Asistencia para la categorización de documentos

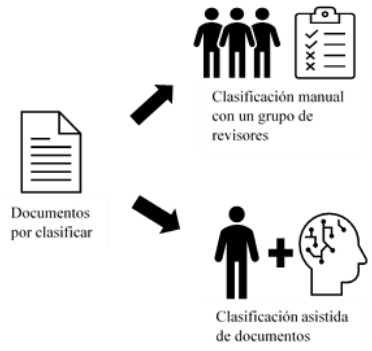
Álvaro Moncada – alvaro.moncada @javeriana.edu.co
Álvaro Pérez – perez.alvaro @javeriana.edu.co
Edwin Turizo – edwin.turizo@javeriana.edu.co

Resumen. El presente documento tiene como intención abordar la problemática de la categorización de documentos mediante el uso de redes neuronales (RN). Puntualmente, el objetivo consiste en entrenar una red neuronal para que asista en la clasificación de documentos enmarcado en un sistema inteligente basado en agentes racionales. Dado el alcance de este documento, se propone una solución para la categorización de noticias utilizando redes neuronales recurrentes y adicionalmente se tendrán procesos de tratamiento de los datos de entrada como *tokenización* y *stop-words*. El set de datos utilizado consta de 2225 noticias de la BBC clasificadas en 5 categorías para finalmente determinar el desempeño del modelo frente a la problemática abordada.

Palabras Clave: Clasificación de documentos, Redes Neuronales, Text Classification, RNN.

I. Contextualización del problema

El descubrimiento de evidencia digital o E-Discovery, permite a investigadores o abogados identificar información almacenada electrónicamente relacionada con el objetivo de un caso. Luego de identificar aquellos archivos relevantes mediante el uso de palabras clave inicia la etapa de revisión que, usualmente, consume muchos recursos y toma mucho tiempo debido al gran volumen de información a revisar. Por lo que es relevante la implementación de técnicas que reduzcan estos tiempos y permitan a los revisores enfocar sus esfuerzos para identificar la mayor cantidad de información relevante ágilmente.



- Incremento de costos debido al alto número de revisores involucrados
- Posibles diferencias en los criterios de clasificación

- Disminución de cantidad de personas involucradas y menor costo asociado
- Homogeneidad en los criterios de clasificación

Ilustración 1. Comparación de modalidades para la revisión de documentos

*. https://www.kaggle.com/c/learn-ai-bbc/data

La **clasificación de documentos** es una problemática que ha sido ampliamente estudiada en la rama de la minería de texto dadas sus amplias aplicaciones. Esta problemática es definida de la siguiente manera. A partir de un **set de documentos de entrenamiento** $D=\{X_1, X_2,\ldots, X_n\}$, cada uno de los cuales cuenta con una "**etiqueta**" que especifica su clasificación dentro de un conjunto de valores {1, …, k}. Los documentos de entrenamiento son utilizados para construir un **modelo de clasificación**, el cual relaciona **los descriptores** *(features)* de un documento con una de las etiquetas. En un caso de prueba, el modelo de entrenamiento es usado para predecir su clasificación, esta clasificación puede ser de dos maneras *hard* y *soft*, en la primera se especifica una clasificación puntual al documento, mientras que en el segundo tipo se asigna una combinación probabilística de diferentes clasificaciones al documento.[1]

II. Análisis del estado del arte

Dado nuestro alcance, nos enfocaremos en indagar el estado del arte para soluciones relacionadas con la clasificación de noticias o de cualquier texto en general. Este tipo de soluciones constan de dos (2) etapas principales, una etapa inicial de alistamiento de los datos que consta de un **preprocesamiento y limpieza del dataset** y un posterior entrenamiento de los **algoritmos utilizados** para la clasificación de documentos [1], dado el alcance de este documento, se limitará a redes neuronales.

En un primer acercamiento, Samir Endavia *et al. u*tilizaron redes neuronales convolucionales (CNN, por sus siglas en inglés) y las redes neuronales recurrentes (RNN, por sus siglas en inglés) GRU y LSTM para la clasificación de documentos legales, e incluso proponen una solución basada en estos métodos llamado *Supreme Court Classifier*. Sin embargo, su solución estaba orientada a la clasificación en 15 categorías generales y 279 específicas, que difiere de nuestra aplicación en la que nos enfocaremos en cinco (5) categorías generales. Para el alistamiento de los datos, compararon la efectividad de sus redes neuronales utilizando tres (3) técnicas: word2vec, fastText y GloVe, todas estas corresponden a técnicas de w*ord embedding,* es decir la representación de palabras, usualmente en forma de vectores, para análisis de texto, a diferencia nosotros utilizaremos Tokenización y técnicas de limpieza como quitar puntuación, números y Stop words. Sus resultados muestran que las mejores soluciones para las categorías generales están CNN + word2vec y GRU + word2vec, con 72,4% y 68,6% de precisión respectivamente [2].

Luego, Haojin Hu *et al.* brindan una solución utilizando RNN, específicamente Independently Recurrent Neural Network (IndRNN) combinado con LSTM con modelo de atención, esta combinación favorece la solución del problema de desvanecimiento del gradiente. Para entrenar a su solución utilizaron un set de tentreamiento de 560.000 items de DBpedia y una muestra de prueba de 70.000, que para

nuestro caso el data set es de 2225 noticias de la BBC clasificadas en 5 categorías, y utilizamos una distribución de 80 – 20, entrenamiento y prueba, respectivamente. Como parte de su experimento compararon con 5 tipos de redes neuronales, a saber, Attention-Based Bidirectional Long Short-Term Memory Networks (AttBLSTM), Hierarchical Attention Network (H-ATT), Adversarial Training Methods for Semi-Supervised Text Classification (ADTM), Convolutional Neural Networks (CNN), Random Multi-model Deep Learning (RMDL), encontrando que los mejores resultados los obtuvo el modelo RMDL (98.82 %), y en segundo lugar se encontraba su solución IndRNN-LSTM (98.51%) [3].

Por otro lado, para la fase de alistamiento de la información, vemos diferentes aproximaciones a esta etapa. La extracción de descriptores (*features*) para la clasificación de texto es especialmente importante para la clasificación de texto, ya que tiene esta problemática posee dos características relevantes, su alta dimensionalidad y la presencia de palabras "ruidosas", es decir que son frecuentes, pero no aportan a la tarea de clasificación. En general un texto puede ser representado de dos maneras. La primera es llamada *bag-of-words,* en la que un documento es representado como un set de palabras junto con su frecuencia asociada en el documento. Este tipo de representación es independiente del orden de las palabras. La segunda manera consiste en representar el texto como *strings,* en este caso se tiene en cuenta la secuencia de las palabras.[1]

Una de las maneras más comunes de extracción de descriptores tanto para modelos supervisados como no supervisados es el descarte de *stop-words* y *stemming.* El primero consiste en la exclusión de palabras que no son específicas para las diferentes categorías analizadas. Por otro lado, en *stemming* las diferentes formas de una palabra son consolidadas en una sola. Para nuestro caso, vamos a utilizar *stop-words*, junto con *tokenization* el cual consiste en reducir un texto a unidades individuales, que para nuestra aplicación serán palabras. [1]

III. Metodología

CRISP DM

A continuación, se presenta una descripción de cada fase lo realizado dentro de cada subproceso que abarca la metodología CRIPS-DM 1.0 [5]:

1. **Entendimiento del negocio**
   En esta fase se busca establecer las metas y objetivos a cumplir con el proyecto de inteligencia artificial basado en los métodos de redes neuronales. En este caso lo que se busca es clasificar noticias en un conjunto de categorías ya definidas con el fin de automatizar el proceso manual a las personas interesadas.
2. **Entendimiento de los datos**
   En esta fase se busca analizar con profundidad el *dataset* a utilizar, identificar la pertinencia de este, y los atributos y características propios, esto es,

hacer un barrido sí se cumplen ciertos criterios definidos para lo que se necesita dentro del problema y con esto, la calidad con que vienen para su posterior utilización.

Nuestra aplicación al estar relacionada con el tratamiento de texto tiene como características que es data no estructurada y tiene una alta dimensionalidad y es disperso, es decir que no todas las palabras van a aparecer en todos los documentos. Así mismo, La calidad de la data es aceptable dado que no se identificaron campos faltantes o caracteres no legibles.

3. **Preparación de los datos**
   Vamos a utilizar una *dataset* de 2225 noticias de la BBC distribuidas en 5 categorías, a saber, Deporte, Entretenimiento, Negocio, Tecnología y Política, así mismo la base cuenta con tres columnas, un identificador, el texto de la noticia y una columna con la clasificación de la noticia. En esta fase se realizó la unión de los datos de entrenamiento y de prueba, luego, se hizo el correcto preprocesamiento de los datos, por medio de las técnicas de limpieza y tokenización con el fin de representar el texto de la noticia en valores numéricos que pueda interpretar la red neuronal.
4. **Modelado**
   En consiste en la aplicación de la técnica a utilizar, en este caso redes neuronales recurrentes, para esto se realizó un modelado de las capas teniendo en cuenta el número de neuronas, funciones de activación y la configuración de los hiperparametros. El modelado abarca toma de decisiones importantes, tales como los métodos de optimización, funciones de perdida y épocas de entrenamiento, que tienen una alta incidencia en los resultados.
5. **Evaluación**
   En esta fase se busca evaluar el desempeño del modelo implementado por medio de los valores de error obtenidos durante el entrenamiento y testeo en las predicciones de valores distintos a los conocidos. Esta fase permite tomar decisiones de mejoras en el modelo y reentrenamiento gracias a la realización de análisis en base a graficas que dan una mejor interpretación de los datos
6. **Despliegue**
   En esta fase se busca llevar el modelo a producción de forma que utilice el modelo con datos reales, sin embargo, al ser un proyecto de investigación en el que lo que se busca es llevar a cabo la aplicación y experimentación de cada una de las técnicas principales en los distintos módulos, los resultados son presentados por medio de los distintos papers publicados.

IV. Modelo para la aplicación de los conceptos y algoritmo de la técnica IA en el caso desarrollado

Se implementó una red neuronal recurrente (RNN) utilizando TensorFlow 2, Keras y la librería "NLTK", esta última es utilizada para el procesamiento natural del lenguaje (NLP). A continuación, se presenta el detalle de la implementación.

Preprocesamiento

Como se ha mencionado anteriormente, se utilizó un *dataset* de 2225 noticias de la BBC que se encuentran clasificadas en 5 categorías. Antes, de poder analizar los datos en nuestra red escogida, es importante realizar un preprocesamiento del texto de las noticias, el cual permita una mejor interpretación de los datos. Para esto, inicialmente se realizó un proceso limpieza del texto utilizando la librería **NLTK**, en el que se **descartaban símbolos, signos de puntuación y números**, luego se descartan las ***stop-words,*** y finalmente para obtener una matriz con el mismo tamaño, se realizó un ***padding*** de las palabras obtenidas por cada documento y se limitó este tamaño del vector a las **4500 palabras más frecuentes**, dada la alta dimensionalidad que se presentaba (25.483 posibles palabras) y el alto tiempo que este tipo de soluciones requiere para su procesamiento. Estos procesos daban como resultado una matriz de entrada de **2225 x 4500.**

Red neuronal

A partir de la revisión del estado del arte, se identificó que una de las redes neuronales con buen comportamiento en la clasificación de texto son las Redes Neuronales Recurrentes (RNN, por sus siglas en inglés). Bingyuan Wang *et al.* utilizan RNN para la clasificación de texto utilizando Bi-LTSM y Bi-GRU, aquí muestran que la diferencia en la precisión entre estas dos es de aproximadamente del 2%, a partir de un *learning rate* de 0.01 y alrededor de 3 o 4 épocas de entrenamiento [6]. En base a esto vamos a implementar una **RNN Bi-LSTM** con parámetros similares que se adapten a los requerimientos de nuestra aplicación.

Luego, se definió que se realizaría el **entrenamiento-prueba con 80-20**, respectivamente y se entrenará mediante la **disminución del gradiente** con un ***learning rate*** de 0.01 por 5 **epocas**. En base a la información recopilada, se definió probar con una RNN Bi-LSTM implementada cuenta con **5 capas**. Dos de las cuales son densas y cuentan con **funciones de activación** de *Tanh* y *Softmax respectivamente,* esta última permite decir si una noticia pertenece a una de las 5 posibles clasificaciones mediante un vector codificado de 1s y 0s. Por otra parte, se aplicó un ***dropout*** **de 0,5** para las capas densas para disminuir el *overfitting* de la red. A continuación, se presentan unas gráficas que ilustran la RNN Bi-LSTM implementada.

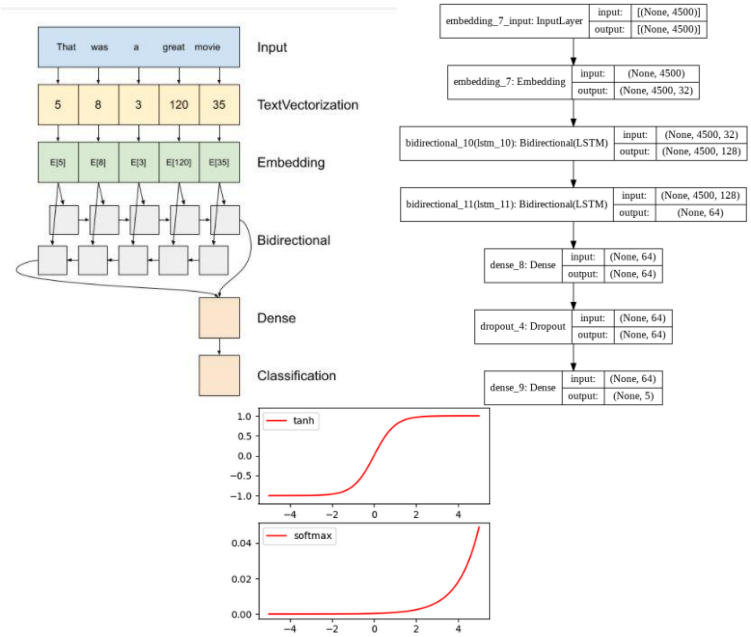


Ilustración 2. Estructura Bi-LSTM

V. Protocolo experimental y análisis de los resultados obtenidos

Como se introdujo en la sección anterior, nuestro modelo base esta parametrizado de la siguiente manera:

- *Epochs*: 5,
- *Learning Rate*: 0.01,
- Neuronas por capa: (64, 32, 64, 5),
- Dropout: 0,5
- Entrenamiento-Test: 80-20.
- Aproximadamente 10 minutos de entrenamiento por época.

A continuación, se presentan los resultados obtenidos a partir de la variación de las variables independientes, factores controlables y la respectiva respuesta de nuestra red neuronal.

Variación *epochs* red neuronal:

Epochs	Accuracy	ValidationAccuracy	Loss	ValidationLoss
5	96.40%	49.66%	13.26%	215.76%
7	97.47%	48.76%	8.11%	263.12%
12	97.13%	50.11%	8.19%	240.05%

Como se puede apreciar en la tabla anterior, a medida que se le coloque un mayor número de épocas de entrenamiento, el algoritmo se vuelve más preciso en cuanto tiene más iteraciones de aprendizaje para los datos tanto de entrenamiento, como de validación, sin embargo, también se observa que el "*validation accuracy*" tiene ciertos momentos de subida y bajada, que se deben a sobreajuste que puede estar presentando la red neuronal. Con respecto a los valores de perdida, lo que se busca es minimizar este valor de forma que no le "cueste" tanto a la red mejorar su desempeño. En cuanto al valor de "*loss*" si se ve una disminución a lo largo de las épocas de entrenamiento, y, por el contrario, en el "*validation loss*" se destaca un aumento en los valores de perdida, lo que muestra que tiene un mayor error con los datos de validación.

Variación del *learning rate:* Al variar el *Learning rate* se obtuvo el siguiente comportamiento de la RNN LSTM:

Learning rate	Accuracy	ValidationAccuracy	Loss	ValidationLoss
0.5	20.06%	16.85%	341.00%	338.62%
0.1	19.78%	22.70%	202.47%	177.85%
0.01	96.40%	49.66%	13.26%	215.76%
0.005	96.93%	53.03%	12.76%	190.48%

Como se puede apreciar en la tabla anterior, a medida que se le coloque una mayor tasa de aprendizaje, el algoritmo pierde precisión puesto que existen casos en los que no converge a una correcta solución dando grandes saltos en los cambios de los coeficientes y pesos neuronales.

Variación cantidad de neuronas*:* Al variar la cantidad de neuronas se obtuvo el siguiente comportamiento de la RNN LSTM:

Cantidad Neuronas	Accuracy	ValidationAccuracy	Loss	ValidationLoss
128 LSTM 64 LSTM 128 Densa	79.04%	44.04%	65.21%	40.00%
64 LSTM 32 LSTM 64 Densa	96.40%	49.66%	13.26%	215.76%

Cantidad Neuronas	**Accuracy**	**ValidationAccuracy**	**Loss**	**ValidationLoss**
24 LSTM 16 LSTM 24 Densa	95.11%	53.93%	19.72%	197.71%

Se evidencia que el número de neuronas hace que los resultados cambien de forma que cada neurona da peso a los parámetros que se limitan, en este caso las palabras y que se tengan diferencias en el *"accuracy"* y el *"loss"* y adicionalmente tiempos de entrenamiento mayores en los distintos casos.

Aplicación de *stopwords* en el preprocesamiento de los datos

Stop words	**Accuracy**	**ValidationAccuracy**	**Loss**	**ValidationLoss**
Sin aplicar	21.74%	22.70%	161.15%	160.78%
Aplicando	96.40%	53.26 %	11.76 %	221.07%

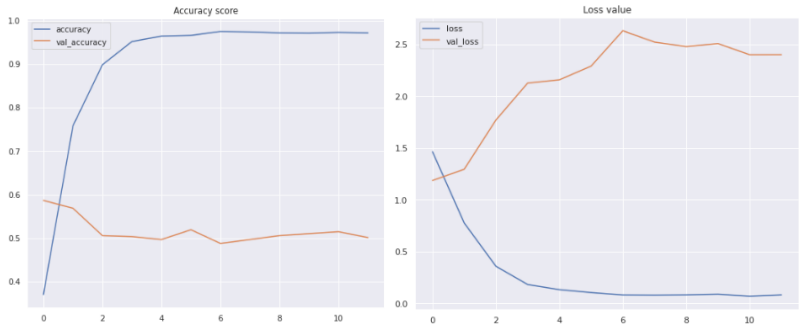
Por otro lado, el efecto de no aplicar *stop words* en la precisión de nuestro sistema es considerable, ya que la precisión y los valores de perdida disminuyen altamente, y afectan el rendimiento de la red neuronal, al no tenerse datos correctos y que permitan un mejor análisis con palabras relevantes en la categorización:

Variación tasa de *split* de los datos de entrenamiento y validación:

Split rate	**Accuracy**	**ValidationAccuracy**	**Loss**	**ValidationLoss**
train:80%6 test: 20%	96.40%	49.66%	13.26%	215.76%
train:70% test: 30%	96.34%	51.35%	14.17%	199.79%

A partir de estos resultados podemos ver que la cantidad de datos de entrenamiento y de prueba, resulta ser una variable importante en base a la tabla anterior que muestra que al tener un mayor número de datos de validación mejora el "*validation accuracy"* y también el "*validation loss*", manteniendo un valor similar el "*accuracy*" en general.

A continuación, se presentan las gráficas de "accuracy score" y de "loss score":



En general, se tuvo un creciente aumento en los valores del "accuracy" a lo largo de las épocas de entrenamiento en que se definió el modelo base, obteniéndose un valor final de 96.40% y ya con la curva del "validation accuracy" se comportó de forma decreciente, en donde se tuvo un valor final de 49.66%. Con respecto a la gráfica de los valores de perdida, el resultado de la curva de "loss" fue decreciente, minimizándose la perdida en los datos de entrenamiento, y ya en el "validation loss" se generando picos de subida y bajada, con una tendencia general a la subida.

Ilustración 3. Accuracy Score y Loss Score

Por último, se generó la matriz de confusión con los datos de validación:

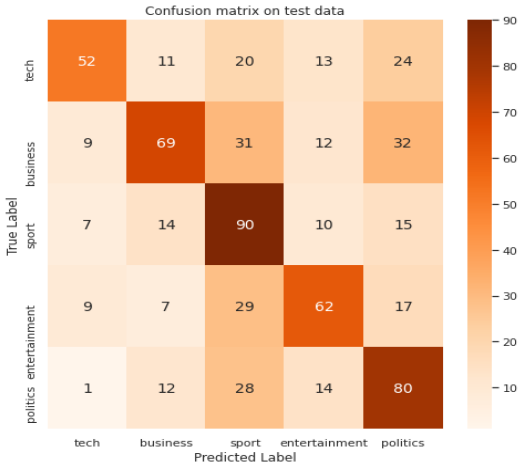


Ilustración 4. Matriz de confusión datos de validación

Esta matriz muestra la cantidad de noticias del conjunto de validación que fueron clasificadas por la red en las categorías definidas, y su correspondiente etiqueta del dataset. Se observa que las noticias de tecnología fueron las que tuvieron un mayor número de predicciones correctas con un 66%,, en cambio las noticias deportivas fueron con las que peor se desenvolvió la red, teniendo un valor de predicciones correctas de 45%.

VI. Conclusiones

- El modelo de redes neuronales recurrentes se ajusta a la problemática abordada ya que su componente de memoria temporal facilita que esta tenga en cuenta el contexto del texto basándose en los patrones vistos y el orden secuencial de las palabras.
- Durante el desarrollo del modelo, se probaron redes neuronales estándar y convolucionales, sin embargo, la que presentó mejor comportamiento fue la RNN Bi-LSTM, dadas las restricciones de nuestra problemática.
- Se evidencia la importancia de implementar stop-words como parte del preprocesamiento puesto que en una de las pruebas realizadas en donde se omitió este paso el desempeño de la red se vio altamente afectado.
- Los tiempos de entrenamiento de la red fueron de aproximadamente 10 minutos por época, pero si el *learning rate* disminuía, estos tiempos incrementaban.
- Por otro lado, al disminuir la proporción de la cantidad de noticias de entrenamiento a 70-30, la red presentó una mayor precisión para la clasificación, lo que nos lleva a pensar que con la proporción de 80-20 se sobre ajusta el modelo a una alta cantidad de datos de entrenamiento.
- La red se comportó mejor para la clasificación de noticias de tecnología en un 66% de los casos mientras que clasificó con menor éxito las noticias de política.
- Para futuros desarrollos se recomienda aumentar el *dataset* de trabajo, ya que como se identificó durante el análisis del estado del arte, los *dataset* utilizados eran de por lo menos 50.000 *items*.

VII. Bibliografía

[1] Charu C. Aggarwal, ChengXiang Zhai, “Mining Text Data”. Springer 2012.

[2] Samir Undavia, Adam Meyers, John E. Ortega, “A Comparative Study of Classifying Legal Documents with Neural Networks” IEEE Xplore 2018.

[3] Haojin Hu , Mengfan Liao , Chao Zhang , Yanmei Jing, “Text classification based recurrent neural network”. IEEE Xplore 2020.

[4] Yu Meng, Jiaming Shen, Chao Zhang, Jiawei Han, “Weakly-Supervised Neural Text Classification”, ACM 2018.

[5] CRISP-DM, SPSS Inc. 2000

[6] Bingyuan Wang, Fang Miao, Xueting Wang, Libiao Jin , “Text Classification Using a Bidirectional Recurrent Neural Network with an Attention Mechanism” IEEE Xplore 2020