos en set de prueba y entrenamiento, para esto se considera un 70 % de los datos como set de entrenamiento y un 30 % set de prueba.

Una vez hecho esto, se realiza la tokenización de los datos de prueba y entrenamiento considerando las 2000 palabras más importantes (ya que así lo realiza la empresa). Esto permite crear un diccionario de 2000 palabras y es con este diccionario que se pasa el texto de productos a una codificación one-hot para que ya puedan ser entregados como entrada a los modelos de deep learning.

En el siguiente Script se encuentra disponible todo el proceso de este punto.

6.1.2. Aplicación del modelo existente

Con las datos ya pre-procesados, se estudia el modelo que posee la empresa actualmente para el proceso de clasificación de categorías, el cual posee la siguiente estructura:

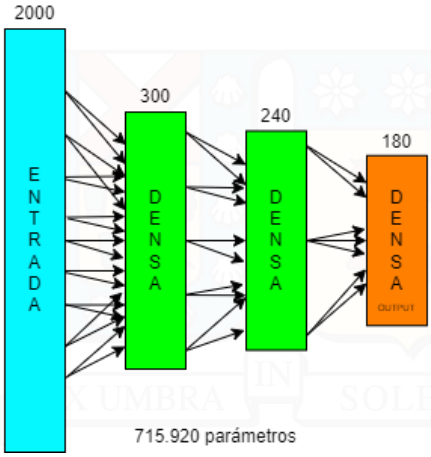


Figura 6.1: Estructura red existente.

Es posible observar en la figura 6.1 que la red esta compuesta por 715.920 parámetros distribuidos entre toda la red. Esta red posee principalmente una capa de entrada de 2.000 neuronas (las 2.000 palabras más repetidas filtradas en el pre-procesamiento), una capa densa de 300 neuronas con función de activación ReLu y dropout, después de esta se tiene otra capa densa pero de 240 neuronas con función de activación ReLu y dropout y finalmente una capa densa de 180 neuronas que corresponde a la salida con función softmax. Si bien se tenían 149 categorías, la salida es de 180 neuronas porque los números de las categorías no son del todo una secuencia (ya que empieza en la categoría 2 y existe hasta la categoría 179).

Con esta red, ya es posible realizar el entrenamiento. Se realiza con 20 épocas y un tamaño de batch de 128 datos, todo esto con optimizador adam y función de pérdida categorical crossentropy, este proceso queda disponible en el siguiente Script.

6.1.3. Determinar precisión de clasificación

Con el modelo ya entrenado, el siguiente paso es calcular la precisión que obtiene para las distintas categorías, por eso en primera instancia se observa la precisión general del modelo haciendo uso de keras que dispone un método para evaluar la red rápidamente (10). Así se obtiene una precisión para el set de entrenamiento de 89 % y una precisión para el set de prueba de 86 %.

Si bien esto nos da una idea del rendimiento general del modelo, es posible observar también las gráficas de la función de pérdida y precisión durante el ajuste del modelo. Este desarrollo y posteriores de esta sección, se encuentran disponibles en el Git de los Script. Sin embargo, el Script que permitió generar las gráficas de la figura 6.2, proviene del libro Deep Learning with Python de (4), el cual solo fue modificado el idioma en los títulos y leyendas. Así entonces, las gráficas obtenidas se presentan en la siguiente figura:

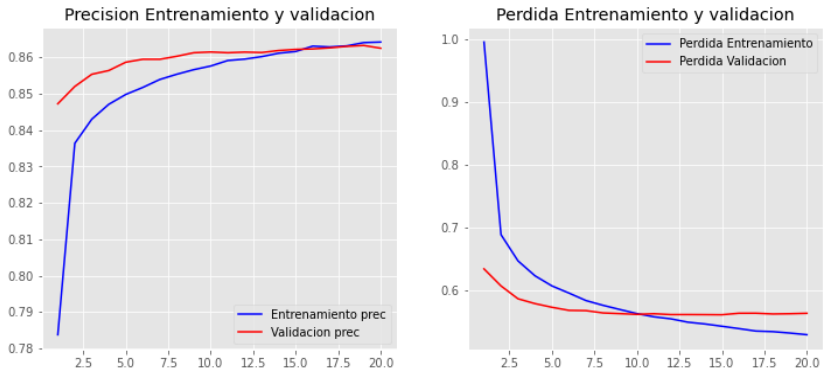


Figura 6.2: Función de pérdida y precisión en las distintas épocas de la red de la empresa.

Es posible notar que el modelo comenzó con una precisión regular del 78.5 % aproximadamente en el set de entrenamiento, pero con una precisión bastante buena en el set de test, iniciando con casi un 85 % de precisión. Y ambas fueron aumentando a lo largo de las épocas. Por su parte la función de pérdida disminuye considerablemente a lo largo de las épocas y por la forma de las curvas se puede deducir que con más épocas, la pérdida en el entrenamiento seguiría disminuyendo, pero finalmente la de validación se ve casi en un estado estacionario por lo que no se requieren más épocas, sino se podría generar un overfitting en el entrenamiento.

Aún con la precisión general que nos entrega keras y las gráficas obtenidas durante el ajuste de la red, la precisión se esta observando de manera muy general considerando la cantidad de categorías, es por esto que se obtiene la matriz de confusión de la red. Esta matriz de confusión se lleva a un mapa de calor, obteniendo así la siguiente figura:

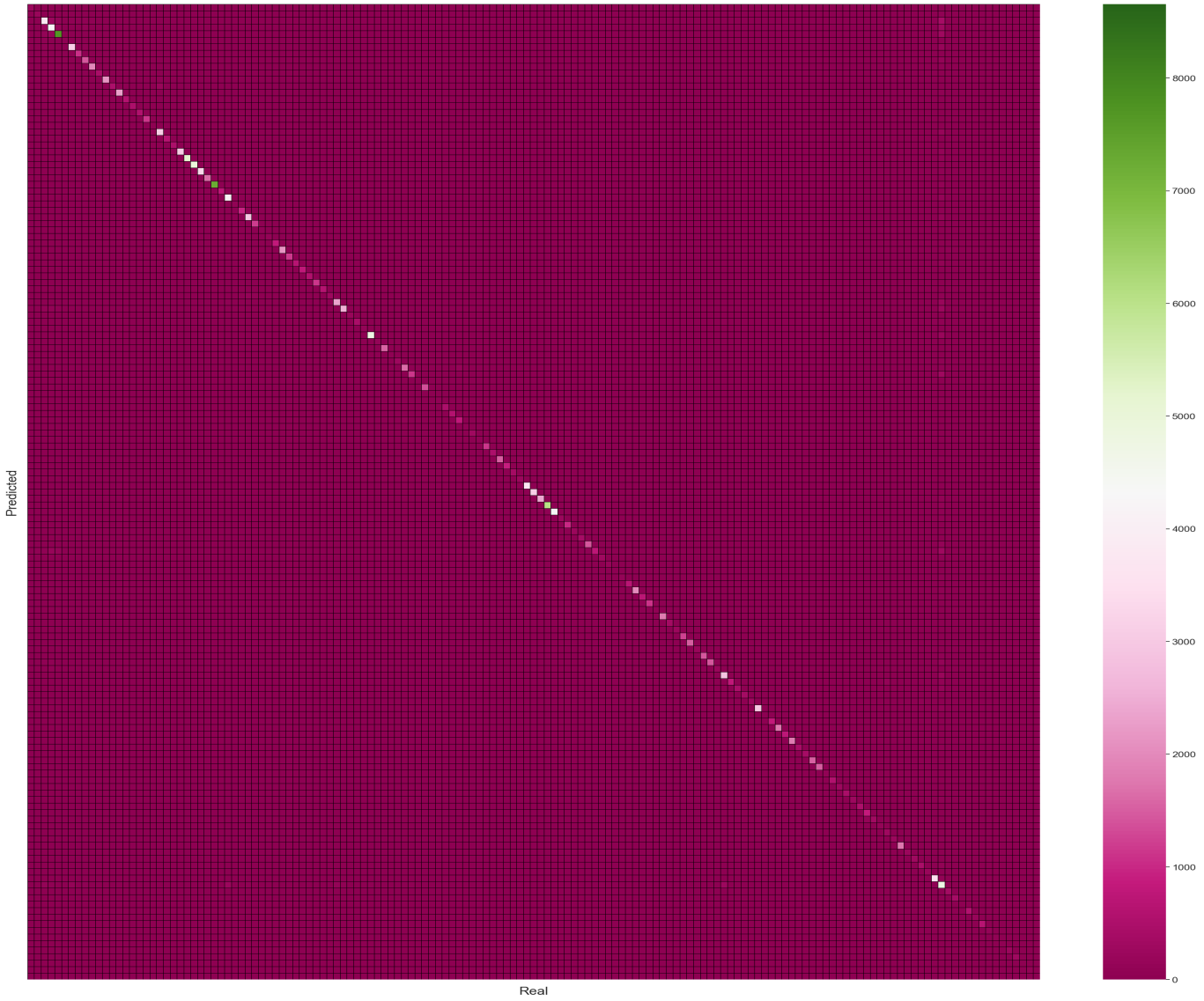


Figura 6.3: Mapa de calor de la matriz de confusión del modelo de la empresa.

A partir de la figura 6.3, es posible notar que principalmente la diagonal tiene un color distinto, esto ya que la mayoría de los datos se encuentran clasificados correctamente. Pero sin embargo existen datos fuera de la diagonal que se pueden ver levemente en algunas categorías.

Para realizar un mejor análisis, se toma la diagonal (que serian los datos clasificados correctamente) para calcular la precisión. Con este calculo, se realiza un gráfico de áreas apiladas para distinguir la cantidad de datos por categoría y el porcentaje de datos clasificados correctamente. Esto se muestra en la figura 6.4

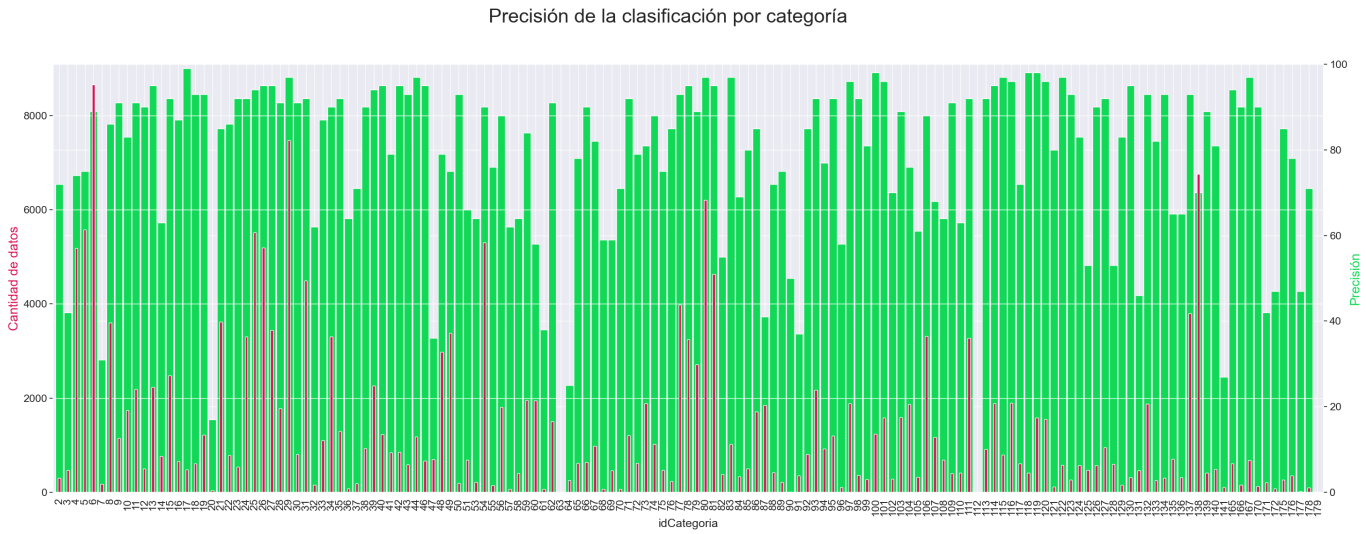


Figura 6.4: Precisión del modelo de la empresa por categoría.

Si bien, se ve generalmente una precisión bastante buena entre las categorias, es posible notar que para las categorías con mayor cantidad de datos, se obtiene una mayor precisión, esto se debe a un desbalance de datos entre las clases, por lo que categorías con menos datos, son mas propensas a tener peor clasificación.

6.1.4. Proponer un modelo de Deep Learning

El último paso, es proponer una arquitectura que supere al proceso actual que posee la empresa. Para esto, se utilizará un modelo de deep learning con una capa de embedding (11). Las capas de embedding facilitan el aprendizaje automático en entradas grandes, como vectores dispersos que representan palabras, tal como es en este caso. Así entonces, se muestra en la figura 6.5 la estructura de la red diseñada:

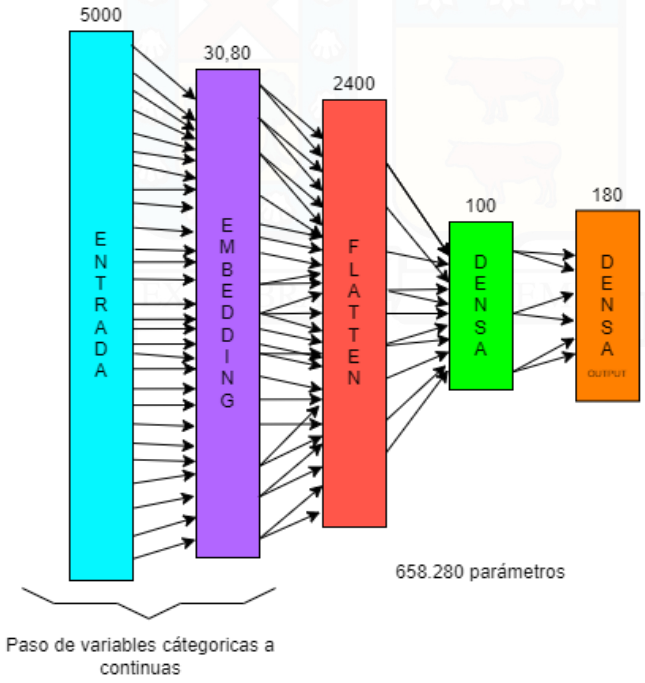


Figura 6.5: Estructura de la red diseñada.

Es posible visualizar que se trata de una red con menos parámetros respecto a la que posee la empresa, dado que esta posee 658.280 parámetros, es decir, 57.640 parámetros menos que en la red de la empresa. Además, esta red considera solo una capa de dropout y se le incorpora regularización L2 (12) para obtener una mejor función de pérdida en el set de validación. Es importante destacar que esta red tiene en su capa oculta una función Linear en lugar de una función ReLu como es en el caso de la empresa. Esto ya que la capa de embedding pasa los valores del vector de entrada a valores flotantes que podrian ser negativos, por lo que para no eliminar esta información, se utiliza la función Linear.

Es importante señalar que si bien esta red también recibe los datos tokenizados, se consideran las 5.000 palabras más repetidas. Esto ya que se tienen aproximadamente 48.000 palabras distintas, sin considerar los números que vienen en el nombre de los productos. Por lo que se considera que las 2.000 que utiliza la empresa son pocas respecto a la cantidad de palabras que se poseen.

Otra diferencia que posee esta red, es que debido a la capa de embedding que aumenta la dimensión de su salida, se requiere una capa flatten para ajustar la dimensión y poder incorporar las otras capas de la red. Finalmente se realiza el entrenamiento de esta red con solamente 10 épocas y un tamaño de batch de 128 datos, todo esto con optimizador adam y función de pérdida categorical crossentropy.

Con el modelo ya entrenado, es posible determinar la precisión tal cual se hizo con el modelo de la empresa. En este caso al observar la precisión general que entrega keras, se obtiene un 95.7 % para el set de entrenamiento y un 88.5 % para el set de prueba. Es posible ver una mejora significativa para el set de prueba y una mejora leve en el set de validación.

Se obtienen también las gráficas de la función de pérdida y precisión durante el ajuste del modelo tal cual se hizo con la red de la empresa y se obtiene lo siguiente:

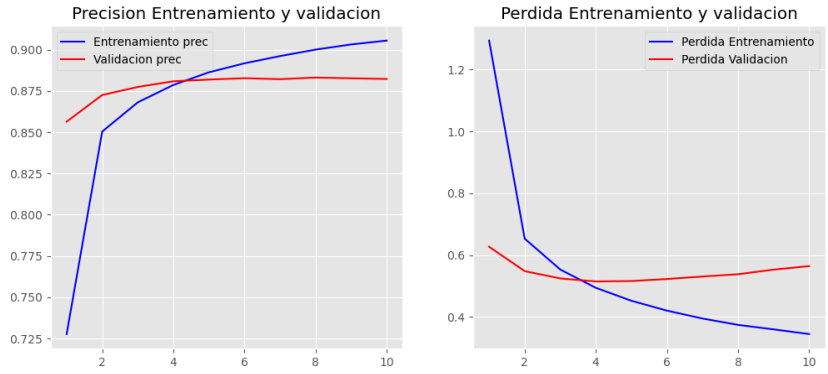


Figura 6.6: Función de pérdida y precisión en las distintas épocas de la red con capa de embedding.

En comparación con el modelo anterior, el set de entrenamiento empezó con una precisión menor de 72 % aproximadamente, pero llego a un valor mucho más alto de poco más el 90 %. Mientras que la precisión del set de prueba empieza cerca 86 % y termina cerca del 88 %. Por su parte la función de pérdida disminuye considerablemente a lo largo de las épocas, pero la función de pérdida de validación comienza a aumentar después de la época 6 aproximadamente, pero aún se mantiene bajo el 0.6 al igual que en el modelo anterior.

Entonces es posible decir que este modelo esta superando de forma general al modelo anterior, pero aún falta analizar categoría a categoría, este análisis se realiza nuevamente con la matriz de confusión. Así entonces, es posible obtener la siguiente figura:

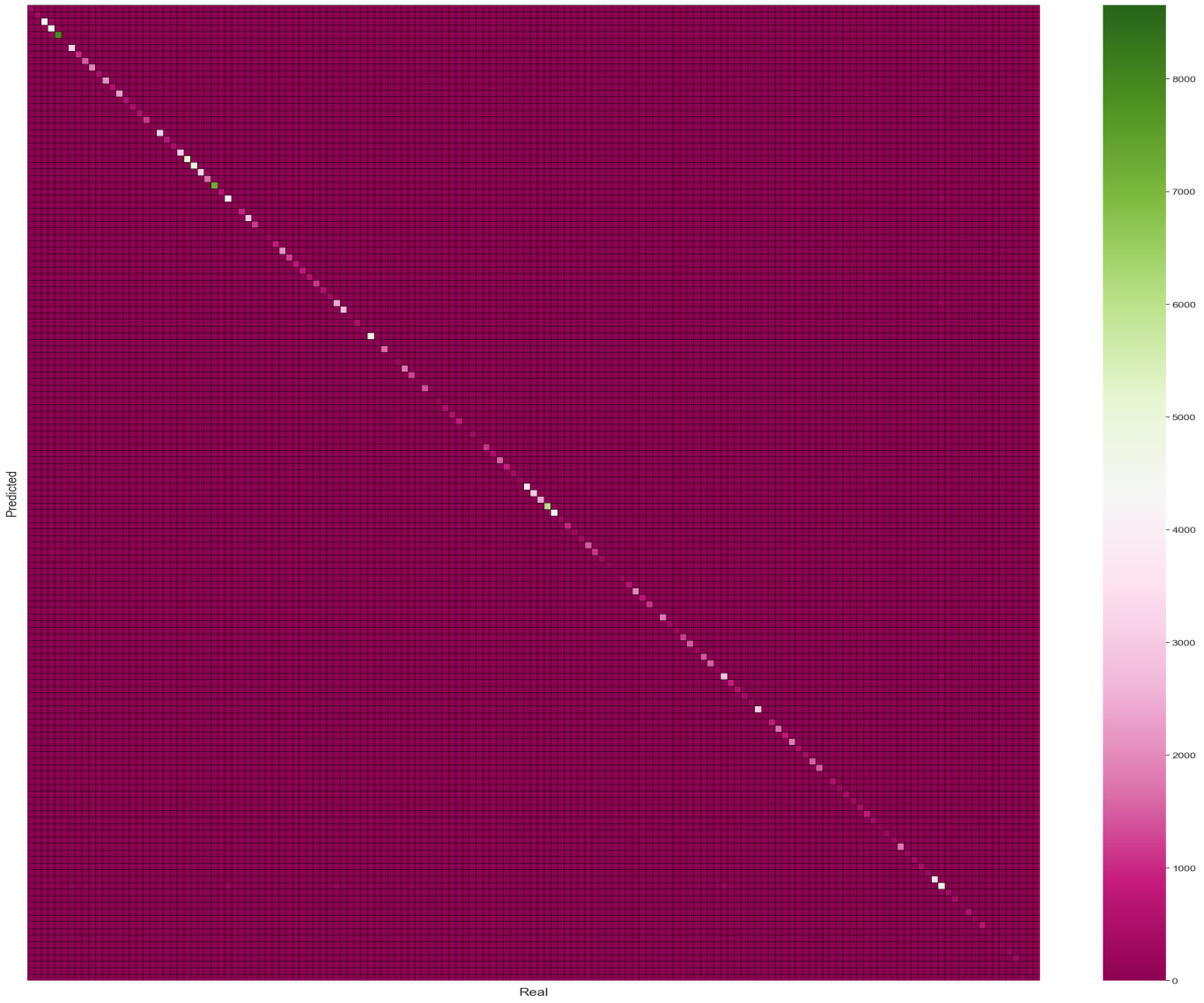


Figura 6.7: Mapa de calor de la matriz de confusión del modelo propio.

De la matriz de confusión es posible notar que al igual que en el caso anterior la mayoría de los datos se encuentran en la diagonal, sin embargo, en este caso pareciera haber menos datos fuera de la diagonal, ya que prácticamente toda la matriz fuera de la diagonal tiene valor 0.

Luego, para realizar un análisis categoría a categoría, se obtiene el gráfico de áreas apiladas de cantidad de datos y precisión para este modelo, tal como se ve en la siguiente figura:

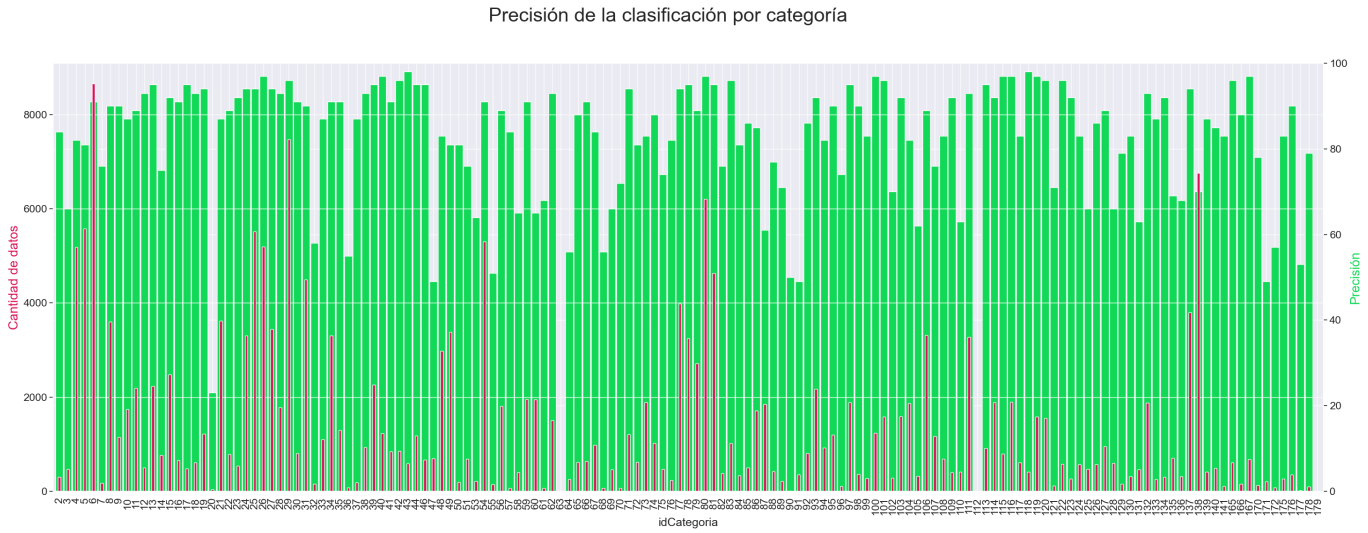


Figura 6.8: Precisión del modelo de la diseñado por categoría.

A primera vista se ve que la precisión obtenida por categoría, es similar a las obtenidas con el modelo de la empresa. Por lo que para conseguir un mejor análisis, se realiza un gráfico considerando solo los porcentajes de precisión para cada categoría, tal como se ve en la siguiente figura:

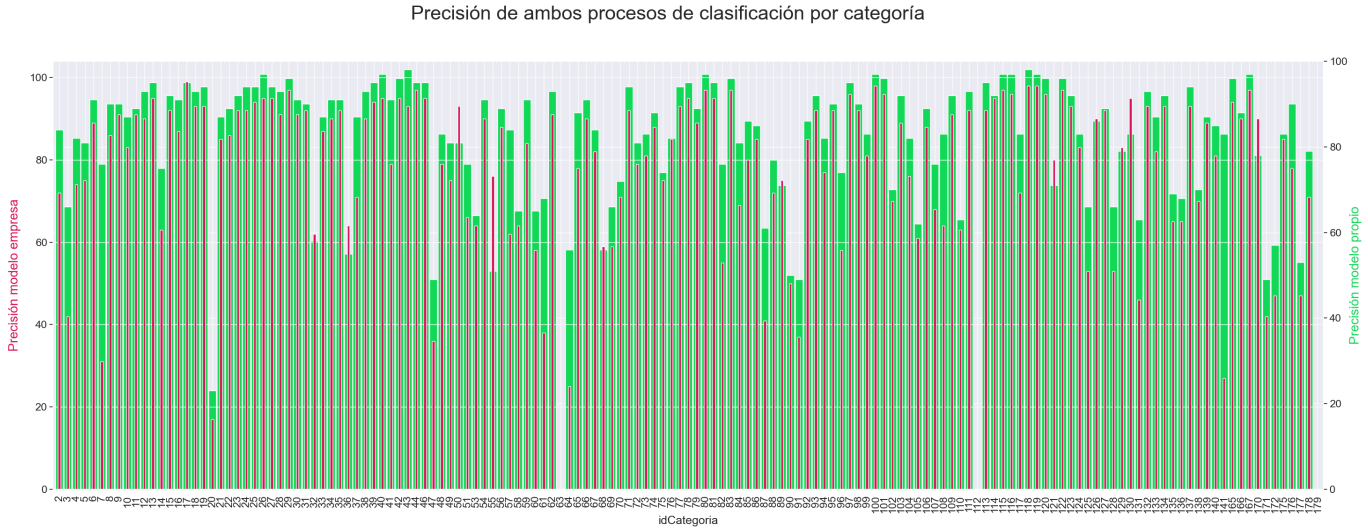


Figura 6.9: Precisión ambos modelos por categoría.

Con los resultados de la figura 6.9 es posible notar que el modelo diseñado supera en todas las categorías la clasificación realizada por el modelo de la empresa. Además, en los casos que se cuenta con pocos datos, suele superar en gran medida el valor obtenido con el modelo de la empresa. Por lo que el modelo diseñado además de estar categorizando mejor, esta siendo a la vez más robusto frente a la falta de datos.

7 | Conclusiones y posibles mejoras a la alternativa seleccionada

Finalmente se puede decir que la capa de embedding permitió mejorar el proceso de clasificación realizado por la empresa aumentando considerablemente la precisión en la fase de entrenamiento (de 89 % a 95.7 %) y aumentando levemente en la fase de validación (de 86 % a 88.5 %). Si bien el cambio no es grande en el set de validación, si permite clasificar de mejor manera en todas las categorías, aún con las clases desbalanceadas y además lo hace con una menor cantidad de parámetros de red. Por lo que se puede decir que el modelo propuesto supera al modelo actual de la empresa y a la vez es mas robusto frente a la falta de datos, cumpliendo así con el objetivo de este proyecto de título.

Una posible mejora a la alternativa seleccionada, podría ser antes de entrenar el modelo, realizar un dataset augmentation controlado a los nombres de productos en las clases desbalanceadas, dado que existe incluso una categoría con 1 dato en el set de prueba y así varían entre decenas, centenas o miles de datos. Por lo que claramente el modelo de deep learning se ajustará mucho mejor a las categorías con más datos en comparación a los que tienen muy pocos. Esto permitiría una mejor precisión tanto en el set de entrenamiento (que ya esta bastante bien clasificado) como en el set de prueba, logrando tener valores de precisión superiores al 90 %.

Bibliografía

[1] . B. M. S. Ruiz C. A. (2001) Redes neuronales: Conceptos básicos y aplicaciones. [Online]. Available: https://www.frro.utn.edu.ar/repositorio/catedras/quimica/5_anio/orientadora1/monograias/matich-redesneuronales.pdf 3.1

[2] IBM., "El modelo de redes neuronales." 2021. [Online]. Available: https://www.ibm.com/docs/es/spss-modeler/SaaS?topic=networks-neural-model 3.1

[3] I. García Ferrero, "Estudio de word embeddings y métodos de generación de meta embeddings." 2018. [Online]. Available: https://addi.ehu.es/bitstream/handle/10810/29088/MemoriaTFG_IkerGarciaFerrero.pdf?sequence=3 3.2.1

[4] F. Chollet, "Deep learning with python, edición 1," 2017. 3.2.2, 6.1.3

[5] M. Campos Mocholí. (2021) Clasificación de textos basada en redes neuronales (doctoral dissertation, universitat politècnica de valència). [Online]. Available: https://riunet.upv.es/bitstream/handle/10251/172276/Campos%20-%20Clasificacion%20de%20textos%20basada%20en%20redes%20neuronales.pdf?sequence=1 3.3

[6] K. C. Greg Corrado Tomas Mikolov and J. Dean., "Efficient estimation of word representations in vector space," 2013. [Online]. Available: https://arxiv.org/abs/1301.3781 3.4.1

[7] J. Pennington, R. Socher, and C. D. Manning., "Glove: Global vectors for word representation. empirical methods in natural language proces- sing (emnlp)," 2013. [Online]. Available: https://aclanthology.org/D14-1162.pdf 3.4.1

[8] F. Chollet *et al.* Keras, losses functions. [Online]. Available: https://keras.io/api/losses/ 4.1.2

[9] C. Francois *et al.* Keras, optimizers. [Online]. Available: https://keras.io/api/optimizers/ 4.1.2

[10] F. Chollet *et al.* Keras. evaluate-method. [Online]. Available: https://keras.io/api/models/model_training_apis/#evaluate-method 6.1.3

[11] C. Francois *et al.* Keras, embedding layer. [Online]. Available: https://keras.io/api/layers/core_layers/embedding/ 6.1.4

[12] C.Francois *et al.* Keras, regularizers. [Online]. Available: https://keras.io/api/layers/regularizers/ 6.1.4