

# Análisis Reputacional de Aerolineas

Rodolfo Jesús Ramirez Lucario

## 1 Introducción

Hoy en día, las redes sociales se han convertido en una de las principales formas en las que las personas expresan sus opiniones. En particular, Twitter es una plataforma donde los usuarios comparten de manera rápida y directa sus experiencias, quejas y comentarios sobre distintos servicios. Esto hace que sea una fuente muy útil para analizar qué piensan realmente los consumidores sobre una empresa. En el caso de las aerolíneas de Estados Unidos, este tipo de análisis es especialmente relevante porque la satisfacción del cliente influye muchísimo en su reputación y en la preferencia de los usuarios.

En este trabajo se realizó un análisis de sentimiento de tweets relacionados con varias aerolíneas estadounidenses, utilizando técnicas de procesamiento de lenguaje natural (NLP). El objetivo principal es identificar si los comentarios de los usuarios son positivos, negativos o neutrales, y ver cómo se distribuyen estos sentimientos entre diferentes aerolíneas.

Para ello, se llevaron a cabo varias etapas: limpieza del texto, análisis exploratorio, tokenización, vectorización y finalmente la construcción y entrenamiento de un modelo de clasificación basado en redes neuronales. También se evalúa el desempeño del modelo y se analizan los resultados obtenidos, tanto a nivel general como por aerolínea.

## 2 Limpieza

La base de Datos es una tabla que cuenta con las columnas que se muestran en la tabla (1) de las cuales sólo nos interesan ocupar **tweet\_id**, **airline\_sentiment**, **airline\_sentiment\_confidence**, **airline** y **text**. La columna de **airline\_sentiment** tiene 3 posibles valores **neutral**, **positive** y **negative**. Empezamos haciendo un análisis de Datos Faltantes en dónde notamos que no se tienen, además cada tweet tiene asignada una emoción. A continuación detectamos que la forma en la que se identifica la aerolínea a la que va dirigida el tweet es a través del símbolo @ por ejemplo @VirginAmerica, @AmericanAir, etc ... también se referencia a otras personas apartir de este símbolo. Dado que consideramos que estas referencias no serán importantes en la identificación de la emoción del tweeter entonces se decidió quitarla, esto mediante expresiones regulares. Así mismo, para que los modelos que entrenaremos no distingan entre palabras en mayúsculas o en minúsculas, se decidió pasar

todo a minúsculas. También encontramos textos con signos de exclamación o interrogración, signos de puntuación etc... y dado que consideramos que nuestro modelo sólo usará palabras para realizar la predicción del sentimiento entonces retiramos también estos, lo mismo con los números. Adicionalmente, quitamos espacios en blanco que pudieran haber de más, esto pues lo consideramos un error de escritura.

Variable	Descripción
tweet_id	Identificador único del tweet.
airline_sentiment	Sentimiento asignado al tweet (positivo, negativo, neutral).
airline_sentiment_confidence	Nivel de confianza del sentimiento asignado.
negativereason	Razón específica del sentimiento negativo (si aplica).
negativereason_confidence	Nivel de confianza de la razón negativa.
airline	Aerolínea mencionada en el tweet.
airline_sentiment_gold	Sentimiento corregido manualmente (si existe).
name	Nombre del usuario que publicó el tweet.
negativereason_gold	Razón negativa corregida manualmente (si existe).
retweet_count	Número de veces que el tweet fue retuiteado.
text	Texto completo del tweet.
tweet_coord	Coordenadas geográficas del tweet (si existen).
tweet_created	Fecha y hora en que se creó el tweet.
tweet_location	Ubicación declarada del usuario.
user_timezone	Zona horaria del usuario.

Cuadro 1: Descripción de variables del dataset de sentimiento de aerolíneas

Continuando con el pretratamiento de los datos, quitamos las StopWords. Estas son palabras muy comunes en un idioma que aportan poca o ninguna información semántica útil para tareas de procesamiento de lenguaje natural (NLP), por ejemplo I, You, What, the, ... El método implementado para retirar las StopWords fue usar un corpus de stopwords en Inglés del módulo **NLTK** (Natural Language Toolkit). Con el fin de reducir el vocabulario a considerar para nuestros modelos, usamos un Lemantizador, el cuál identifica primero la categoría gramatical de una palabra (verbo, sustantivo, adjetivo, etc.) y luego consulta un lexicón morfológico para convertir la palabra original a su forma raíz. Este procedimiento permite agrupar diferentes variaciones de una misma palabra en una sola representación, reduciendo la dimensionalidad del vocabulario y alineando palabras semánticamente equivalentes.

Finalmente, notamos que había muchos tweets que tenían emojis y consideramos que estos tenían gran valor en determinar si un tweet tenía un mensaje positivo o uno negativo. Para tratar estos casos y evitar que nuestro modelo los rechace, ocupamos el módulo emoji para llevar cada uno de los emojis a cadenas de texto y de esta forma queda como una palabra más dentro del corpus. Podemos ver un ejemplo de lo que hace este preprocesamiento en la Figura 2.

```
'@SouthwestAir 😱 you won't let me change my reservation online so now I'm just wasting my time. http://t.co/mHA3xXaeD5'
```

```
'emoji_grinning_face_with_sweat let change reservation online wasting time'
```

Figura 1: Limpieza de texto.

### 3 Análisis Exploratorio

La intención de este proyecto es evaluar la reputación de las aerolíneas, pero dado que no existen APIs que gratuitamente permitan descargar información Histórica reciente de Tweeter, se decidió partir el conjunto de datos en 3. Tenemos un total de 14640 tweets, se dejó como conjunto de testo a los tweets que aparecen aparitr del 24 de Febrero del 2015 (un total de 1344). Para realizar el análisis exploratorio se partió el conjunto de Datos restante en 2, un conjunto de entrenamiento y otro de validación en proporción 80-20 (10636 y 2660 datos respectivamente). Todo el análisis exploratorio se realizó sobre el conjunto de entrenamiento y el conjunto de validación se ocupó para escoger al mejor modelo.

Podemos ver en la Figura 2 la distribución del número de tweets por Aerolínea, United es la más comentada durante el periodo de tiempo que se recabó la información, en el otro extremo está Virgin América. Por otro lado, en la Figura 3 podemos ver que en general, las aerolíneas tienen más tweets negativos que positivos.

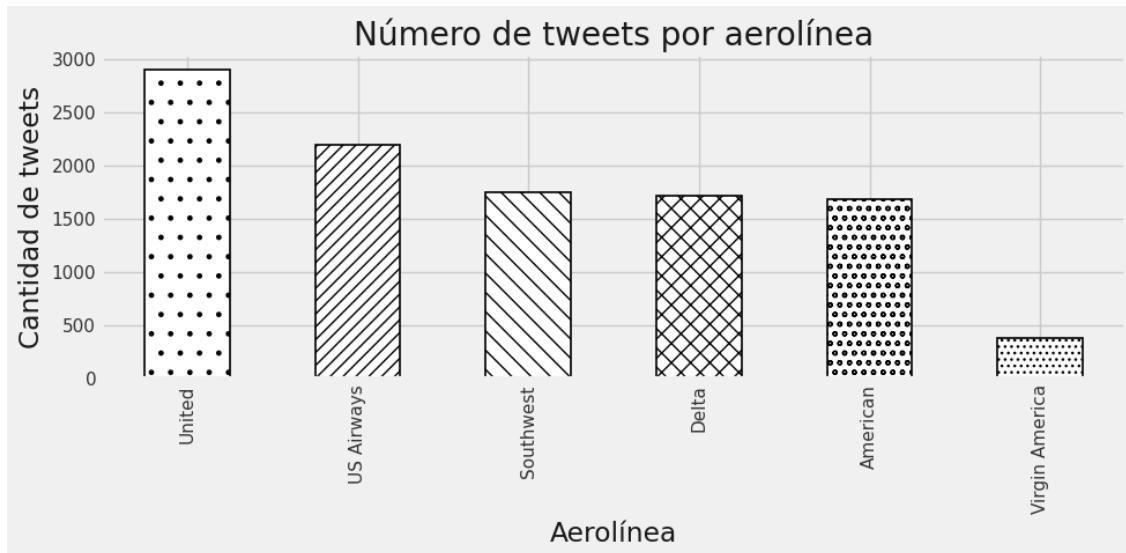


Figura 2: Número de Tweets por Aerolinea.

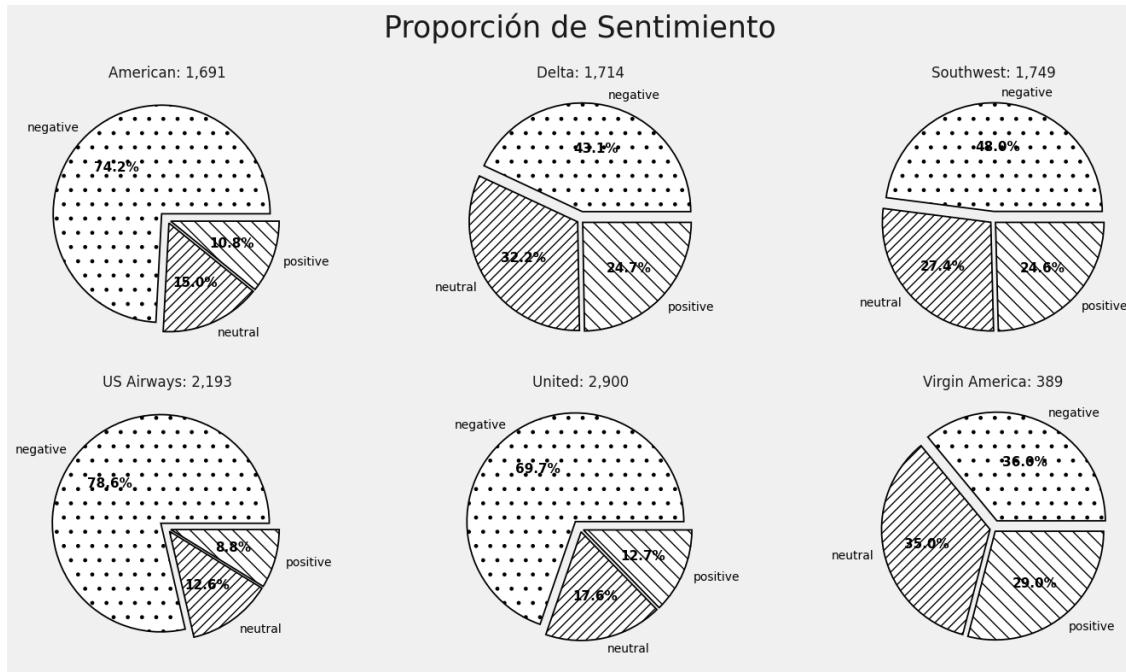


Figura 3: Porporción de sentimiento de Tweets por Aerolínea.

En la Figura 4 podemos ver las palabras más populares en los tweets separadas por sentimiento. Cómo se puede notar, hay palabras que se repiten en las tres gráficas, de aquí podemos darnos una idea de que el sentimiento reflejado en un tweet no proviene de las palabras aisladas, si no del contexto en el que se usan. Requerimos de un modelo que tome esto en consideración.



Figura 4: Palabras más populares por sentimiento.

Aunque no tomaremos en cuenta un análisis temporal en nuestra modelación, es interesante observar la evolución temporal del sentimiento por Aerolínea en la Figura (5). Podemos ver que los picos en las quejas ocurren mas o menos por las mismas fechas, eso puede haberse debido a alguna condición climática o a algo que afectó a todas las compañías.

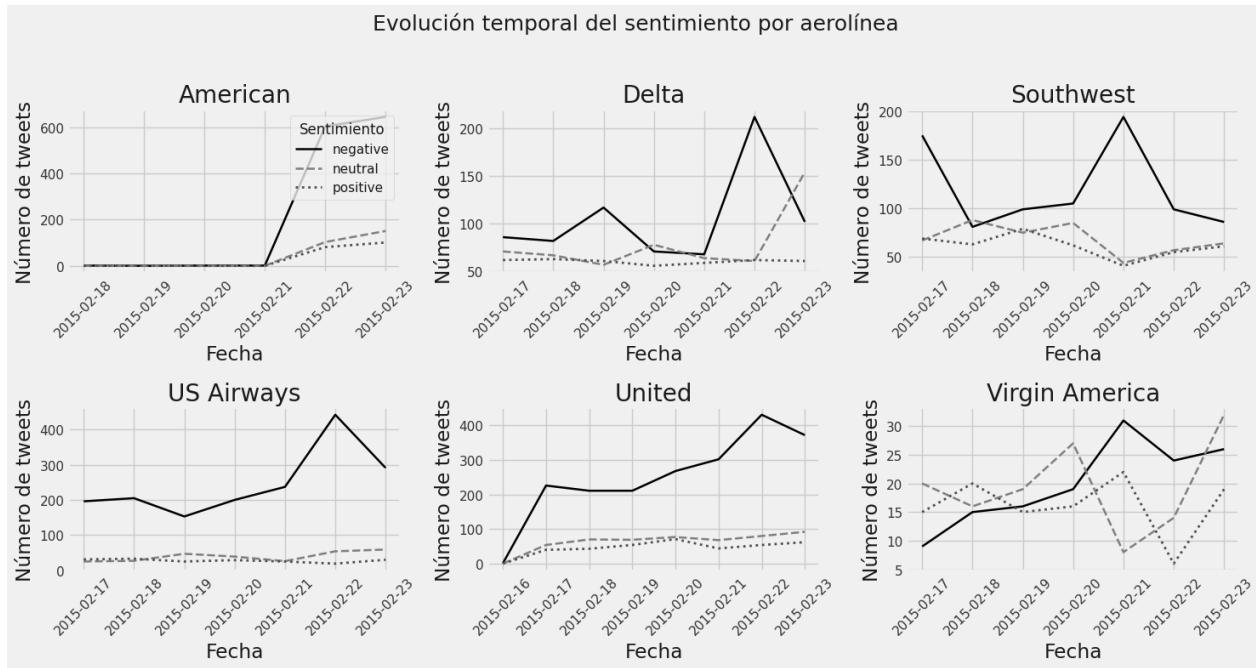


Figura 5: Evolución temporal del sentimiento por Aerolínea.

## 4 Modelación

Dado que queremos evitar sesgos positivos o negativos, es decir, evitar que el modelo tienda a predecir más tweets como positivos que como negativos, tomamos la decisión de balancear las muestras para el entrenamiento y evaluar en un conjunto que refleje la proporción verdadera del sentimiento de los tweets. Para balancear las clases minoritarias, usamos el método de remuestreo.

### 4.1 Random Forest

Ocuparemos dos enfoques para la modelación; primeramente utilizamos un modelo clásico de Machine Learning: un Random Forest con el cuál, podemos tener un poco más de interpretabilidad pues en particular podemos saber la importancia de las variables en el modelo. La importancia de una variable se basa en la reducción de impureza que producen los splits donde dicha variable es utilizada.

Si consideramos un conjunto de entrenamiento con  $N$  observaciones y un bosque compuesto por  $B$  árboles  $\{T_1, \dots, T_B\}$ .

Para un nodo  $t$  de un árbol, la impureza se denota por  $i(t)$ , que esta dada por el coeficiente de Gini i.e.

$$i(t) = 1 - \sum_c p(c | t)^2,$$

donde  $p(c | t)$  es la proporción de ejemplos de la clase  $c$  en el nodo  $t$ .

Si un split en el nodo  $t$  lo divide en nodos hijos  $t_L$  y  $t_R$  con tamaños  $N_L$  y  $N_R$ , la reducción

de impureza asociada al split es

$$\Delta i(t) = i(t) - \left( \frac{N_L}{N_t} i(t_L) + \frac{N_R}{N_t} i(t_R) \right),$$

donde  $N_t = N_L + N_R$  es el tamaño del nodo padre.

Si el split en el nodo  $t$  utiliza la variable  $x_j$ , entonces su contribución a la importancia de  $x_j$  en el árbol  $T_b$  es

$$\Delta I_b(x_j; t) = \Delta i(t) \frac{N_t}{N},$$

lo cual pondera la reducción de impureza por el tamaño relativo del nodo.

La importancia total de la variable  $x_j$  en el Random Forest es

$$\text{Imp}(x_j) = \frac{1}{B} \sum_{b=1}^B \sum_{\substack{t \in T_b \\ \text{si } x_j \text{ separa el nodo } t}} \Delta I_b(x_j; t).$$

Finalmente, las importancias se normalizan para que su suma sea uno:

$$\text{Imp}_{\text{norm}}(x_j) = \frac{\text{Imp}(x_j)}{\sum_k \text{Imp}(x_k)}.$$

Las variables con mayor importancia tienen un  $\text{Imp}_{\text{norm}}$  alto. Se realizó un CrossValidation para escoger el número de árboles adecuado para el modelo, se propusieron 40, 60, 80, 100, 120, 200 y 300.

Es importante mencionar que para ingresar el texto a este modelo se le realizó una transformación usando el módulo de python llamado **TfidfVectorizer**. El **TfidfVectorizer** transforma el conjunto de tweets en una matriz numérica basada en la ponderación *TF-IDF* (*Term Frequency-Inverse Document Frequency*), que mide la relevancia de cada término dentro de un tweet y en el corpus completo. Su funcionamiento puede describirse mediante los siguientes pasos:

1. **Tokenización:** Cada tweet se divide en palabras o *tokens*. Dependiendo de la configuración, también puede generar  $n$ -gramas (por ejemplo, unigramas o bigramas).
2. **Construcción del vocabulario:** Se crea un diccionario de todos los términos que aparecen en el corpus.
3. **Cálculo de la frecuencia del término (TF):** Para un término  $t$  en un tweet  $d$ , su frecuencia se define como

$$\text{TF}(t, d) = \frac{\text{número de apariciones de } t \text{ en } d}{\text{total de términos en } d}.$$

4. **Cálculo del inverso de la frecuencia de documentos (IDF):** Penaliza términos muy frecuentes en el corpus completo. Para un corpus con  $N$  tweets y un término presente en  $n_t$  tweets,

$$\text{IDF}(t) = \log \left( \frac{N}{1 + n_t} \right) + 1.$$

5. **Cálculo del peso TF-IDF:** El peso final asignado al término  $t$  en el documento  $d$  es

$$\text{TFIDF}(t, d) = \text{TF}(t, d) \cdot \text{IDF}(t).$$

6. **Normalización:** Cada vector de un tweet puede normalizarse.

7. **Salida matricial:** El resultado final es una matriz dispersa de tamaño

$$(\text{número de tweets}) \times (\text{tamaño del vocabulario}),$$

donde cada entrada contiene el valor TF-IDF del término correspondiente.

## 4.2 LSTM

Como segundo modelo, proponemos una red recurrente con la arquitectura que podemos ver en la Figura 6.

Antes de ingresar el texto a al Red, se utilizó un módulo de python llamado Word2Vec para

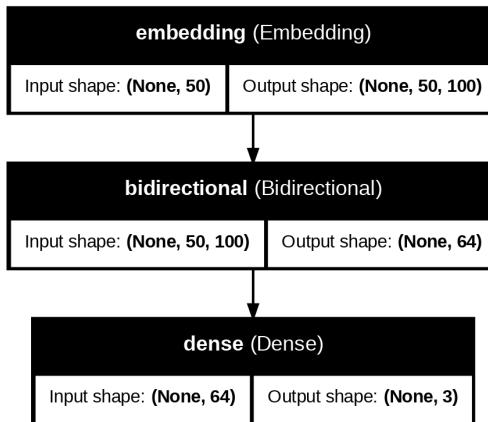


Figura 6: Arquitectura de la Red secuencial.

incializar los embeddings estos son una representación vectorial (de una longitud a elegir) para cada una de las palabras de nuestro corpus. El modelo SkipGram de Word2vec asume que una palabra puede ser usada para generar a las palabras a su alrededor. Técnicamente, el modelo *skip-gram* busca aprender representaciones vectoriales de palabras optimizando la probabilidad de predecir el contexto a partir de una palabra central. Sea  $w$  la palabra central y  $\{w_c\}$  el conjunto de palabras de contexto dentro de una ventana de tamaño  $m$ . El objetivo del modelo es maximizar

$$\prod_{w_c \in \text{contexto (en la ventana)}} P(w_c | w),$$

donde cada probabilidad se modela mediante una función softmax:

$$P(w_c | w) = \frac{\exp(v_{w_c}^\top u_w)}{\sum_{w' \in V} \exp(v_{w'}^\top u_w)},$$

con  $u_w$  la representación vectorial de la palabra central,  $v_{w_c}$  el vector de la palabra de contexto y  $V$  el índice del vocabulario (corpus). Dada una secuencia de texto de longitud  $T$ , donde la palabra en el instante  $t$  se denota por  $w^{(t)}$ , se asume que las palabras de contexto son generadas de manera independiente dado cualquier palabra central. Para una ventana de contexto de tamaño  $m$ , la función de verosimilitud del modelo *skip-gram* es la probabilidad de generar todas las palabras de contexto dado cualquier palabra central:

$$\prod_{t=1}^T \prod_{\substack{-m \leq j \leq m \\ j \neq 0}} P(w^{(t+j)} | w^{(t)}) ,$$

Durante el entrenamiento, los vectores  $u_w$  y  $v_{w_c}$  se actualizan mediante descenso estocástico del gradiente, minimizando la función de pérdida:

$$-\sum_{t=1}^T \sum_{\substack{-m \leq j \leq m \\ j \neq 0}} \log P(w^{(t+j)} | w^{(t)})$$

La matriz que se ocupa para los embeddings es la que está conformada por la representación cuándo tomamos las palabras como contexto i.e. los vectores  $v$ .

A continuación explicamos para qué sirve cada capa de la red recurrente de la Figura 6:

1. **Capa de embedding.** La primera capa del modelo es una capa *Embedding* que asigna a cada palabra del vocabulario un vector de dimensión `embedding_size = 100`. Es una matriz de pesos que se va actualizando con el Back Propagation.
2. **Capa Bidirectional LSTM.** Sobre la secuencia de embeddings regularizada se emplea una capa *Bidirectional LSTM* con 32 unidades por dirección. Esta capa procesa el texto tanto en el sentido temporal hacia adelante como hacia atrás, permitiendo capturar dependencias contextuales en ambas direcciones. Más adelante explicaremos más a detalle el funcionamiento de esta capa.
3. **Capa de salida densa con softmax.** Finalmente, se incluye una capa densa de 3 neuronas de salida (pues nuestro target consta de 3 valores diferentes) con función de activación *softmax*, que produce una distribución de probabilidad sobre las tres clases de sentimiento consideradas (negativo, neutral y positivo). El modelo se entrena utilizando la función de pérdida *categorical crossentropy* y el optimizador *Adam*.

Para entender el funcionamiento de una capa LSTM Bidireccional, debemos entender el funcionamiento de una capa LSTM. Podemos ver un gráfico de su arquitectura en la Figura 7. A grandes rasgos una capa LSTM funciona de la siguiente manera:

1. **Forget Gate ( $f$ ).** La puerta de olvido determina qué partes del estado de celda previo deben descartarse y cuáles deben conservarse. Esto lo podemos ver pues cada una de las entradas del vector que traemos de la memoria larga de la neurona anterior ( $c_{k-1}$ ) es multiplicada por un número entre 0 y 1 (nótese que se aplica la función sigmoide) i.e. estamos eligiendo qué porcentaje de cada entrada queremos conservar.

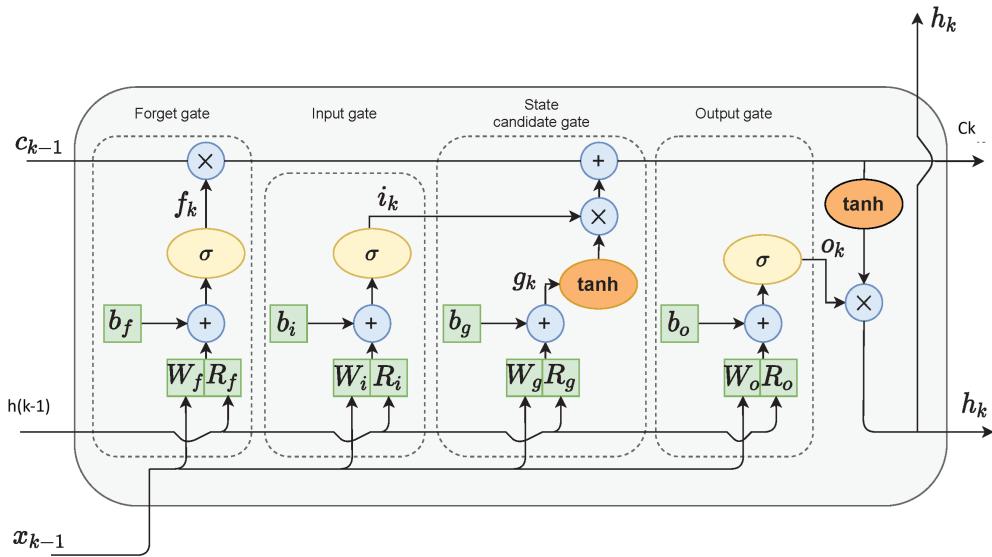


Figura 7: Gráfico de una LSTM

2. **Input Gate ( $i$ )**. Esta puerta asigna la proporción de información que será incorporada a la memoria de tiempo largo (nótese la función sigmoide). Esta información la da la puerta State candidate Gate.
3. **State candidate Gate ( $g$ )**. El candidato del estado de celda controla el flujo de información aplicando la función tanh al estado oculto previo y a la entrada actual. Este candidato se multiplica por la salida de la puerta de entrada para formar la contribución propuesta al nuevo estado de celda. Finalmente, este valor se suma a la información preservada de la memoria larga de la neurona anterior.
4. **Output Gate ( $o$ )**. Esta puerta determina qué proporción de la memoria larga que sale de la neurona debe ser la entrada de la memoria corta de la siguiente neurona. Una vez que se multiplica por la información propuesta para la memoria corta de la siguiente neurona, este resultado también es la salida de nuestra neurona (un vector).

A diferencia de una LSTM tradicional, que recorre la secuencia únicamente en un sentido, una Bi-LSTM procesa la entrada en las dos direcciones (adelante y atrás). Para ello emplea dos capas LSTM independientes: una que va del primer al último paso temporal y otra que va en sentido inverso. Podemos ver un diagrama de esta en la Figura (8). A continuación explicamos brevemente su funcionamiento:

1. **Paso hacia adelante (forward pass)**. La secuencia de entrada se introduce en la LSTM que avanza desde el primer hasta el último instante. En cada paso, la red actualiza su estado oculto y su celda de memoria considerando la entrada actual y los estados del paso anterior.
2. **Paso hacia atrás (backward pass)**. Paralelamente, la misma secuencia se procesa en orden inverso. La LSTM que recorre del último al primer paso temporal realiza los

mismos cálculos: actualiza su estado oculto y su memoria interna empleando la entrada del paso actual y los estados previos (en sentido inverso).

3. **Combinación de estados hacia adelante y hacia atrás.** Una vez completados los dos recorridos, los estados ocultos de ambas LSTM se combinan en cada paso temporal, estos se concatenan.

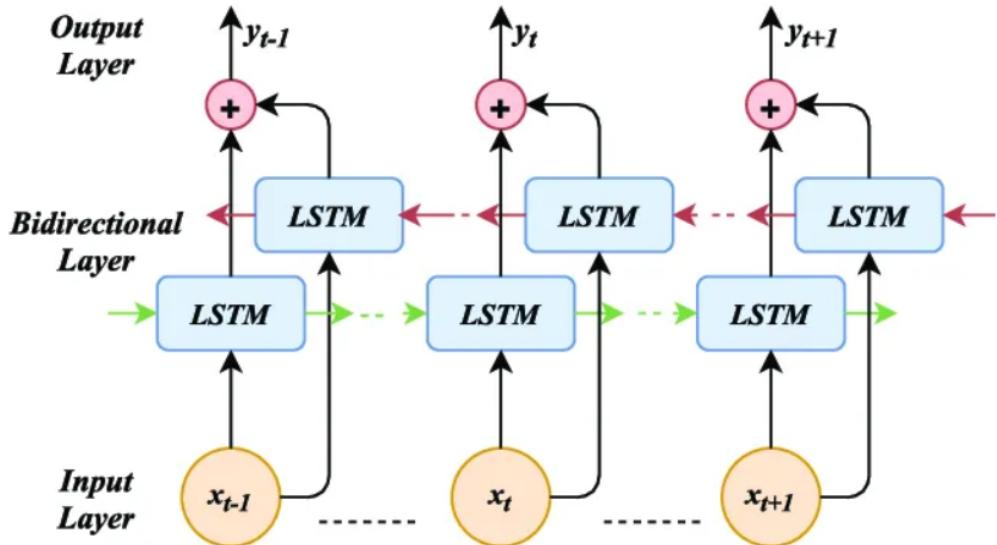


Figura 8: Gráfico representativo de una LSTM bidireccional.

## 5 Resultados

### 5.1 Evaluación del Modelo Random Forest

Cómo se comentó en la sección de Modelación, se usó cross validation con 3 folds para escoger el número de árboles, los resultados se pueden ver en la Figura 9. En esta gráfica, la línea azul es el promedio de la evaluación en los 3 folds mientras que la línea sombreada se construye apartir de la desviación estandar de las métricas obtenidas en cada fold. Podemos ver que el Accuracy promedio va subiendo hasta que llega a los 200 árboles, a partir de ahí baja, por lo que decidimos quedaron con el modelo que tiene 200 árboles.

Las métricas de este modelo se pueden ver en la Tabla 2.

El modelo Random Forest obtuvo un *accuracy* global del 75 % en el conjunto de validación. Sin embargo, al examinar las métricas por categoría se observan diferencias relevantes en su comportamiento.

La clase **negative** presenta el mejor rendimiento, con precisión, *recall* y F1-score iguales a 0.84. Esto muestra que el modelo clasifica de forma consistente los textos negativos, lo cual sugiere que esta categoría contiene patrones lingüísticos más claros y fáciles de identificar.

Por el contrario, la clase **neutral** es la más difícil de distinguir. Sus métricas son notablemente menores (precisión = 0.54, *recall* = 0.56 y F1-score = 0.55), reflejando que el modelo confunde con frecuencia los textos neutrales con otras categorías.

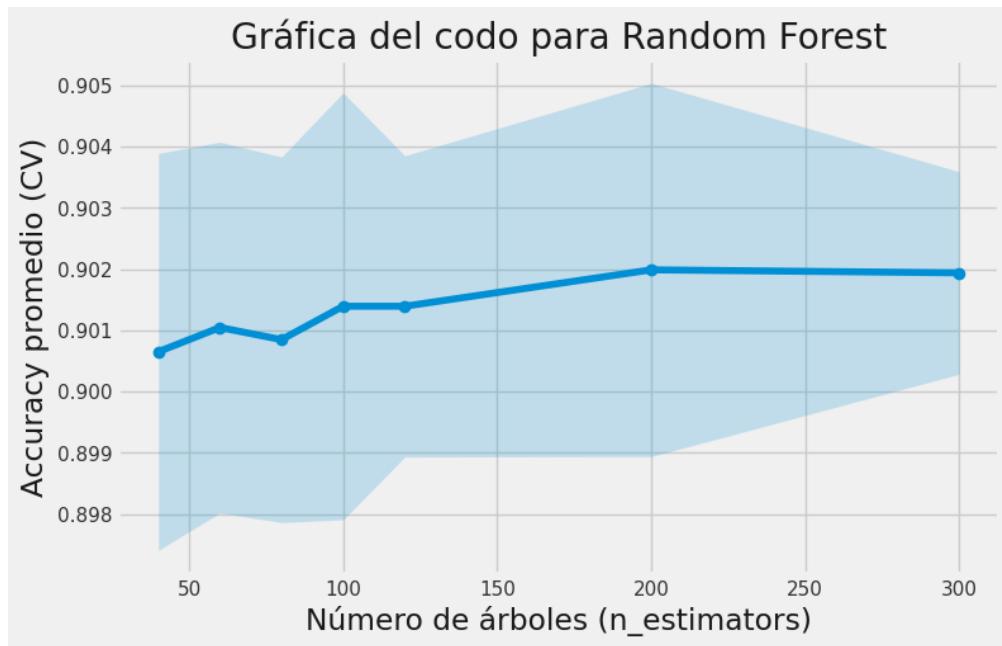


Figura 9: Gráfico del codo para escoger el número de árboles.

Clase	Precisión	Recall	F1-score	Soporte
negative	0.84	0.84	0.84	1680
neutral	0.54	0.56	0.55	553
positive	0.64	0.63	0.63	427
<b>accuracy</b>			0.75	2660
<b>macro avg</b>	0.68	0.68	0.68	2660
<b>weighted avg</b>	0.75	0.75	0.75	2660

Cuadro 2: Reporte de clasificación para el modelo Random Forest.

Las medias macro (0.68) y ponderada (0.75) muestran que, aunque el rendimiento general es aceptable, existe una variabilidad considerable entre clases. En conjunto, el modelo es robusto al clasificar textos negativos, competente para positivos y menos preciso en la detección de textos neutrales.

La matriz de confusión puede verse en la figura 10 el color azul fuerte prevalece en la diagonal de la matriz; sin embargo pareciera haber algo de confusión entre los tweets neutrales y negativos, pues en ocasiones los negativos se predicen como neutrales y viceversa. El Top 10 de variables más importantes para el Random Forest las podemos ver en la Figura 11.

## 5.2 Evaluación del Modelo LSTM

Se corrió la red Neuronal y se obtuvieron las métricas que se muestran en la Tabla 3. El modelo LSTM alcanzó un *accuracy* del 77% en el conjunto de validación, superando ligeramente al modelo basado en Random Forest.

La clase **negativo** muestra el desempeño más sólido, con precisión de 0.86, *recall* de 0.84 y un F1-score de 0.85, estas métricas nos indican que: De 100 predicciones que hace el

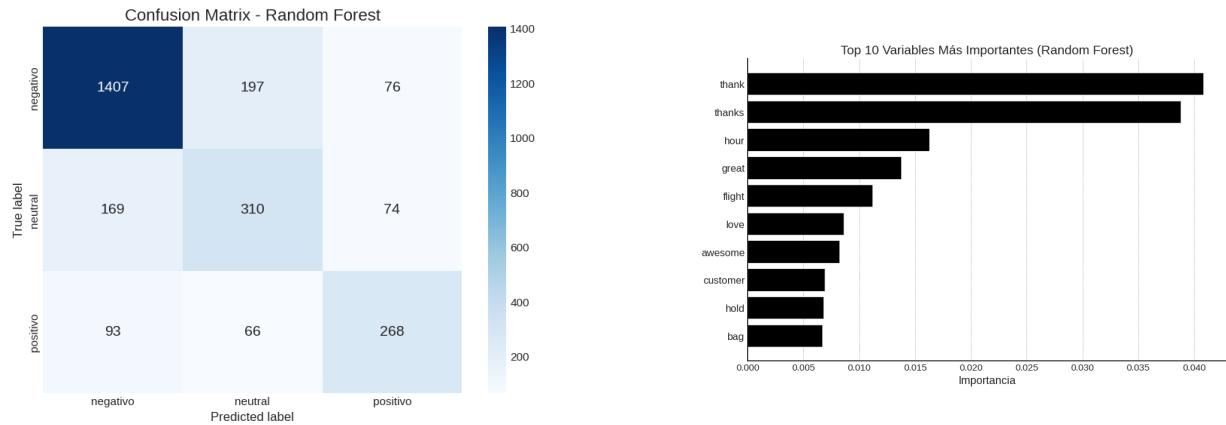


Figura 10: Matriz de confusión del Random Forest.

Figura 11: Variables más importantes Random Forest.

modelo asignando la clase negativa, 86 % de ellas realmente lo son. Del total de ejemplos en la clase negativa, nuestro modelo le atinó al 84 %. Finalmente, Recordemos que el *F1-score* es la media aritmética del Recall y la Precisión. Podemos ver que para el caso de las clases negativas, esta métrica es más grande que la correspondiente en el modelo Random Forest.

Para la clase **neutral**, las métricas mejoran ligeramente respecto al modelo clásico, obteniéndose una precisión de 0.58 y un recall de 0.59, podemos atribuirle a esta métrica la razón de que cuándo un comentario no es lo suficientemente negativo o suficientemente positivo cae aquí (precisión), también el desempeño de puede deber a que para esta clase en particular falta más variedad de tweets para poder distinguirla de mejor manera (recall). En general podríamos pensar que el desempeño se debe al desbalance de muestra pero después veremos que el Racall de la clase negativa no es tan baja (a pesar de ser la más pequeña) esto indica que el desbalance de muestra no es la razón.

La clase **positivo** presenta métricas intermedias, con un *F1-score* de 0.68 y un *recall* de 0.70, lo que indica una mejor capacidad de la LSTM para reconocer textos positivos en comparación con el modelo de árboles.

Las medias macro (0.70) y ponderada (0.77) reflejan un rendimiento global favorable y relativamente equilibrado entre categorías.

Clase	Precisión	Recall	<i>F1-score</i>	Soporte
negativo	0.86	0.84	0.85	1680
neutral	0.58	0.59	0.58	553
positivo	0.65	0.70	0.68	427
<b>accuracy</b>			0.77	2660
<b>macro avg</b>	0.70	0.71	0.70	2660
<b>weighted avg</b>	0.77	0.77	0.77	2660

Cuadro 3: Reporte de clasificación para el modelo LSTM.

En la Figura 12 se muestran las curvas de aprendizaje del modelo, donde se comparan la pérdida y el accuracy tanto en el conjunto de entrenamiento como en el de validación a lo

largo de las épocas. Las gráficas muestran el comportamiento de la función de pérdida y del accuracy durante las primeras épocas del entrenamiento del modelo LSTM. En la curva de pérdida, la *perdida del entrenamiento* disminuye rápidamente, lo cual indica que el modelo está aprendiendo a ajustar cada vez mejor los datos de entrenamiento. Sin embargo, la *perdida de validación* aumenta de forma continua desde la primera época. Este patrón es una señal clara de sobreajuste, pues el modelo mejora su rendimiento sobre los datos vistos, pero empeora al generalizar sobre datos nuevos.

Un comportamiento similar se observa en las curvas de Accuracy. El Accuracy del entrenamiento crece rápidamente y alcanza valores muy altos en pocas épocas, mientras que el Accuracy de validación se mantiene prácticamente estable o incluso desciende ligeramente. Esto refuerza la conclusión de que el modelo está memorizando los patrones del conjunto de entrenamiento en lugar de aprender representaciones generalizables.

Para abordar el problema de sobreajuste, se le colocó al entrenamiento un EarlyStopping el cuál hace que el entrenamiento se detenga si en las últimas 3 épocas la pérdida del conjunto de validación no ha disminuido. Así, obtenemos un modelo que no sobreajusta a los datos de entrenamiento. El mejor modelo resulta ser el obtenido en la primera época.

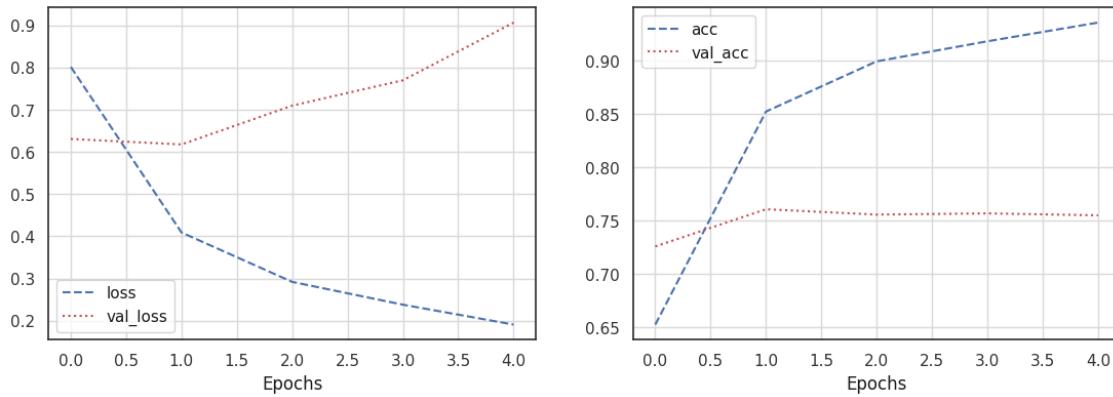


Figura 12: Curvas de aprendizaje del modelo: pérdida y exactitud en entrenamiento y validación.

En la Figura 13 podemos apreciar la matriz de confusión del modelo, vemos que el color azul fuerte prevalece en la diagonal de la matriz; sin embargo pareciera haber algo de confusión entre los tweets neutrales y negativos, pues en ocasiones los negativos se predicen como neutrales y viceversa.

En conjunto, el modelo LSTM demuestra ser más competente que el Random Forest para capturar dinámicas secuenciales del lenguaje y mejorar la clasificación tanto de textos positivos como neutrales, manteniendo un aceptable desempeño en la detección de textos negativos.

## 6 Prueba en Nuevos Tweets

Basándonos en los comentarios sobre los modelos anteriores, decidimos quedarnos con la red recurrente. Así, en la Figura 14 podemos ver la predicción de la porporción de sentimiento

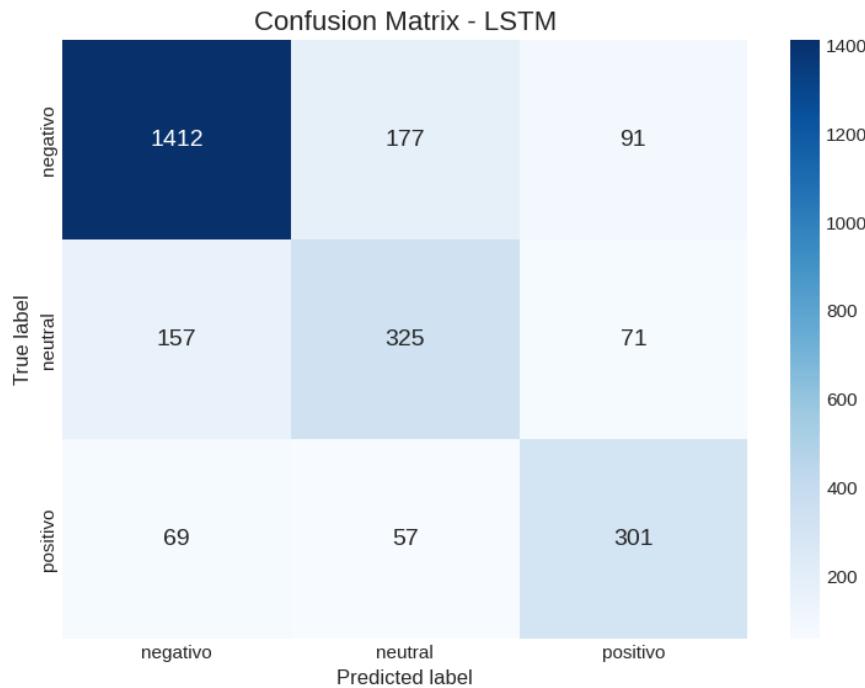


Figura 13: Matriz de Confusión del Modelo.

de los tweets por aerolínea en el conjunto de testeo.

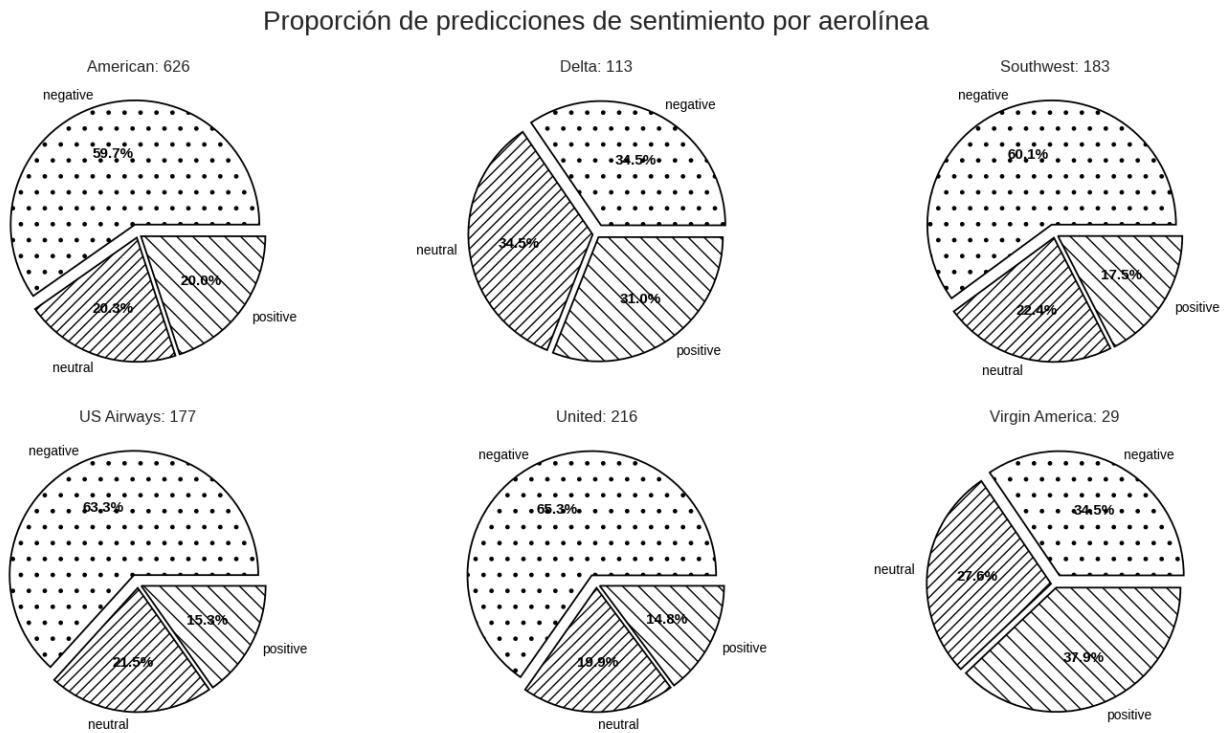


Figura 14: Sentimiento por Aerolínea predicho.

Tiene sentido pues preserva aproximadamente las mismas proporciones que veíamos en el conjunto de entrenamiento. A partir de este gráfico las Aerolíneas pueden saber cómo está su reputación para cualquier periodo de tiempo dado. Esto puede llegar a ser muy útil después de implementar políticas nuevas, anuncios sobre el servicio al cliente y estrategias comerciales. En la práctica, de poseer información diaria, o semanal podríamos ir calculando KPI's sobre la reputación de la Aerolínea en función de las predicciones y visualizar series de tiempo para analizar su evolución. En la Figura 15 podemos ver una muestra de cómo predice nuestro modelo.

	text	true_sentiment	pred_sentiment
	@VirginAmerica What @dhepburn said.	1	1
	@VirginAmerica plus you've added commercials to the experience... tacky.	2	1
	@VirginAmerica I didn't today... Must mean I need to take another trip!	1	1
	@VirginAmerica it's really aggressive to blast obnoxious "entertainment" in your guests' faces & they have little recourse	0	0
	@VirginAmerica and it's a really big bad thing about it	0	2
	...	...	...
	@AmericanAir WORST SERVICE EVER!! Delayed flights for more than 5 hours plus you missed my bag! And your employees are rude 😦 😦	0	0
	@AmericanAir thanks for the generic computer generated response. How about you accommodate your travelers instead of just saying sorry	0	0
	@AmericanAir they are giving cots to the people that did not get hotel rooms... that is terrible..	0	0
	@AmericanAir I did twice got a letter back saying that your company "doesn't issue refunds for phone bills" helpline will be shut off	0	0
	@AmericanAir oh, yeah. I guess those are two different things. 3 am does weird things to my brain. Thanks again! xox	1	2

Figura 15: Muestra de cómo predice el modelo.

## 7 Referencias

- Rivera, Mariano. *Introducción a Redes Neuronales Recurrentes (RNN)*. Noviembre 2018.
- **Word2Vec.** Dive into Deep Learning. Disponible en: [https://d2l.ai/chapter\\_natural-language-processing-pretraining/word2vec.html](https://d2l.ai/chapter_natural-language-processing-pretraining/word2vec.html)
- Lim, Seungbum. *Twitter Sentiment Analysis — EDA and ML/DL*. Disponible en: <https://www.kaggle.com/code/seungbumlim/twitter-sentiment-analysis-eda-and-ml-dl>.
- *DL: Introduction to Deep Learning*. Temas Selectos en Ciencia de Datos: Búsqueda de Arquitecturas Neuronales. Centro de Investigación en Matemáticas A.C., s.f.
- Aquino López, Marco Antonio. *Introducción a la Ciencia de Datos*. Presentación 11 del curso de la Maestría en Probabilidad y Estadística, Centro de Investigación en Matemáticas A.C., Agosto–Diciembre 2025.