



Pontificia Universidad
JAVERIANA
Bogotá

Proyecto 1 Tópicos Avanzados de Analítica

Entregado por:

Juan David Villate Lemus

Laura Katherine Moreno Giraldo

William Ricardo Fernández Garnica

José Rafael Peña Gutiérrez

PONTIFICIA UNIVERSIDAD JAVERIANA

Marzo 2024

Adquisición de datos

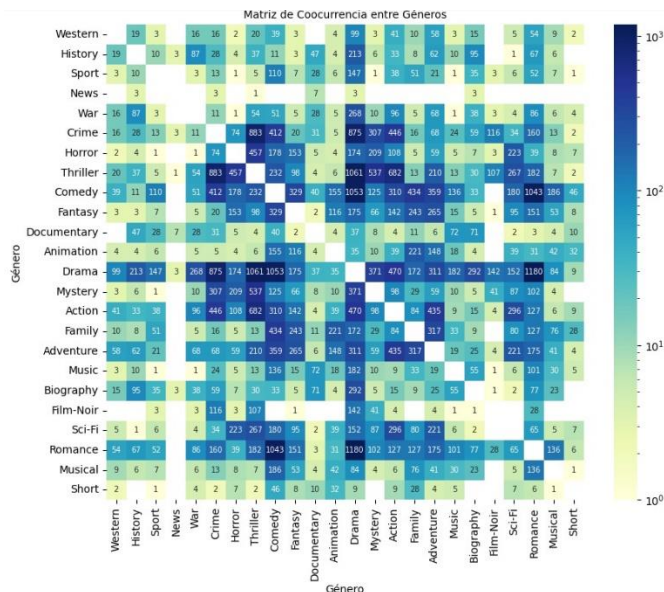
El objetivo del proyecto será clasificar los géneros de películas, usando procesamiento de lenguaje natural para extraer información relevante de la trama de la película. El conjunto de datos contiene información del año de estreno, título, trama, rating y géneros de 7895 películas.

Los datos fueron obtenidos del Profesor Fabio González, PhD y su estudiante John Arévalo. Revisar <https://arxiv.org/abs/1702.01992>.

También se utilizaron las siguientes fuentes para realizar una exploración adicional para este proyecto. (Agarwal, 2020), (NS, 2019) (Paul Matthews, 2020)

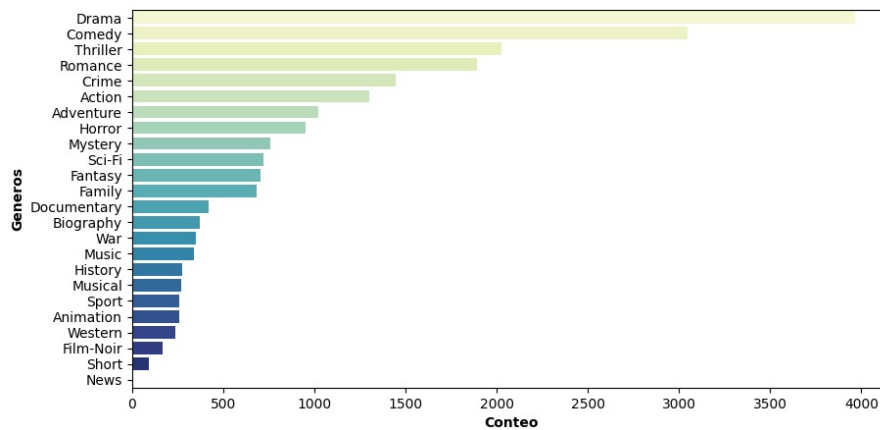
Análisis descriptivo

Es importante explorar los datos y reconocer patrones que faciliten y respalden el procesamiento y modelado de los mismos. En la matriz de coocurrencia de géneros representada en la gráfica 1, se puede observar que el género "drama" presenta una alta frecuencia de coocurrencia con otros géneros. En particular, se destaca la alta incidencia de "drama" junto a géneros como romance, thriller y comedia. También destacan pares como comedia y romance, que muestran una coocurrencia significativa.



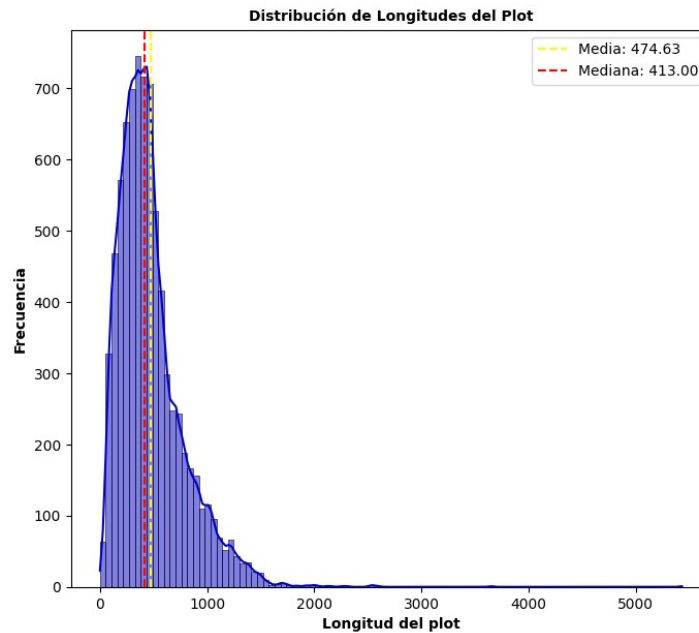
Gráfica 1. Matriz de coocurrencia entre géneros

La matriz de coocurrencia sugiere que los pares mencionados anteriormente son probablemente los géneros más frecuentes en el conjunto de datos. La Gráfica 2 muestra la frecuencia de cada género en el conjunto de datos agregado, donde los géneros "drama" y "comedia" destacan por su alta presencia.



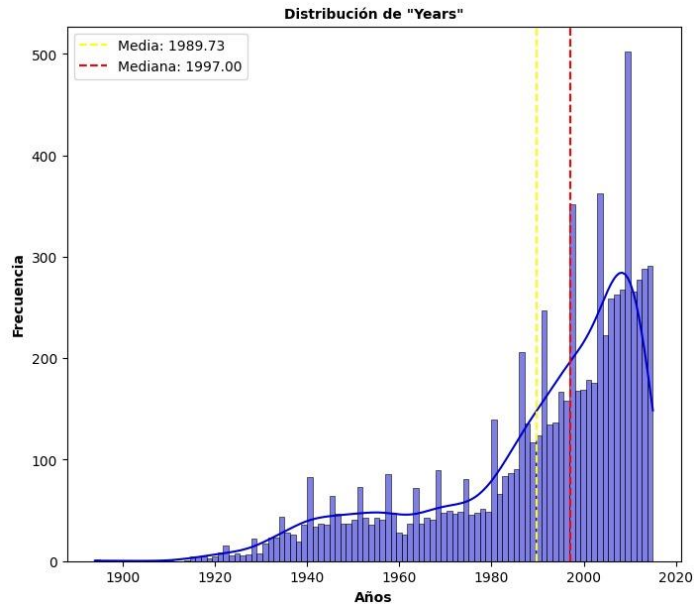
Gráfica 2. Frecuencia por género

En la Gráfica 3 se puede observar que la longitud media de la trama es de aproximadamente 475 palabras. Aunque existen películas con tramas que sobrepasan la barrera de las mil palabras, la frecuencia relativa de estas es muy baja.



Gráfica 3. Distribución de longitud tramas

Por último, en la Gráfica 4 se presentan la distribución de los años de estreno de las películas en el conjunto de dato. Se puede observar que a medida que incrementan los años, se presenta mayor número de películas. Aunque la media y mediana están por debajo del siglo XXI, la moda se presenta posterior a este año alrededor de 2015.



Gráfica 4. Distribución de la variable año de estreno

Procesamiento de datos

Limpieza

Se identificaron algunas películas que contenían la trama en un idioma diferente al inglés, idioma principal del análisis, la presencia de otro idioma puede generar ruido para el modelo.

```
# Se borran algunas instancias que poseen la trama en un idioma diferente al
de estudio (Inglés)
indices_to_delete= [6131, 752, 2750, 4429, 5479, 8281]
plot_to_delete = dataTraining.loc[indices_to_delete, 'plot']
print(plot_to_delete)

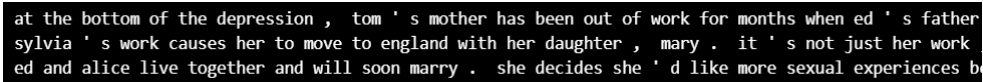
dataTraining= dataTraining.drop(index= indices_to_delete)
```

Bloque de código 1. Limpieza de tramas en otro idioma

Otro paso importante en la limpieza de los datos fue la corrección de caracteres especiales. En este proceso, se realizaron dos reemplazos significativos:

1. Se modificaron los caracteres que contenían signos ortográficos no propios del inglés, como la diéresis, la tilde diacrítica, la cedilla, entre otros, por sus equivalentes sin estos signos. Por ejemplo, la palabra "fiancé", de origen francés y comúnmente utilizada en inglés, fue reemplazada por su versión en inglés sin la diéresis.
2. Se eliminaron los espacios a la izquierda y derecha de las comillas simples porque se identificó que en las contracciones como 've, 's, entre otras, existía un espacio en ambos lados de la comilla simple, lo cual impediría que, con cualquier estrategia de vectorización, el modelo lograra interpretar las contracciones en el texto.

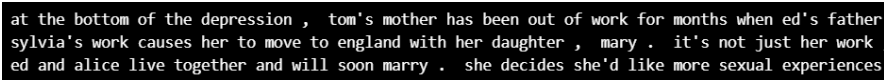
En la Imagen 1 se puede observar algunas de las tramas que tienen este problema de contracciones, donde la comilla simple está separada de la palabra madre y de los caracteres posteriores que componen la contracción



```
at the bottom of the depression , tom ' s mother has been out of work for months when ed ' s father
sylvia ' s work causes her to move to england with her daughter , mary . it ' s not just her work
ed and alice live together and will soon marry . she decides she ' d like more sexual experiences b
```

Imagen 1. Tramas con problema de contracciones

Luego de aplicar la limpieza, las dos correcciones mencionadas son aplicadas a todo el conjunto. En la Imagen 2, se presentan las mismas observaciones, pero posterior procesamiento, donde se puede observar que las contracciones ahora sí están bien escritas y se pueden representar si así se desea.



```
at the bottom of the depression , tom's mother has been out of work for months when ed's father
sylvia's work causes her to move to england with her daughter , mary . it's not just her work
ed and alice live together and will soon marry . she decides she'd like more sexual experiences
```

Imagen 2. Tramas con problemas de contracciones corregidas

Preprocesamiento

Posterior a la limpieza de los datos, se optó por algunos procesamientos generales en los problemas de lenguaje natural. El Bloque de código 2 muestra que se incluyó lematización y se extrajeron las palabras vacías. También, se realizarán pruebas sin quitar las palabras vacías, para tomar la decisión final basándose en el desempeño del modelo en conjunto de prueba.



```
def custom_preprocess(text):
    text = text.lower()
    doc = nlp(text)
    text = ' '.join([token.lemma_ for token in doc if not token.is_stop])
    return text

tqdm.pandas()
dataTraining['plot'] =
dataTraining['plot'].progress_apply(custom_preprocess)
```

Bloque de código 2. Procesamiento lematización y stop words

Feature Engineering

Cuando el conjunto de datos está limpio y preprocesado, la dirección del proceso posterior depende de la técnica de modelado elegida. En este proyecto se probaron diferentes estrategias de representación de texto como la técnica de bolsa de palabras y también espacios vectoriales que capturen la semántica de las palabras mediante vectores densos como los embeddings de palabras que pueden proporcionar un mayor poder expresivo y capturar mejor la semántica y el contexto del texto.

En este informe, se documentará únicamente los procedimientos seguidos para el modelo basado en redes neuronales, mientras que los detalles de los modelos de ML estarán disponibles en el notebook correspondiente.

La preparación del conjunto de datos para ser usado como entrada de un modelo de redes neuronales comenzó con la conversión de secuencias de palabras en secuencias de índices. Para esto, se utilizó la función *Tokenizer* de la librería Keras. Los parámetros clave

en este paso incluyen el tamaño del vocabulario deseado, que determina las palabras reconocidas por el tokenizador y la longitud de la secuencia de índices. Las entradas que superen esta longitud serán truncadas, mientras que a las que les falte longitud, se les añadirán índices cero para alcanzar el tamaño requerido.

```
vocab_size = vocabulary_size
max_length = sequence_length
tokenizer = Tokenizer(num_words=vocab_size, oov_token="<OOV>")
tokenizer.fit_on_texts(df['plot'])
sequences = tokenizer.texts_to_sequences(df['plot'])
padded_sequences = pad_sequences(sequences, maxlen=max_length,
padding='post')
```

Bloque de código 3. Tokenización y Padding

Teniendo la secuencia de índices, donde cada uno de estos representa una palabra, el objetivo es convertir cada índice en un vector denso que ofrezca información al modelo sobre la semántica de esta palabra. Con este objetivo se probaron tres estrategias:

1. **Generar una primera capa de embedding en la red neuronal:** esta capa se entrenará al parejo de los otros pesos a medida que se recorre el conjunto de datos. Esta estrategia no mostró buenos resultados, muchas veces aumentando la pérdida incluso en el conjunto de entrenamiento.
2. **Entrenar un embedding propio antes de implementar la red neuronal:** Este embedding ya no sería entrenado junto a los demás parámetros de la red neuronal, sino que se usó el módulo Word2Vec para crear este embedding basándose en el corpus del problema. Esta estrategia demostró ofrecer resultados más favorables en términos de desempeño.
3. **Usar un embedding pre-entrenado:** Este embedding ya entrenado con un corpus mucho más significativo y de contenido generales. Para este proyecto se decidió usar los embedding de GloVe. Se inició probando con uno de tamaño 50, pero se observó que a medida que se aumentaba el tamaño, el modelo final generaba mejores resultados. El embedding elegido fue 'glove-wiki-gigaword-200' con una longitud de dimensión de 200. Así que cada índice se convertiría en un nuevo vector, teniendo como entrada de la red neuronal un tensor de tres dimensiones. Los detalles específicos de este procedimiento están disponibles para su revisión en el Bloque de código 4 del notebook.

```
def map_to_glove(sequence):
    embeddings = np.zeros((max_length, 200))
    for i, idx in enumerate(sequence):
        if i >= max_length:
            break
        if idx != 0:
            word = tokenizer.index_word.get(idx, None)
            if word in model_glove:
                embeddings[i] = model_glove[word]
            else:
                embeddings[i] = np.zeros(200)
    return embeddings
```

Bloque de código 4. Mapeo de palabras en GloVe

Se llegó a la conclusión de las dos primeras estrategias no produjeron resultados óptimos debido a la naturaleza de entrenar un embedding, que demanda una cantidad considerable de datos. Considerando que se disponía de 7895 tramas, se determinó que esta cantidad de datos podría no ser suficiente para permitir que el modelo aprendiera de manera efectiva.

El último paso precedente a la modelación es la transformación de la variable objetivo para permitir la predicción de múltiples géneros asociados con cada película y así poder trabajar con un problema de género multi-etiqueta. Como cada película puede pertenecer a uno o más géneros, se opta por qué “y” tenga distintas columnas binarias, donde cada una representa un género.

Modelo GRU

A continuación, se explicará el desarrollo del modelo, sin embargo, resaltaremos las características más importantes, ya que la documentación completa está disponible en el código.

Se realiza una división del conjunto de datos en entrenamiento y prueba, utilizando el 15% para el conjunto de prueba, también se realiza una estratificación por el género “News” debido a que este género tiene menos de 10 observaciones y dependiendo de la semilla utilizada, puede que no haya suficientes observaciones en uno de los conjuntos resultantes.

Para la arquitectura de la red neuronal, se estableció un callback de EarlyStopping, una técnica esencial para prevenir el sobreajuste durante el entrenamiento de nuestro modelo. Monitorea la métrica 'val_auc' (área bajo la curva ROC en el conjunto de validación) y detiene el entrenamiento si no observa mejora después de 20 épocas consecutivas.

```
early_stopping = EarlyStopping(  
    monitor='val_auc',  
    patience=20,  
    verbose=1,  
    restore_best_weights=True  
)
```

Una vez definido el conjunto de datos, se define la arquitectura de un modelo secuencial utilizando Keras. Inicia con una capa GRU de 256 unidades para procesar secuencias, seguido de varias capas densas para la clasificación, incluyendo una con activación 'PReLU' y otra con 'relu', y una capa de 'Dropout' para reducir el sobreajuste. La capa de salida emplea activación 'sigmoid', adecuada para tareas de clasificación binaria múltiple, con el número de neuronas igual al de clases de salida. El modelo se compila con un optimizador Adam y una tasa de aprendizaje inicial de 0.0001, usando la pérdida de entropía cruzada binaria y el AUC como métrica. Además, se incluye un callback 'ReduceLROnPlateau' para ajustar dinámicamente la tasa de aprendizaje en función del rendimiento, reduciéndola cuando 'val_loss' no mejora tras 10 épocas, con un límite inferior establecido.

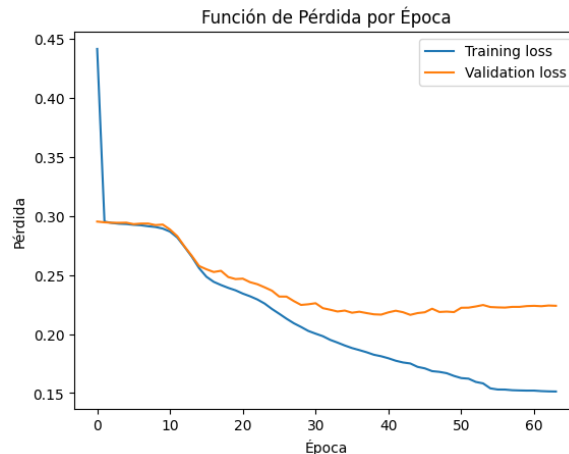
```

model = Sequential()
model.add(GRU(256, return_sequences=False, input_shape=(max_length, 200)))
model.add(Dense(256, activation='PReLU'))
model.add(Dropout(0.1))
model.add(Dense(256, activation='relu'))
model.add(Dense(outputClasses, activation='sigmoid'))
model.compile(optimizer=Adam(learning_rate=0.0001),
              loss='binary_crossentropy', metrics=[AUC(name='auc')])

# Callback para reducir la tasa de aprendizaje cuando la métrica de
# rendimiento deja de mejorar
reduce_lr = ReduceLRonPlateau(monitor='val_loss', factor=0.1, patience=10,
                              min_lr=0.000001, verbose=1)

```

Finalmente, el modelo se entrena sobre el conjunto de entrenamiento, haciendo una división para el subconjunto de validación que representará el 20% del total de entrenamiento, se opta por un tamaño de lote de 32 y 500 épocas. En la Gráfica 5, se presenta la historia del entrenamiento de la arquitectura, donde se puede observar que, aunque se había dado un número grande de épocas el entrenamiento fue detenido en la época 64.



Gráfica 5. Historia del entrenamiento

Al evaluar el modelo mediante la métrica del AUC, se observa un desempeño sobresaliente del modelo para distinguir entre las distintas clases. En la gráfica XXXX, se detalle que el modelo contó con un AUC Micro Score de 0.9196, se demuestra una alta eficacia en la clasificación a nivel individual, reflejando la precisión del modelo al manejar el balance entre la sensibilidad y la especificidad en el conjunto de datos. El AUC Macro Score de 0,8768, aunque inferior, sigue mostrando un rendimiento robusto al considerar el promedio de la capacidad de discriminación del modelo en todas las clases sin considerar el peso de cada una en total de datos.

```

ROC AUC Score: 0.9196248210496943
ROC AUC Score: 0.8768307018935043

```

Gráfica 6. AUC Micro y Macro del modelo GRU

Comparación de modelos y conclusiones

El modelo mencionado fue el que mejor rendimiento tuvo, pero se probaron diferentes modelos que también tuvieron resultados muy decentes y se mencionan a continuación.

Modelos	AUC Micro	AUC Macro
---------	-----------	-----------

Red Neuronal GRU	0.92	0.88
Random Forest + TF-IDF	0.90	0.87
Red Neuronal Convolutacional	0.90	0.76
XGBClassifier+Word2vec	0.86	0.74
SVC+Word2vec	0.87	0.79
Sentiment_polarity + TF-IDF	0.85	0.86

Al analizar la tabla de resultados del AUC Micro y Macro para los distintos modelos aplicados a la clasificación de géneros de películas a partir de su descripción, se puede concluir que la Red Neuronal GRU destaca como el modelo más eficaz, logrando un equilibrio entre alto rendimiento general y una buena capacidad de clasificación en todas las clases de género.

Para el modelo de random forest, se vectorizó con TF-IDF limitando el número de características a 50,000. A pesar no ser aprendizaje profundo, muestra resultados sólidos, con una capacidad de clasificación ligeramente inferior a la GRU, lo cual lo convierte en una opción viable cuando se busca un modelo que sea fácil de interpretar y que aun así ofrezca un rendimiento alto y confiable.

El modelo con red neuronal convolutacional usa un mapeo con embedding pre-entrenado y a pesar de tener un buen AUC Micro, revela una debilidad en la clasificación equitativa entre las clases, lo cual se refleja en su AUC Macro más bajo. Esto podría ser un indicativo de que ciertas clases no están siendo tan bien representadas o diferenciadas por el modelo.

Para el modelo de boosting usando gradiente extremo se optó por no usar un embedding pre-entrenado, sino entrenarlo por medio de la librería Word2Vec, aunque es conocido por su eficacia en una variedad de problemas de clasificación, en este contexto particular muestra un desempeño menos competitivo. Esto puede deberse a la necesidad de una optimización más fina de hiperparámetros o a limitaciones inherentes del modelo para este tipo específico de datos textuales.

En el modelo basado en la polaridad del sentimiento también se optó por una representación usando TF-IDF, el modelo muestra un rendimiento balanceado entre las clases, aunque su rendimiento general es menor que los modelos más complejos. Esto sugiere que la polaridad del sentimiento por sí sola es una característica informativa y podría ser útil como parte de un enfoque de modelado en conjunto o cuando se busca un modelo más interpretable y simple.

Revisión del estado del arte

Los enfoques sin embeddings tienen dificultades para modelar relaciones entre palabras y contexto. Bengio et al. (2003) propusieron un modelo de lenguaje basado en redes neuronales que aprende representaciones distribuidas de palabras, capturando información contextual mediante embeddings. Word2vec puede encontrar relaciones semánticas y sintácticas utilizando operaciones entre vectores, lo que permite representar películas como el promedio de los vectores de las palabras en la trama. La motivación principal para utilizar

vectores word2vec radica en su propiedad de composicionalidad aditiva, demostrada en diversas tareas como analogías de palabras. (John Arevalo, 2017)

Los anteriores argumentos fueron probados para algunos modelos y confirman que los modelos que tienen embeddings de palabras en su mayoría se comportan mejor que los que no lo incluyen.

Igualmente se probaron los stemmers de Porter, Lancaster y RBMVRs (Maheswari S, 2019), para normalizar el texto antes de la tokenización, en el que se evidenció un rendimiento similar para los modelos, especialmente para Xgboost utilizando el Lancaster Stemmer, aunque esto depende en parte de cómo se haya tratado la tokenización, utilizando subword tokenization (Una de las dificultades fue establecer las reglas para las oraciones del texto) buscando explorar una estructura más compleja en la tokenización, mostró inicialmente un rendimiento aceptable, pero al realizar un avance con el modelo presentó un pobre rendimiento al predecir en test.

Los parámetros utilizados en el stemmer de Porter fueron principalmente con length, se probaron diferentes dimensiones diferentes a su estándar de 3 y en que tan agresivo era el stemmer, en el de Lancaster se utilizó el set de reglas estándar y se omitió el parámetro de mayúsculas (Debido a la estandarización en minúsculas).

Referencias bibliográficas

- Agarwal, A. (07 de 08 de 2020). *Movie Genre Classifier using Natural Language Processing*. Obtenido de Medium: <https://medium.com/@aagarwal691/movie-genre-classifier-using-natural-language-processing-4d5e73da7b78>
- John Arevalo, T. S.-y.-G. (2017). Gated Multimodal Units for Information Fusion. *Universidad Nacional de Colombia*, 17.
- Maheswari S, K. A. (2019). Rule Based Morphological Variation Removable . *International Journal of Recent Technology and Engineering*, 6.
- NS, A. (13 de 01 de 2019). *Movie genre classifier using a dataset created using Google Images*. Obtenido de Medium: <https://towardsdatascience.com/building-a-movie-genre-classifier-using-a-dataset-created-using-google-images-4752f75a1d79>