INFORME EJECUTIVO COMPETENCIA MIAD2024-12 Clasificación de género de películas

Desarrollado por:

- KAREN YORLADY ROJAS GIRALDO
- JHOCEL DUVAN SUESCUN TORRES
- JUAN SEBASTIAN HERNANDEZ RAMIREZ
- JUAN PABLO MOGOLLœN AVAUNZA

1 Exploración preliminar de los datos

1.1 Se tienen dos archivos, uno para entrenar y comprobar el desempeÌ o del algoritmo (dataTraining.zip) y otro para obtener las predicciones para subir a la competencia en Kaggle (dataTesting.zip). Donde el conjunto de dataTraining tiene 7895 registros y 5 columnas o variables, mientras que el conjunto dataTesting tiene 3383 registros y un total de 3 columnas o variables.

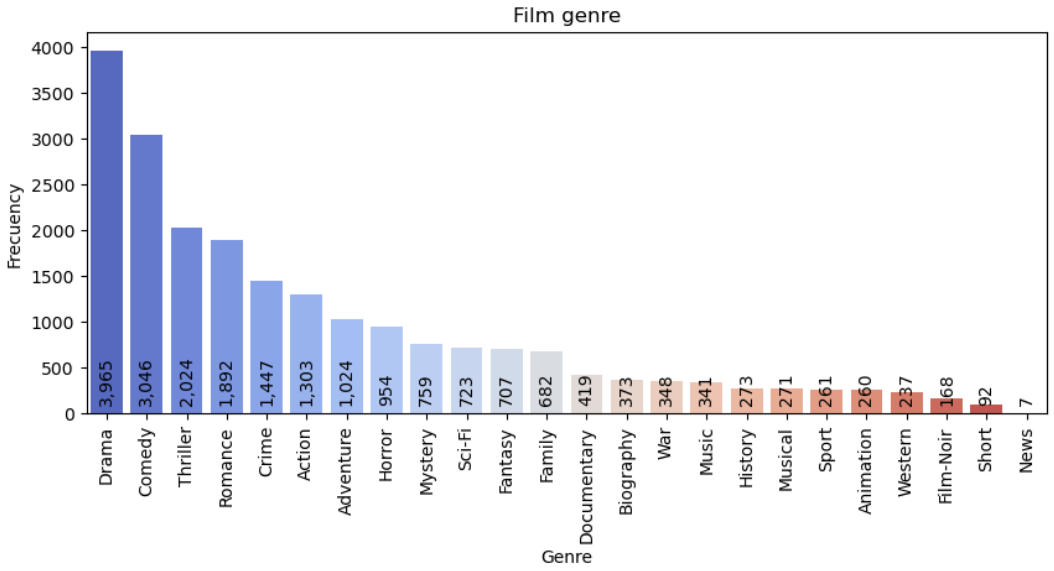
1.2 Los conjuntos de datos presentan informaciÔn de pelÈculas, especÈficamente 3 columnas categÔricas (title,plot,genres) y 2 numÂricas (year,rating), donde la variable genres corresponde a la variable de an› lisis a predecir, mientras que la variable plot es la variable con la que se har› el entrenamiento del modelo y a partir de ella se generar› n las correspondientes predicciones.

1.3 Debido a que la ˜nica variable que se usar› como predictora es ¢ploté no se har› un an› lisis exploratorio m› s profundo sobre las dem› s variables.

1.4 No hay valores faltantes en ninguno de los dos conjuntos de datos.

1.5 Se observa que la variable ¢genresées una combinaciÔn de m˜ ltiples valores ˜nicos del gÂnero de la pelÈcula, lo que quiere decir que las pelÈculas pueden tener uno o m› s gÂneros a los que pertenecen. Donde existen 1336 combinaciones ˜nicas de gÂneros que se conforman de 24 gÂneros base ˜nicos.

1.6 Por ˜ltimo, es posible observar la distribuciÔn de los gÂneros en dataTraining, donde es posible observar que la moda del gÂnero es ¢Dramaéy el gÂnero menos representativo es ¢Newsé



2 Preprocesamiento de datos

2.1 Se ajusta la naturaleza de la variable a predecir ¿genreé Inicialmente, esta tiene tipo string, por lo que es necesario aplicar la funciÔn eval sobre esta variable para que pase de strings a listas reales.

2.2 Estas listas de gÂneros se binarizan, para esto se hace uso de una instancia del objeto MultiLabelBinarizer que permite pasar de una columna de listas categÔricas a una matriz binaria que representa la pertenencia o no pertenencia de una pelÈcula a un gÂnero especÈfico.

2.3 Se realizo la divisiÔn de los datos en set de entrenamiento y prueba usando train_test_split utilizando el conjunto de datos de entrenamiento ètdataTrainingê esto con el ¿n de poder realizar evaluaciÔn de los resultados antes de obtener las predicciones en ètdataTestê para subir a la competencia.

2.4 Este es un fragmento del cÔdigo con el que se realizÔ la implementaciÔn descrita en los puntos anteriores:

```
# Cargar el dataframe
allData = pd.read_csv('https://github.com/albahnsen/MIAD_ML_and_NLP/raw/main/datasets/dataTraining.zip', encoding='UTF-8', index_col=0)

# Preparar los datos
X = allData['plot'].tolist()
y = allData['genres'].map(lambda x: eval(x))  # Asegurarse de que los géneros están en formato de lista

# Convertir las etiquetas a formato binarizado
mlb = MultiLabelBinarizer()
y_binarized = mlb.fit_transform(y)

# Dividir los datos en conjuntos de entrenamiento y prueba
X_train, X_test, y_train, y_test = train_test_split(X, y_binarized, test_size=0.33, random_state=42)
```

2.5 Para la transformaciÔn del texto que describe la pelÈcula incluido en la columna ¶plotése probaron diversos mÂtodos y su impacto en la capacidad predictiva del modelo a travÂs de la mÂtrica AUC(çmacroê).

2.6 Utilizamos las siguientes estrategias para preprocesar el texto: CountVectorizer, CountVectorizer con stop words, lematizaciÔn, n_gramas, T¸ dfVectorizer(), y combinaciones de ellas. A continuaciÔn, un ejemplo de la implementaciÔn de esta transformaciÔn.

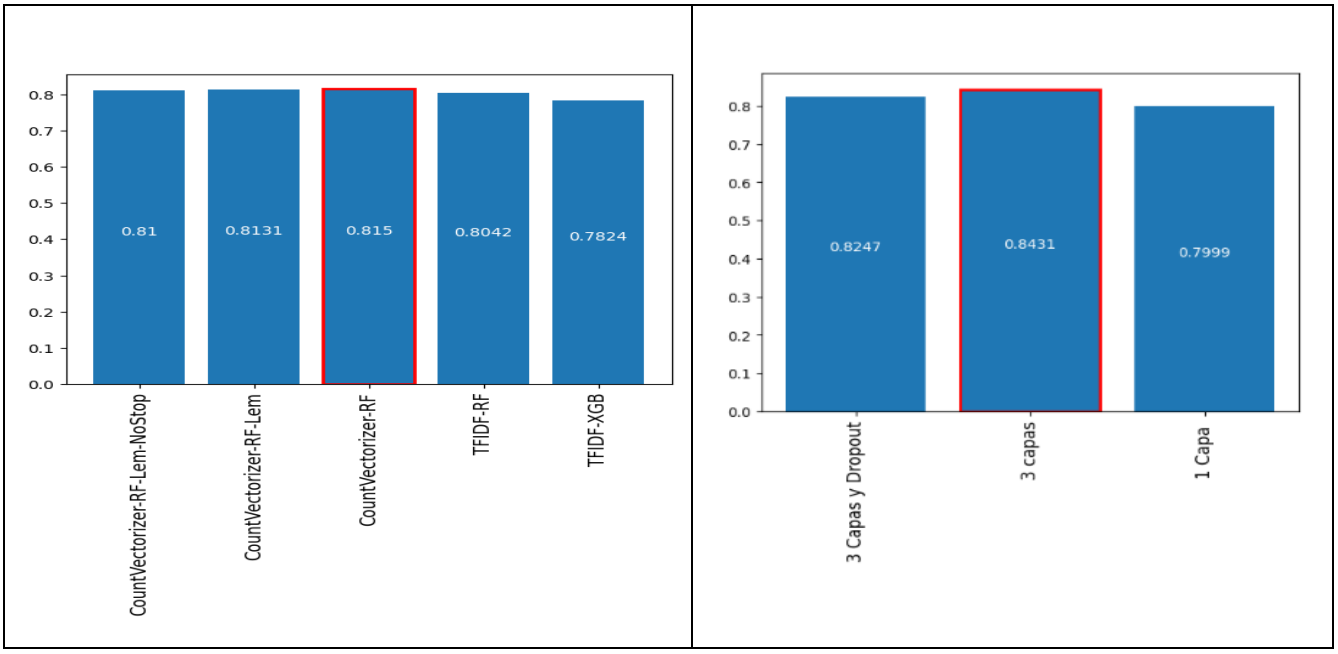
```
# Descargar las stopwords
nltk.download('stopwords')

# Lista de stop words
stop_words = set(stopwords.words('english'))
```

```
16.
17. # Función para eliminar stop words
18. def remove_stop_words(text):
19.     # Dividir el texto en palabras
20.     words = re.findall(r'\b\w+\b', text.lower())
21.     # Filtrar las palabras que no están en las stop words
22.     filtered_words = [word for word in words if word not in stop_words]
23.     # Unir las palabras filtradas de nuevo en una oración
24.     return ' '.join(filtered_words)
25.
26. # Descargar recursos de nltk
27. nltk.download('punkt')
28.
29. # Cargar el modelo en inglés de spaCy
30. nlp = spacy.load("en_core_web_sm")
31.
32. # Función para lematizar una frase
33. def lemmatize_sentence(sentence, nlp):
34.     # Procesar la frase con spaCy
35.     doc = nlp(sentence)
36.     # Extraer las lemas
37.     lemmas = [token.lemma_ for token in doc]
38.     # Unir las lemas en una frase
39.     lemmatized_sentence = " ".join(lemmas)
40.     return lemmatized_sentence
41.
42. # Cargar y preprocesar los datos
43. data = pd.read_csv('https://github.com/albahnsen/MIAD_ML_and_NLP/raw/main/datasets/dataTraining.zip', encoding='UTF-
8', index_col=0)
44. data.drop_duplicates(inplace=True)
45. data.dropna(subset=['plot', 'genres'], inplace=True)
46.
47. # Tokenizar las tramas de las películas
48. data['plot'] = data['plot'].apply(lambda x: ' '.join(word_tokenize(x.lower())))
49. data['plot'] = data['plot'].apply(remove_stop_words)
50. data['plot'] = data['plot'].apply(lambda x: lemmatize_sentence(x, nlp))
51.
52. # Definición de variable de interés (y)
53. data['genres'] = data['genres'].map(lambda x: eval(x))
54.
55. # Convertir géneros a etiquetas binarizadas
56. mlb = MultiLabelBinarizer()
57. genre_labels = mlb.fit_transform(data['genres'])
58.
59. # Separación de variables predictoras (X) y variable de interés (y) en set de entrenamiento y test usandola función
train_test_split
60. X_train, X_test, y_train_genres, y_test_genres = train_test_split(data['plot'], genre_labels, test_size=0.33, random_state=42)
61.
62. # Vectorizar los textos con TfidfVectorizer
63. tfidf_vectorizer = TfidfVectorizer()
64. X_train_tfidf = tfidf_vectorizer.fit_transform(X_train)
65. X_test_tfidf = tfidf_vectorizer.transform(X_test)
```

2.7 En este ejemplo realizamos la tokenizacion usando la biblioteca nltk, posteriormente eliminanos èstop wordsê y lematizamos utilizando spacy. Esto antes de dividir los datos en un conjunto de entrenamiento y pruebas. Luego vectorizamos los textos utilizando T¸ dfVectorizer(), para lo cual aplicamos è¸ t_transformê a los datos de entrenamiento y ètrasnformê a los datos de pruebas.

2.8 Para obtener datos preliminares de como las diferentes formas de prepocesamiento de los datos podrÌan afectar el rendimiento predicitvo de un modelo, utilizamos ramdom forest, xgboost y redes nueronales, calibrando sus par› metros y asÈcomo cambiando el procesamiento de los datos.

2.9 En las gr› ¸ cas siguientes un resumen breve de los resultados:



2.10 Para calibra RandomForest y XGBoost usamos el mÂtodo RandomizedSearchCV de debido a su e¸ciencia computacional, las redes neuronales las calibramos utilizando ParameterGrid. A continuaciÔn, un ejemplo del cÔdigo utilizado para realizar predicciones con XGBoost y calibrarlo utilizando RandomizedSearchCV. En este caso obtuvimos los siguientes resultados: En este caso obtuvimos los siguientes resultados:

- Best hyperparameters: {'subsample': 0.7, 'n_estimators': 500, 'min_child_weight': 3, 'max_depth': 7, 'learning_rate': 0.1, 'colsample_bytree': 0.5}
- Best macro ROC AUC: 0.7499537185773792
- Test macro ROC AUC: 0.8148595931613704

```
# Descargar recursos de nltk
nltk.download('punkt')

# Cargar y preprocesar los datos
data = pd.read_csv('https://github.com/albahnsen/MIAD_ML_and_NLP/raw/main/datasets/dataTraining.zip', encoding='UTF-8', index_col=0)
data.drop_duplicates(inplace=True)
data.dropna(subset=['plot', 'genres'], inplace=True)

# Tokenizar las tramas de las películas
data['plot'] = data['plot'].apply(lambda x: ' '.join(word_tokenize(x.lower())))

# Convertir géneros a etiquetas binarizadas
data['genres'] = data['genres'].map(lambda x: eval(x))
mlb = MultiLabelBinarizer()
genre_labels = mlb.fit_transform(data['genres'])

# Dividir los datos en conjunto de entrenamiento y prueba
X_train, X_test, y_train, y_test = train_test_split(data['plot'], genre_labels, test_size=0.2, random_state=42)

# Vectorizar los textos con TfidfVectorizer
tfidf_vectorizer = TfidfVectorizer()
X_train_tfidf = tfidf_vectorizer.fit_transform(X_train)
X_test_tfidf = tfidf_vectorizer.transform(X_test)

def macro_roc_auc_score(y_true, y_pred_proba):
    aucs = []
    for i in range(y_true.shape[1]):
        auc = roc_auc_score(y_true[:, i], y_pred_proba[:, i])
        aucs.append(auc)
    return np.mean(aucs)

def macro_roc_auc_scorer(estimator, X, y):
    y_pred_proba = estimator.predict_proba(X)
```

```
    return macro_roc_auc_score(y, y_pred_proba)

# Configurar el modelo XGBoost
xgb_clf = xgb.XGBClassifier(objective='binary:logistic', use_label_encoder=False, eval_metric='logloss')

# Definir los hiperparámetros a ajustar
param_distributions = {
    'learning_rate': [0.01, 0.1, 0.3],
    'max_depth': [3, 5, 7, 10],
    'min_child_weight': [1, 3, 5, 10],
    'subsample': [0.5, 0.7, 1.0],
    'colsample_bytree': [0.5, 0.7, 1.0],
    'n_estimators': [50, 100, 200, 500]
}

# Configurar la búsqueda de hiperparámetros con RandomizedSearchCV
opt = RandomizedSearchCV(
    estimator=xgb_clf,
    param_distributions=param_distributions,
    n_iter=5,
    scoring=macro_roc_auc_scorer,
    cv=2,
    n_jobs=-1,
    verbose = 1
)

# Realizar la búsqueda de hiperparámetros
opt.fit(X_train_tfidf, y_train)

# Obtener los mejores hiperparámetros y el mejor resultado
print("Best hyperparameters:", opt.best_params_)
print("Best macro ROC AUC:", opt.best_score_)

# Evaluar el modelo en el conjunto de prueba
best_model = opt.best_estimator_
y_pred_proba_test = best_model.predict_proba(X_test_tfidf)
test_macro_roc_auc = macro_roc_auc_score(y_test, y_pred_proba_test)
print("Test macro ROC AUC:", test_macro_roc_auc)

```

3 Selección del modelo para la competencia

Hasta el momento los modelos probados no lograron tener un desempeÌo superior a 0.85. De forma intuitiva consideramos que el mejor modelo podrÌa ser una red preentrenada, pero ∫cu› l? Al realizar una b˜squeda r› pida en EBSCO nos llamÔ la atenciÔn el uso de BERT en la soluciÔn de este tipo de problemas, particularmente una publicaciÔn de Alwyn y otros (2024) quienes re¸eren haber usado BERT y LSTM para encontrar una soluciÔn a este problema, obteniendo en ambos mÂtodos una exactitud superior al 95%. El articulo completo no est› disponible, aun asÈ decidimos probar los dos modelos. No logramos implementar un modelo LSTM adecuado para el conjunto de datos, obteniendo desempeÌos muy por debajo de los modelos previamente probados (con AUC inferiores a 0.75); por otro lado con el modelo de BERT logramos obtener AUC superiores a 0.89, por lo cual nos concentramos en dicho modelo.

En conclusiÔn, para la competencia seleccionamos un modelo preentrenado de BERT. A continuaciÔn describimos su implementaciÔn.

a) Cargamos los datos de training en allData, usamos estos datos para hacer predicciones preliminares antes de subir los datos a la competencia.

```
allData = pd.read_csv('dataTraining.zip', encoding='UTF-8', index_col=0)
```

b) Separamos los predictores y la variable de interÂs, y transformamos la variable de interÂs.

```
X = allData['plot'].tolist()
y = allData['genres'].map(lambda x: eval(x))
mlb = MultiLabelBinarizer()
y_binarized = mlb.fit_transform(y)
# como la transformación de y se hace antes del Split de datos, no es necesario aplicar transform en los datos de prueba.
```

c) Dividimos los datos en conjuntos de entrenamiento y prueba

```
X_train, X_test, y_train, y_test = train_test_split(X, y_binarized, test_size=0.2, random_state=42)
```

d) De¸ nimos una clase de Dataset para la compatibilidad con la biblioteca preentrenada

```
class MovieGenreDataset(Dataset):
    def __init__(self, texts, labels, tokenizer, max_length):
        self.texts = texts
        self.labels = labels
        self.tokenizer = tokenizer
        self.max_length = max_length

    def __len__(self):
        return len(self.texts)

    def __getitem__(self, idx):
        text = self.texts[idx]
        labels = self.labels[idx]
        encoding = self.tokenizer(
            text,
            padding='max_length',
            truncation=True,
            max_length=self.max_length,
            return_tensors='pt'
        )
        item = {key: val.squeeze(0) for key, val in encoding.items()}
        item['labels'] = torch.tensor(labels, dtype=torch.float)
        return item
```

e) Preparamos el tokenizer y el dataset con el modelo preentrenado

```
tokenizer = BertTokenizer.from_pretrained('bert-base-uncased')
train_dataset = MovieGenreDataset(X_train, y_train, tokenizer, max_length=128)
test_dataset = MovieGenreDataset(X_test, y_test, tokenizer, max_length=128)
```

Nota: se transforman los datos de entrenamiento y prueba usando el modelo Bert, por la clase implementada no se aplican los mÂtodos ¸ t_transform, ni transform, sino que se usa la clase de¸ nida para la compatibilidad con transform de Hugging Face.

f) Con¸ guramos el modelo

```
model = BertForSequenceClassification.from_pretrained('bert-base-uncased', num_labels=len(mlb.classes_))
```

4 Calibración y entrenamiento del modelo

4.1 Para la calibraciÔn del modelo utilizamos el mÂtodo ParameterGrid de sklearn. Este es un mÂtodo similar a GridSearchCV, que nos permite probar m˜ ltiples combinaciones de par› metros aunque con un costo computacional alto. Los hiperpar› metros que se calibraron fueron: 'num_train_epochs', 'per_device_train_batch_size', 'per_device_eval_batch_size', 'learning_rate', 'weight_decay' y 'warmup_steps'.

4.2 Con el¸ n de mitigar el costo computacional, corrimos el modelo m˜ ltiples veces, probando diferentes combinaciones de par› metros, sin que la combinaciÔn m› xima superara 30 posibilidades.

```
from sklearn.model_selection import ParameterGrid

param_grid = {
    'num_train_epochs': [3, 5],
    'per_device_train_batch_size': [8, 16, 32],
    'per_device_eval_batch_size': [8, 16, 32],
    'learning_rate': [1e-5],
    'weight_decay': [0.01],
    'warmup_steps': [500]
}

best_auc = 0
best_params = None

for params in ParameterGrid(param_grid):
    training_args = TrainingArguments(
```

```
        output_dir='./results',
        num_train_epochs=params['num_train_epochs'],
        per_device_train_batch_size=params['per_device_train_batch_size'],
        learning_rate=params['learning_rate'],
        weight_decay=params['weight_decay'],
        warmup_steps=params['warmup_steps'],
        logging_dir='./logs',
        logging_steps=10,
        evaluation_strategy="epoch"
    )

    trainer = Trainer(
        model=model,
        args=training_args,
        train_dataset=train_dataset,
        eval_dataset=test_dataset
    )

    trainer.train()
    predictions, labels, _ = trainer.predict(test_dataset)
    predictions = torch.sigmoid(torch.tensor(predictions)).numpy()
    auc_macro = roc_auc_score(y_test, predictions, average='macro')

    print(params, " auc: ", auc_macro)

    if auc_macro > best_auc:
        best_auc = auc_macro
        best_params = params
```

4.3 Dentro del for que toma la combinaciÔn de par› metros de la grilla previamente escrita, se de¸ ne tambiÂn el cÔdigo de entrenamiento del modelo y se obtienen y almacenan las predicciones de cada combinaciÔn probada, para seleccionar el mejor modelo probado. Estos son algunos de los resultados obtenidos en la calibraciÔn:

- {'learning_rate': 1e-05, 'num_train_epochs': 3, 'per_device_eval_batch_size': 16, 'per_device_train_batch_size': 32, 'warmup_steps': 500, 'weight_decay': 0.01} auc: 0.8722502314632695
- {'learning_rate': 1e-05, 'num_train_epochs': 3, 'per_device_eval_batch_size': 8, 'per_device_train_batch_size': 8, 'warmup_steps': 500, 'weight_decay': 0.01} auc: 0.8767313582601867
- {'learning_rate': 1e-05, 'num_train_epochs': 3, 'per_device_eval_batch_size': 16, 'per_device_train_batch_size': 8, 'warmup_steps': 500, 'weight_decay': 0.01} auc: 0.884627919970742

4.4 De acuerdo, con el comportamiento observado de las diferentes corridas podemos anotar lo siguiente sobre los hiperp› rametros calibrados:

- num_train_epochs (N˜ mero de Âpocas de entrenamiento): Indica cu› ntas veces el modelo pasar› por todo el conjunto de entrenamiento. Aumentar el n˜ mero de Âpocas puede mejorar el rendimiento del modelo hasta cierto punto; sin embargo, demasiadas Âpocas pueden llevar al sobreajuste, donde el modelo se ajusta demasiado a los datos de entrenamiento y su rendimiento empeora.
- per_device_train_batch_size (Tamaì o del lote de entrenamiento por dispositivo): Indica cu› ntos ejemplos de entrenamiento se procesan juntos en cada paso de optimizaciÔn. Un tamaì o de lote m› s grande puede disminuir el tiempo de entrenamiento y a una estimaciÔn m› s estable del gradiente, pero requiere m› s memoria. Un tamaì o de lote m› s pequeì o puede permitir un mejor ajuste en caso de limitaciones de memoria.
- per_device_eval_batch_size (Tamaì o del lote de evaluaciÔn por dispositivo): Indica cu› ntos ejemplos de evaluaciÔn se procesan juntos durante la evaluaciÔn del modelo. Similar al tamaì o del lote de entrenamiento, un tamaì o de lote de evaluaciÔn m› s grande puede hacer que la evaluaciÔn sea m› s r› pida pero requiere m› s memoria. No afecta directamente al entrenamiento, pero puede influir en la rapidez con que se pueden evaluar los resultados durante la validaciÔn.
- learning_rate (Tasa de aprendizaje): De¸ ne el tamaì o de los pasos que el optimizador dar› al actualizar los pesos del modelo. Una tasa de aprendizaje alta puede hacer que el modelo converja r› pidamente, pero corre el riesgo de saltarse mÈnimos Ôptimos y puede causar oscilaciones o

divergencia. Una tasa de aprendizaje baja puede llevar a una convergencia m› s estable y precisa, pero tambiÂn puede hacer que el entrenamiento sea muy lento.

- weight_decay: Tasa de decaimiento de los pesos utilizada para evitar el sobreajuste.El decaimiento de los pesos es una forma de regularizaciÔn que puede ayudar a evitar que el modelo se sobreajuste a los datos de entrenamiento. Sin embargo, si es demasiado alto, puede hacer que el modelo subajuste y no aprenda bien los patrones de los datos.
- warmup_steps: N˜mero de pasos de calentamiento durante los cuales la tasa de aprendizaje aumenta linealmente desde cero hasta su valor inicial. Los pasos de calentamiento pueden ayudar a estabilizar el entrenamiento en sus primeras etapas, evitando cambios bruscos en los gradientes que pueden causar inestabilidad. Una fase de calentamiento adecuada puede mejorar la convergencia del modelo.
- La calibraciÔn combinada de estos hiperpar› metros es crucial para obtener un rendimiento Ôptimo del modelo BERT

4.5 Hay otro hiperp› rmetro que no puede pasar desapercibido y es el çmax_lengthé de la clase MovieGenreDataset. Un mayor valor de este par› metro permite capturar m› s informaciÔn y mejora el rendimiento predictivo del modelo, pero requiere m› s recursos computacionales y m› s tiempo de entrenamiento. Para efectos del desarrollo de este ejercicio dejamos ¿jo el valor de este hiperpar› metro en 128.

4.6 Seleccionamos tres de las mejores combinaciones obtenidas para realizar las predicciones para la competencia:

- {'learning_rate': 1e-05, 'num_train_epochs': 3, 'per_device_eval_batch_size': 16, 'per_device_train_batch_size': 8, 'warmup_steps': 500, 'weight_decay': 0.01} AUC: 0.896693211994311
- {'learning_rate': 1e-05, 'num_train_epochs': 5, 'per_device_eval_batch_size': 16, 'per_device_train_batch_size': 8, 'warmup_steps': 500, 'weight_decay': 0.01} AUC: 0.8942387180530393
- {'learning_rate': 1e-05, 'num_train_epochs': 10, 'per_device_eval_batch_size': 16, 'per_device_train_batch_size': 8, 'warmup_steps': 500, 'weight_decay': 0.01} AUC: 0.8903683340544387

4.7 Para realizar las predicciones en los datos de prueba de la competenciafue necesario hacer una ajuste en la clase de compatibilidad para no usar çlabelsé, asÈ

```
class MovieGenreTestDataset(Dataset):
    def __init__(self, texts, tokenizer, max_length):
        self.texts = texts
        self.tokenizer = tokenizer
        self.max_length = max_length

    def __len__(self):
        return len(self.texts)

    def __getitem__(self, idx):
        text = self.texts[idx]
        encoding = self.tokenizer(
            text,
            padding='max_length',
            truncation=True,
            max_length=self.max_length,
            return_tensors='pt'
        )
        item = {key: val.squeeze(0) for key, val in encoding.items()}
        return item
```

Los resultados obtenidos en la competencia con este modelo, y los hiperpar› metros anotados en el punto anterior fueron:

4.8 Realizamos un ensamble promediando estos tres resultados, y los subimos a la competencia, obteniendo un desempeÌ o superior a 0.91 en la mÂtrica de la competencia.

4.9 Ya cerrando la competencia cambiamos el hiperparˆmetro ¸max_lengthé de la clase MovieGenreDataset a 256 y el n˜mero de epocas a 5 y 10; promediamos los resultados obtenidos y los subimos los resultados a la competencia y obteniendo nuestra mejor prediccIÔn con un score de 0.91754.

5 Publicación del modelo

Para la publicaciÔn del modelo se realizaron los siguientes pasos:

a. Se creÔen el servicio EC2 de AWS Cloud la instancia t2.small donde se habilitÔel puerto 5000 para consumir la api. Este es el enlace a la API. En caso que no estÂdisponible al momento de probarla por favor contactarnos.

b. El api tiene una ˜nica variable de entrada que corresponde al plot con la descripciÔn de la pelÈcula.

```
parser.add_argument(
    'Plot',
    type=str,
    required=True,
    help='Movie description',
    location='args')
```

c. Una vez realizadas las predicciones y obtenidas las probabilidades, se de¸ niÔun umbral de 0.5 para determinar los gÂneros vˆlidos para el plot enviado a la API.

```
umbral = 0.5

    # Encontrar los índices donde numeros es igual a 1
    indices_1 = np.where(prediction >= umbral)[1]

    total_generos = len(indices_1)

    if total_generos == 0:
        rta = 'No fue posible establecer el genero de la pelicula'
    else:
        generos = ['Action', 'Adventure', 'Animation', 'Biography', 'Comedy', 'Crime', 'Documentary', 'Drama', 'Family',
        'Fantasy', 'Film-Noir', 'History', 'Horror', 'Music', 'Musical', 'Mystery', 'News', 'Romance',
        'Sci-Fi', 'Short', 'Sport', 'Thriller', 'War', 'Western']

        # Obtener las palabras correspondientes a los índices encontrados
        genero = [generos[i] for i in indices_1]
        rta = f'Movie genres: {", ".join(genero)}'
```

d. Se implementÔ Docker para incluir dentro de un container todo lo necesario para ejecutar la aplicaciÔn. Para esto se desarrollÔ el respectivo Docker¸ le, donde dentro del folder proyecto tenemos los archivos: model.pkl: librerÈa del modelo desarrollado; app.py : api desarrollada; requirements.txt: contiene las diferentes librerÈas de Python utilizadas en la api

e. Una vez creado el container, se ejecuta y podemos consultar en el navegador la api desarrollada, se probaron 4 plot del set de prueba obteniendo los siguientes resultados:

- "alison parker is a successful top model , living with the lawyer michael lerman in his apartment . she tried to commit suicide twice in the past : the ¸ rst time , when she was a teenager and saw her father cheating her mother with two women in her home , and then when michael ' s wife died . since then , she left christ and the catholic church behind . alison wants to live alone in her own apartment and with the help of the real state agent miss logan , she ¸ nds a wonderful furnished old apartment in brooklyn heights for a reasonable rental . she sees a weird man in the window in the last floor of the building , and miss logan informs that he is father francis matthew halloran , a blinded priest who lives alone supported by the catholic church . alison moves to her new place , and

once there , she receives a visitor : her neighbor charles chazen welcomes her and introduces the new neighbors to her . then , he invites alison to his cat jezebel ' s birthday party in the night . on the next day , weird things happen with alison in her apartment and with her health . alison looks for miss logan and is informed that she lives alone with the priest in the building . a further investigation shows that all the persons she knew in the party were dead criminals . frightened with the situation , alison embraces christ again , while michael investigates the creepy events . alison realizes that she is living in the gateway to hell ." **{"result": "Movie genres: Drama, Horror, Thriller"}**

- "last time we see saw bella swan she was narrowly escaping the clutches of the evil vampire james while ¸ nding love with ' ' vegetarian ' ' vampire edward cullen . bella and edward ' s lives have been full of nothing but love and bliss however , it all changes one fateful day . on bella ' s birthday , her new found friend and sister of edward , alice , decides to throw her lavish party , complete with balloons , ribbons and cake that could feed an army . all is well until bella accidentally cuts her ¸ nger whilst opening a present . the result is that jasper hale , the newest addition to the cullen clan , succumbs to his blood lust and attacks bella . edward decides that while he and his family are around , bella ' s life will always be at risk . so he decides to leave her for her own good . bella feels her life is over . enter jacob black , a member of the quilite tribe who manages to bring some joy and meaning back into bella ' s life . however as the two become closer , bella discovers jacob has a secret of his own - he ' s a werewolf . as if that wasn ' t bad enough bella can ' t seem to get the love of her life , edward out of her mind . with new dangers , new friends and new enemies , bella ¸ nds herself choosing between holding on to the past or accepting a new future . but what and more importantly who will she choose ?" **{"result": "Movie genres: Adventure, Drama, Fantasy, Romance"}**
- "at her husband ' s funeral , pearl (shirley maclaine) , jewish mother of two divorced and antagonistic daughters , meets an old italian friend (marcello mastroianni) of her husband , whose advice years previously had stopped the husband leaving home . for N years he , now a widower , has secretly loved pearl . . ." **{"result": "Movie genres: Comedy, Drama, Romance"}**
- "john rambo is a vietnam veteran , winner of the medal of honour for serving his country in the vietnam war and the last surviving member of the unit he was in . rambo arrives in a small town , where he is arrested by the abusive local sheriff will teasle for refusing to leave town . rambo is mistreated and he relives his painful memories of being tortured in a prison camp , which goes too far and rambo escapes from police custody . rambo is pursued by teasle and the local police into the woods and rambo begins a personal war with teasle , and uses his combat skills and hunts down teasle and his men . rambo ' s former commanding of¸ cer colonel samuel trautman arrives believing teasle and his men don ' t stand a chance with rambo , and tries to put rambo ' s personal war to a end , as teasle wants rambo dead ." **{"result": "Movie genres: Drama, Thriller, War"}**

6 Conclusiones

6.1 La red neuronal pre entrenada puesta a prueba en este contexto presentÔ los mejores resultados (esto era de esperarse teniendo en cuenta que ha sido entrenada con millones de par› metros por lo que termina siendo altamente robusta)

6.2 Los mÂtodos de tokenizaciÔn convencionales y las redes neuronales tambiÂn generaron resultados satisfactorios a pesar de no ser los mejores modelos para este contexto.

6.3 Aunque las redes neuronales pre entrenadas suelen generar los mejores resultados al tener los entrenamientos m› s robustos, la implementaciÔn y calibraciÔn no es tan sencilla como con los dem› s mÂtodos convencionales.

6.4 Las redes neuronales pre entrenadas demandan la implementaciÔn de GPU, de no ser asÈel gasto computacional es tan severo que no hace sentido implementarlas.

6.5 El mejor desempeÌ o en la competencia se logro utilizando la red preentrenada de BERT, sin embargo, no se logro calibrar para obtener los resultados de alto desempeÌ o mencionados por Alwyn y otros (2024). Para este set de datos los mejores resultados se obtuvieron con un n˜ mero de Âpocas pequeÌ o menor a 10, siendo el mejor resultado obtenido con 3 Âpocas.

6.6 Tal como se ha evidenciado en otros desarrollos en general el valor de los hiperp› rametros en relaciÔn con el desempeÌ o predictivo del modelo describe una curva en la que el incremento o decremento del valor del hiperpar› metro mejora el desempeÌ o hasta cierto punto, a partir del cual el modelo se sobreajusta y pierde poder predictivo.

7 Referencias

Alwyn, A., Pranoto, E. J. P., Ichsan, I., Halim, K., Justin, W., & Girsang, A. S. (2024). Movie genre classi¸ cation using BERT and LSTM. AIP Conference Proceedings, 2927(1), 1í5. https://doi-org.ezproxy.uniandes.edu.co/10.1063/5.0193424