NLTK vs BERT: Is there a significant difference in the performances of the models on English and Spanish text?

Research idea:

- We are addressing the problem “Is BERT more accurate than NLTK at determining the sentiment of English and Spanish reviews”. There seems to be limited literature and documentation on the difference in NLP models NLTK and BERT’s abilities to correctly analyze the sentiment of text in Spanish and English. So, answering this question will give us insights into the potential strengths and weaknesses of BERT and NLTK at predicting sentiment. As a result, we may have an idea of which NLP model to use depending on the given language.

Dataset:

- The dataset we are using is titled “IMDB Dataset of 50K Movie Reviews (Spanish)”(Fig. 1), which is open and available on Kaggle. The dataset includes 50,000 IMDB movie reviews in English, the Spanish translation, and the sentiment of the review. The Kaggle dataset is based on an original dataset from the paper Learning Word Vectors for Sentiment Analysis (L. Mass et al) which includes only the English reviews with the associated sentiment. The dataset contains 50,000 movie reviews with an equal number of positive and negative reviews. Sentiment labels were determined by the score of the review: a negative review had a score less than or equal to 4 out of 10 and a positive review had a score greater than or equal to 7 out of 10. To get the spanish translations for the reviews, the Googletrans python library which uses the Google Translate API was applied. Furthermore, we plan on using the movie reviews as input for the NLTK and BERT models to acquire the sentiment analysis for each model individually.
- Link to the dataset: https://www.kaggle.com/datasets/luisdiegofv97/imdb-dataset-of-50k-movie-reviews-spanish

Previous analysis on this data:

- There have been previous analyses performed on similar datasets. Since the internet evolved to be a shared space for users all over the world, statistical analysis of language sentiment through the usage of algorithms has been a heated topic, as efficient understanding of either user-reviews or comment in different languages can be beneficial for marketing, customer service, or overall online environment improvements. For example, Grader et al. utilized a different type of language model, MLSLDA, for sentiment analysis on a dataframe that contains German and English movie reviews to understand its performance when dealing with multilingual data. Similarly, Ozturk et al., trained a

Rsentiment model to process Turkish tweets about the Syrian war and compared the result to that of the English tweets, which yielded interesting results that showed interesting polarizing opinions between the 2 countries about the war. These studies inspired us to compare the performance of more efficient language processing models such as NLTK and BERT on different languages.

NLTK for English and Spanish text

- NLTK's sentiment intensity analyzer is a tool fine tuned to read and predict the sentiment of a sentence, based on the words in it. It's a rule based and deterministic model built on bag of words representation of tokens in a sentence. It returns a value between -1 and 1, -1 corresponding to a highly negative sentence, and 1 corresponding to a highly positive one. For the sake of simplicity and consistency, we classified any number <= 0 to be a negative sentence, and > 0 to be a positive one. For instance, if a sentence has more "positive words", i.e, "happy" or "good", it would most likely be predicted to be a positive statement, and a sentence with more words such as "hate" and "detest" to be negative.

In [9]:
```
import numpy as np
import pandas as pd
import scipy.stats as stats
import statsmodels

import nltk
from nltk.sentiment import SentimentIntensityAnalyzer

from nltk.sentiment import SentimentIntensityAnalyzer
import seaborn as sns
import matplotlib.pyplot as plt
```

Function to use the the sentiment intensity analyzer from NLTK:

In [10]:
```
def score_discrete(score):
    if score <= 0:
        return -1
    elif score > 0:
        return 1

def sentiment_analysis(text):
    sia = SentimentIntensityAnalyzer()
    sentiment_scores = sia.polarity_scores(text)
    return sentiment_scores['compound']
```

In [11]:
```
df_eng_span = pd.read_csv('test_df.csv', encoding='utf8')
df_eng_span.head()
```

```
Out[11]:
```

	Unnamed: 0	review_en	review_es	sentiment
0	40548	As a Spanish tourist in Los Angeles and a fana...	Como turista español en Los Ángeles y un amant...	negative
1	25110	Excellent movie about a big media firm and the...	Excelente película sobre una gran firma de med...	positive
2	25040	As someone else has already said here, every s...	Como alguien más ya ha dicho aquí, cada escena...	positive
3	23051	Of all movies (and I'm a film graduate, if tha...	De todas las películas (y soy un graduado de c...	negative
4	14182	This is the second Eytan Fox film I have seen....	Esta es la segunda película de Eytan Fox que h...	positive

change positve and negative to 1 and 0

```
In [12]: df_eng_span['sentiment'] = df_eng_span['sentiment'].replace({'positive' : 1, 'negative
         df_eng_span.head()
```

```
Out[12]:
```

	Unnamed: 0	review_en	review_es	sentiment
0	40548	As a Spanish tourist in Los Angeles and a fana...	Como turista español en Los Ángeles y un amant...	-1
1	25110	Excellent movie about a big media firm and the...	Excelente película sobre una gran firma de med...	1
2	25040	As someone else has already said here, every s...	Como alguien más ya ha dicho aquí, cada escena...	1
3	23051	Of all movies (and I'm a film graduate, if tha...	De todas las películas (y soy un graduado de c...	-1
4	14182	This is the second Eytan Fox film I have seen....	Esta es la segunda película de Eytan Fox que h...	1

take a random sample of size n

```
In [13]: df_nltk_sample = df_eng_span.sample(n=1000, replace = True)
```

```
In [17]: #import nltk
         #nltk.download('vader_lexicon')

[nltk_data] Downloading package vader_lexicon to
[nltk_data]     /home/svyasabattu/nltk_data...

Out[17]: True
```

applying nltk to english reviews

```
In [18]: df_nltk_sample['nltk_score_eng'] = df_nltk_sample['review_en'].apply(sentiment_analysi
         df_nltk_sample['nltk_score_eng_discrete'] = df_nltk_sample['nltk_score_eng'].apply(sco
df_nltk_sample.head()
```

Out[18]:

	Unnamed: 0	review_en	review_es	sentiment	nltk_score_eng	nltk_score_eng_discrete
14	15046	I saw this movie one time at a kiddie matinée ...	Vi una vez en la película una vez en un Kiddie...	-1	0.9789	1
28	39616	It must be a long time ago that I have seen su...	Debe pasar hace mucho tiempo que he visto una ...	-1	-0.9654	-1
35	22706	This is truly terrible: painfully irritating s...	Esto es verdaderamente terrible: los artistas ...	-1	-0.9531	-1
38	37835	First off, I would like to point out that whil...	En primer lugar, me gustaría señalar que, si b...	-1	0.4506	1
2	25040	As someone else has already said here, every s...	Como alguien más ya ha dicho aquí, cada escena...	1	0.9817	1

applying nltk to spanish reviews

In [19]:
```
#applying nltk to spanish reviews
df_nltk_sample['nltk_score_es'] = df_nltk_sample['review_es'].apply(sentiment_analysis
df_nltk_sample['nltk_score_es_discrete'] = df_nltk_sample['nltk_score_es'].apply(score

df_nltk_sample.head()
```

Out[19]:

	Unnamed: 0	review_en	review_es	sentiment	nltk_score_eng	nltk_score_eng_discrete	nltk_sco
14	15046	I saw this movie one time at a kiddie matinée ...	Vi una vez en la película una vez en un Kiddie...	-1	0.9789	1	-0
28	39616	It must be a long time ago that I have seen su...	Debe pasar hace mucho tiempo que he visto una ...	-1	-0.9654	-1	-0
35	22706	This is truly terrible: painfully irritating s...	Esto es verdaderamente terrible: los artistas ...	-1	-0.9531	-1	-0
38	37835	First off, I would like to point out that whil...	En primer lugar, me gustaría señalar que, si b...	-1	0.4506	1	-0
2	25040	As someone else has already said here, every s...	Como alguien más ya ha dicho aquí, cada escena...	1	0.9817	1	0

compute accuracy

In [20]:
```
en_accuracy = df_nltk_sample[df_nltk_sample["nltk_score_eng_discrete"] == df_nltk_samp

es_accuracy = df_nltk_sample[df_nltk_sample["nltk_score_es_discrete"] == df_nltk_sampl
```

In [21]:
```
en_accuracy
```

Out[21]:
```
0.668
```

In [22]:
```
es_accuracy
```

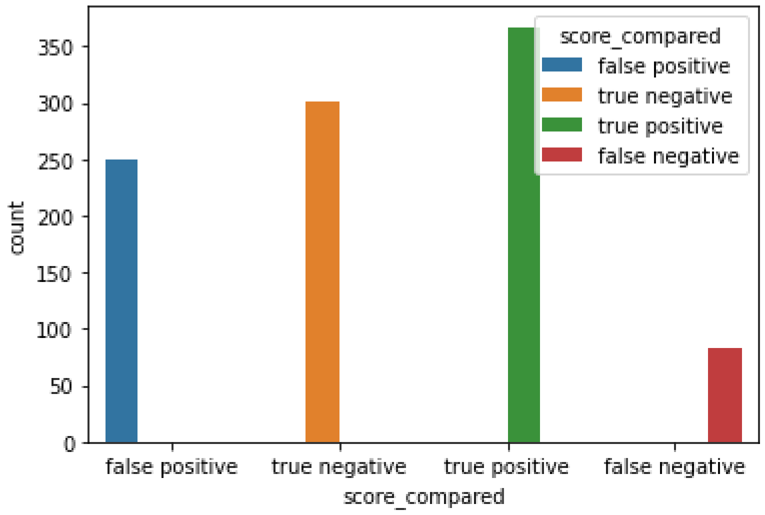
Out[22]:
```
0.678
```

In [23]:
```
def case_sorter(score, binary_nltk_score):
    if score == 1 and binary_nltk_score == 1:
        return 'true positive'
    elif score == 1 and binary_nltk_score == -1:
        return 'false negative'
    elif score == -1 and binary_nltk_score == 1:
        return 'false positive'
    else:
        return 'true negative'
```

check english accuracy

```
In [24]: score_counts = pd.DataFrame({'score_compared' : pd.Series(df_nltk_sample.apply(lambda
         sns.countplot(data = score_counts, x = 'score_compared', hue = 'score_compared')
```

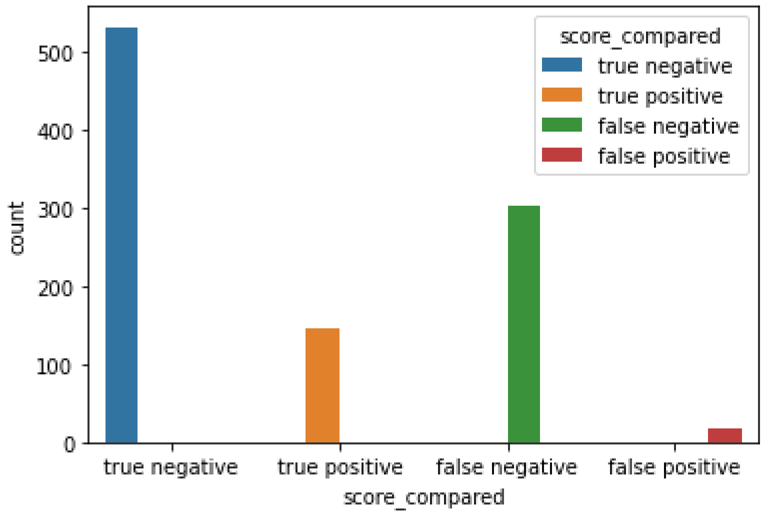
```
Out[24]: <AxesSubplot:xlabel='score_compared', ylabel='count'>
```



check spanish accuracy

```
In [25]: score_counts = pd.DataFrame({'score_compared' : pd.Series(
             df_nltk_sample.apply(lambda row: case_sorter(row['sentiment'], row['nltk_score_es_
         sns.countplot(data = score_counts, x = 'score_compared', hue = 'score_compared')
```

```
Out[25]: <AxesSubplot:xlabel='score_compared', ylabel='count'>
```



```
In [26]: df_eng_span.head()
```

Out[26]:

	Unnamed: 0	review_en	review_es	sentiment
0	40548	As a Spanish tourist in Los Angeles and a fana...	Como turista español en Los Ángeles y un amant...	-1
1	25110	Excellent movie about a big media firm and the...	Excelente película sobre una gran firma de med...	1
2	25040	As someone else has already said here, every s...	Como alguien más ya ha dicho aquí, cada escena...	1
3	23051	Of all movies (and I'm a film graduate, if tha...	De todas las películas (y soy un graduado de c...	-1
4	14182	This is the second Eytan Fox film I have seen....	Esta es la segunda película de Eytan Fox que h...	1

In [27]:
```
df_nltk_sample = df_eng_span.sample(n=1000, replace = True)
```

applying nltk to english reviews

In [28]:
```
df_nltk_sample['nltk_score_eng'] = df_nltk_sample['review_en'].apply(sentiment_analysi
df_nltk_sample['nltk_score_eng_discrete'] = df_nltk_sample['nltk_score_eng'].apply(sco

df_nltk_sample.head()
```

Out[28]:

	Unnamed: 0	review_en	review_es	sentiment	nltk_score_eng	nltk_score_eng_discrete
16	35585	Ti%s and As*, lots of boobies. Some great char...	TI% S y AS *, muchos piqueros.Algunos grandes ...	1	0.9897	1
4	14182	This is the second Eytan Fox film I have seen....	Esta es la segunda película de Eytan Fox que h...	1	0.7552	1
35	22706	This is truly terrible: painfully irritating s...	Esto es verdaderamente terrible: los artistas ...	-1	-0.9531	-1
31	1943	I saw The Merchant of Venice in London last we...	Vi al comerciante de Venecia en Londres la sem...	1	0.8770	1
8	29180	A friend of mine decided to rent this thing, I...	Un amigo mío decidió alquilar esta cosa, suert...	-1	-0.9579	-1

applying nltk to spanish reviews

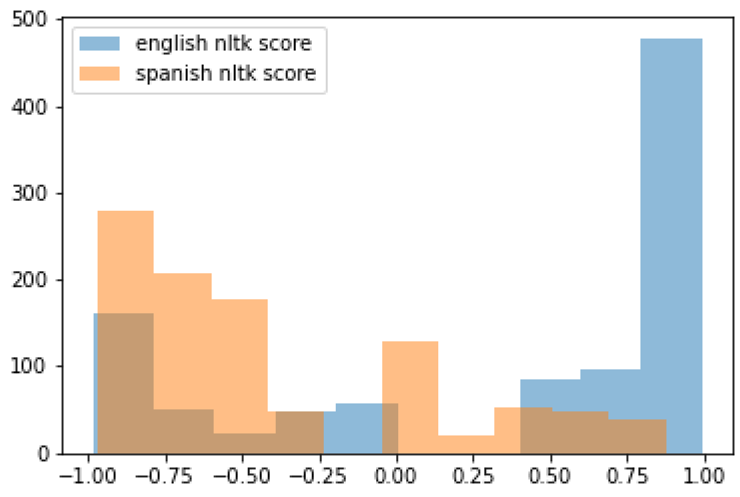
```
In [29]: df_nltk_sample['nltk_score_es'] = df_nltk_sample['review_es'].apply(sentiment_analysis
         df_nltk_sample['nltk_score_es_discrete'] = df_nltk_sample['nltk_score_es'].apply(score

         df_nltk_sample.head()
```

Out[29]:

	Unnamed: 0	review_en	review_es	sentiment	nltk_score_eng	nltk_score_eng_discrete	nltk_sc
16	35585	Ti%s and As*, lots of boobies. Some great char...	TI% S y AS *, muchos piqueros.Algunos grandes ...	1	0.9897	1	
4	14182	This is the second Eytan Fox film I have seen....	Esta es la segunda película de Eytan Fox que h...	1	0.7552	1	-
35	22706	This is truly terrible: painfully irritating s...	Esto es verdaderamente terrible: los artistas ...	-1	-0.9531	-1	-
31	1943	I saw The Merchant of Venice in London last we...	Vi al comerciante de Venecia en Londres la sem...	1	0.8770	1	
8	29180	A friend of mine decided to rent this thing, I...	Un amigo mío decidió alquilar esta cosa, suert...	-1	-0.9579	-1	-

Based on the plot below, nltk seems to assign more positive ratings to english than spanish, and there seems to be more neutral scores for spanish. there is a higher density of scores at the extremes close to -1 and 1 than in the center

```
In [30]: plt.hist(df_nltk_sample['nltk_score_eng'], alpha = 0.5, label = 'english nltk score')
         plt.hist(df_nltk_sample['nltk_score_es'], alpha = 0.5, label = 'spanish nltk score')
         plt.legend()
         plt.show()
```



Permutating 15 tests to get a distribution for NLTK's accuracy on english text:

In [31]:
```
# run nltk on 15 samples of size 1000

nltk_en_values = []
for i in range(15):
    df_nltk_sample = df_eng_span.sample(n=1000, replace = True)
    df_nltk_sample['nltk_score_en'] = df_nltk_sample['review_en'].apply(sentiment_anal
    df_nltk_sample['nltk_score_en_discrete'] = df_nltk_sample['nltk_score_en'].apply(s
    acc = df_nltk_sample.query('sentiment == nltk_score_en_discrete').shape[0]/df_nltk
    nltk_en_values.append(acc)
```

Distribution of NLTK's accuracy on sentimental analysis of english text:

In [32]:
```
nltk_en_values
```

Out[32]:
```
[0.707,
 0.694,
 0.722,
 0.7,
 0.692,
 0.698,
 0.702,
 0.711,
 0.717,
 0.684,
 0.712,
 0.71,
 0.702,
 0.697,
 0.717]
```

Permutating 15 tests to get a distribution for NLTK's accuracy on spanish text:

In [33]:
```
nltk_es_values = []
for i in range(15):
    df_nltk_sample = df_eng_span.sample(n=1000, replace = True)
    df_nltk_sample['nltk_score_es'] = df_nltk_sample['review_es'].apply(sentiment_anal
    df_nltk_sample['nltk_score_es_discrete'] = df_nltk_sample['nltk_score_es'].apply(s
    acc = df_nltk_sample.query('sentiment == nltk_score_es_discrete').shape[0]/df_nltk
    nltk_es_values.append(acc)
```

Distribution of NLTK's accuracy on sentimental analysis of spanish text:

In [34]:
```
nltk_es_values
```

Out[34]:
```
[0.647,
 0.634,
 0.654,
 0.644,
 0.633,
 0.651,
 0.634,
 0.648,
 0.653,
 0.663,
 0.653,
 0.634,
 0.68,
 0.677,
 0.653]
```

BERT for English and Spanish text

- BERT (Bidirectional Encoder Representations from Transformers) predicts the sentiment of a sentence by looking at the whole sentence, unlike NLTK which pays attention to singular tokens in a sentence. It can understand how each word affects the sentence, and how this changes the sentiment. This way, it is also influenced by by the feelings in the sentence like sarcasm or negation, and hence our speculation that it's more accurate than NLTK
- After being fine-tuned on our data, BERT predicts the label of the sentiment of the sentence, rather than giving a sentiment score like NLTK. So we wouldn't need to create a scale for the labels.
- It uses a next word prediction algorithm, similar to that of chatGPT, by encoding 15% of the sentences and adding noise, and learning how to decode it via the training process.
- We used a pre-trained model from hugging face called bert base uncased. It was pre-trained on english text in a supervised manner and is not affected by the casing of the tokens, i.e, 'Word' is considered the same as 'word'.

In [1]:
```python
import os
import shutil
import tarfile
import tensorflow as tf

from tensorflow.keras.optimizers import Adam
from transformers import BertTokenizer, TFBertForSequenceClassification

import numpy as np
import pandas as pd
import random

import matplotlib.pyplot as plt
import plotly.express as px
import plotly.offline as pyo
import plotly.graph_objects as go

from sklearn.model_selection import train_test_split
from sklearn.metrics import classification_report

# Not required but useful
from transformers import logging
logging.set_verbosity_error()

from tqdm.notebook import tqdm
tqdm.pandas()
```

```
2024-03-14 03:52:34.131987: I tensorflow/core/platform/cpu_feature_guard.cc:182] This
TensorFlow binary is optimized to use available CPU instructions in performance-criti
cal operations.
To enable the following instructions: AVX2 FMA, in other operations, rebuild TensorFl
ow with the appropriate compiler flags.
```

In [1]:
```
#!pip install transformers
```

Data Cleaning

In [3]:
```
df = pd.read_csv('IMDB Dataset SPANISH.csv', encoding='utf8')
df = df[['review_en', 'review_es', 'sentiment']]
```

In [4]:
```
df.head()
```

Out[4]:

	review_en	review_es	sentiment
0	One of the other reviewers has mentioned that ...	Uno de los otros críticos ha mencionado que de...	positive
1	A wonderful little production. The filming tec...	Una pequeña pequeña producción.La técnica de f...	positive
2	I thought this was a wonderful way to spend ti...	Pensé que esta era una manera maravillosa de p...	positive
3	Basically there's a family where a little boy ...	Básicamente, hay una familia donde un niño peq...	negative
4	Petter Mattei's "Love in the Time of Money" is...	El "amor en el tiempo" de Petter Mattei es una...	positive

In [5]:
```
# Change sample size here
train_n = 5000
test_n = 5000
```

In [6]:
```
# Splits data between train(60%) and test(40%)
random.seed(42)

train_df = df.sample(train_n, random_state=42) #df[['review_en', 'sentiment', 'review_
test_df = (df[~df.isin(train_df)].dropna()).sample(test_n, random_state=42) #[['review
```

Sentiment Analysis: English

In [8]:
```
# Splits train between train(40%) and validation(20%)
x_train_eng, x_val_eng, y_train_eng, y_val_eng = train_test_split(
                                                    train_df['review_en'],
                                                    train_df['sentiment'],
                                                    test_size=0.25,
                                                    stratify = train_df['sentiment'],
                                                    random_state=42
                                                )
```

In [9]:
```
# Tokenizes and encodes the sentences for training
tokenizer = BertTokenizer.from_pretrained('bert-base-uncased', do_lower_case=True)
max_len= 128

X_train_encoded_eng = tokenizer.batch_encode_plus(x_train_eng.tolist(),
                                              padding=True,
                                                        truncation=True,
                                                        max_length = max_len,
                                                        return_tensors='tf')

X_val_encoded_eng = tokenizer.batch_encode_plus(x_val_eng.tolist(),
                                                        padding=True,
                                                        truncation=True,
                                                        max_length = max_len,
                                                        return_tensors='tf')
```

```
2024-03-14 03:52:54.546847: I tensorflow/core/common_runtime/gpu/gpu_device.cc:1639]
Created device /job:localhost/replica:0/task:0/device:GPU:0 with 10174 MB memory:  ->
device: 0, name: NVIDIA A30 MIG 2g.12gb, pci bus id: 0000:c3:00.0, compute capabilit
y: 8.0
```

In [10]:
```
# Gets model and fine-tunes
LEARNING_RATE = 5e-5

model = TFBertForSequenceClassification.from_pretrained('bert-base-uncased', num_label

optimizer = tf.keras.optimizers.legacy.Adam(learning_rate=LEARNING_RATE)
loss = tf.keras.losses.SparseCategoricalCrossentropy(from_logits=True)
metric = tf.keras.metrics.SparseCategoricalAccuracy('accuracy')

model.compile(optimizer=optimizer,
              loss=loss,
              metrics=[metric])
```

In [11]:
```
# Cleans datasets for model fitting
train_dataset = tf.data.Dataset.from_tensor_slices((
    dict(X_train_encoded_eng),
    y_train_eng.apply(lambda x: True if x=='positive' else False)
))

test_dataset = tf.data.Dataset.from_tensor_slices((
    dict(X_val_encoded_eng),
    y_val_eng.apply(lambda x: True if x=='positive' else False)
))
```

- Epoch = total number of training iterations
- We used 4 as we observed that training loss was decreasing but the validation loss stopped at the 4th epoch.

In [12]:
```
# Fits model based on data
history = model.fit(train_dataset.batch(16),
              epochs=4,
              batch_size=16,
              validation_data=test_dataset.batch(16))
```

```
Epoch 1/4
  1/235 [..............................] - ETA: 53:28 - loss: 0.6893 - accuracy: 0.43
75
```

```
2024-03-14 03:53:13.172169: I tensorflow/compiler/xla/stream_executor/cuda/cuda_blas.
cc:606] TensorFloat-32 will be used for the matrix multiplication. This will only be
logged once.
```

```
235/235 [==============================] - 55s 176ms/step - loss: 0.4462 - accuracy:
0.7824 - val_loss: 0.3863 - val_accuracy: 0.8184
Epoch 2/4
235/235 [==============================] - 39s 167ms/step - loss: 0.2269 - accuracy:
0.9123 - val_loss: 0.4606 - val_accuracy: 0.8264
Epoch 3/4
235/235 [==============================] - 39s 168ms/step - loss: 0.1318 - accuracy:
0.9539 - val_loss: 0.8067 - val_accuracy: 0.7736
Epoch 4/4
235/235 [==============================] - 39s 168ms/step - loss: 0.1184 - accuracy:
0.9573 - val_loss: 0.5121 - val_accuracy: 0.8400
```

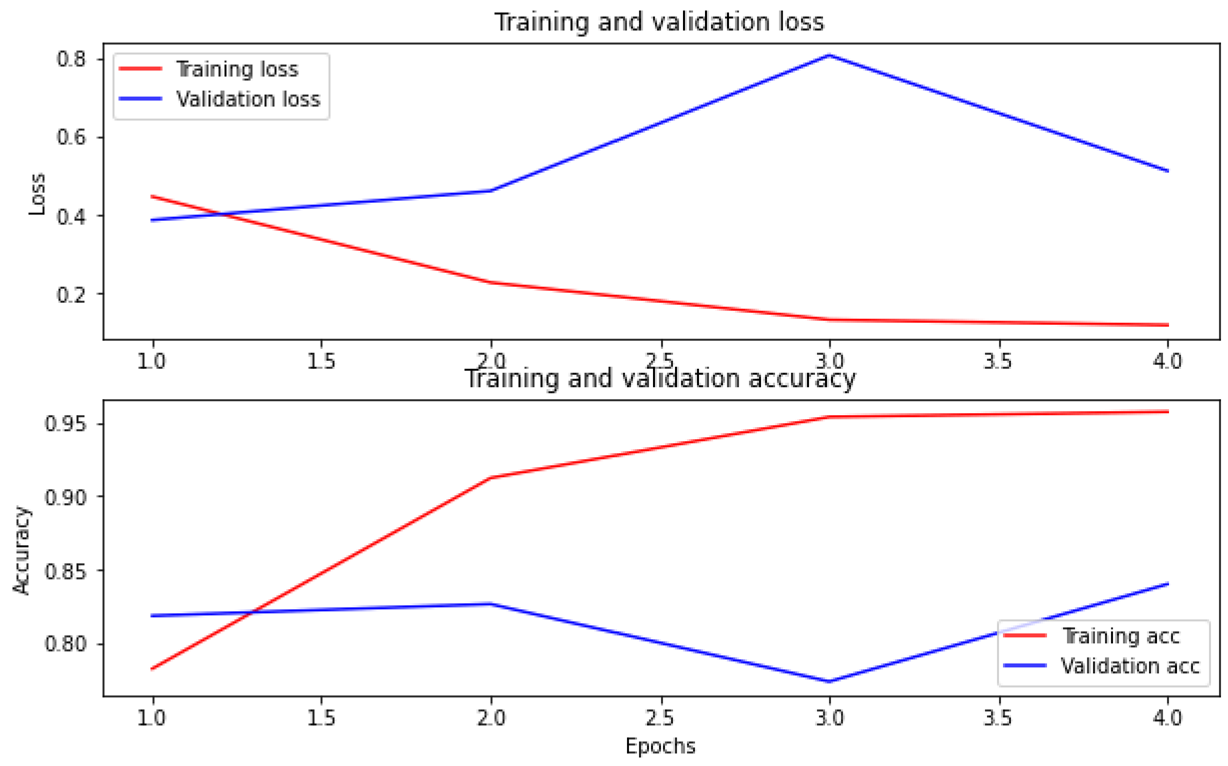
In [14]:
```
history_dict = history.history
print(history_dict.keys())
```

```
dict_keys(['loss', 'accuracy', 'val_loss', 'val_accuracy'])
```

In [15]:
```
print(history_dict)
```

```
{'loss': [0.44616207480430603, 0.22691364586353302, 0.13184481859207153, 0.1183848083
0192566], 'accuracy': [0.7824000120162964, 0.9122666716575623, 0.9538666605949402, 0.
9573333263397217], 'val_loss': [0.3863130211830139, 0.4606159031391144, 0.80674737691
87927, 0.5120529532432556], 'val_accuracy': [0.8184000253677368, 0.8263999819755554,
0.7735999822616577, 0.8399999737739563]}
```

In [16]:
```
acc = history_dict['accuracy']
val_acc = history_dict['val_accuracy']
loss = history_dict['loss']
val_loss = history_dict['val_loss']

epochs = range(1, len(acc) + 1)
fig = plt.figure(figsize=(10, 6))
fig.tight_layout()

plt.subplot(2, 1, 1)
# r is for "solid red line"
plt.plot(epochs, loss, 'r', label='Training loss')
# b is for "solid blue line"
plt.plot(epochs, val_loss, 'b', label='Validation loss')
plt.title('Training and validation loss')
# plt.xlabel('Epochs')
plt.ylabel('Loss')
plt.legend()

plt.subplot(2, 1, 2)
plt.plot(epochs, acc, 'r', label='Training acc')
plt.plot(epochs, val_acc, 'b', label='Validation acc')
plt.title('Training and validation accuracy')
plt.xlabel('Epochs')
plt.ylabel('Accuracy')
plt.legend(loc='lower right')

plt.show()
```



```
In [19]: # Predicts sentiment based on model
def predict(test_sentence):
    pred_input = tokenizer.encode(
        test_sentence,
        truncation=True,
        padding=True,
        return_tensors="tf"
    )

    tf_output = model.predict(pred_input, verbose=False)[0]
    tf_prediction = tf.nn.softmax(tf_output, axis=1)
    labels = ['negative','positive'] #(0:negative, 1:positive)
    label = tf.argmax(tf_prediction, axis=1)
    label = label.numpy()
    return labels[label[0]]

test_sentence = 'What a lovely and beautiful day'
predict(test_sentence)
```

```
Out[19]: 'positive'
```

```
In [20]: # Gets BERT sentiment and accuracy
test_df['bert_en_sentiment'] = test_df['review_en'].progress_apply(predict)
test_df.query('sentiment == bert_en_sentiment').shape[0]/test_df.shape[0]
```

```
  0%|          | 0/5000 [00:00<?, ?it/s]
```

```
Out[20]: 0.8872
```

```
In [21]: test_df
```

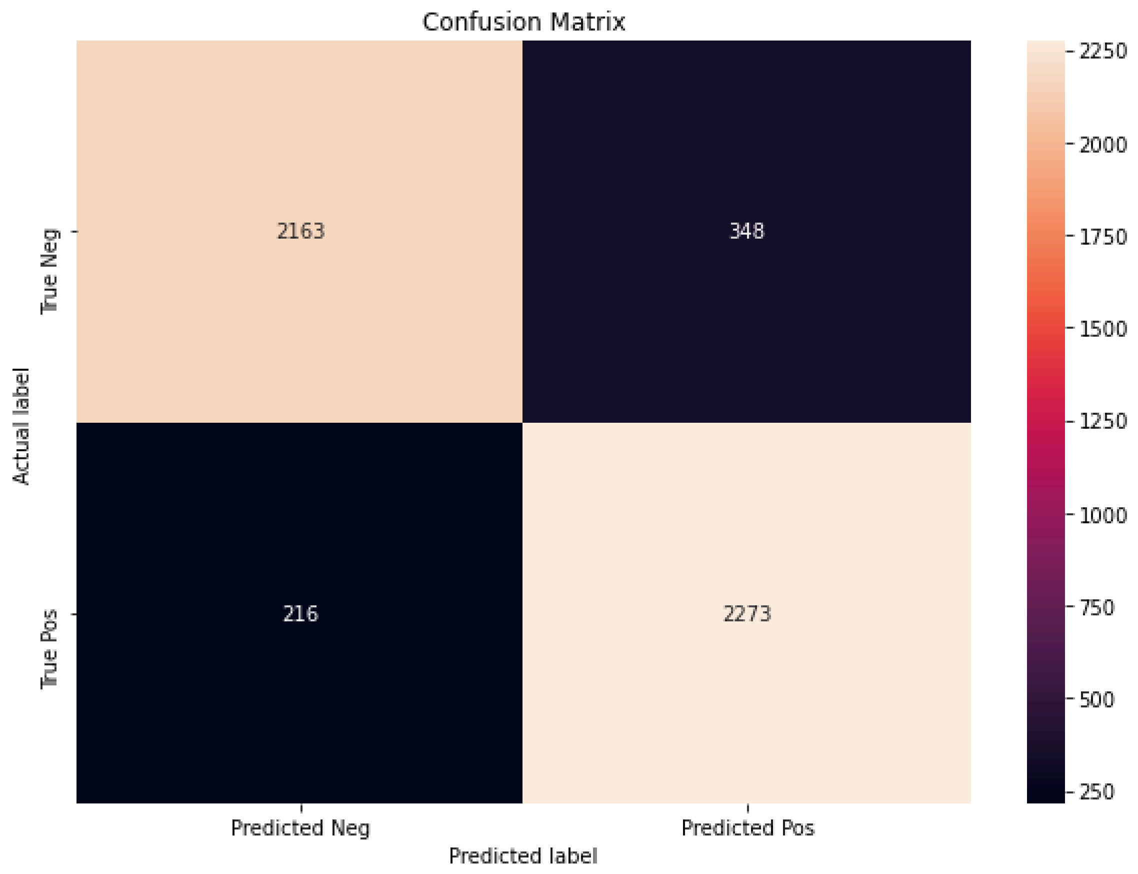
Out[21]:

	review_en	review_es	sentiment	bert_en_sentiment
42187	I had high hopes when I went into the theatre-...	Tenía grandes esperanzas cuando entré en el te...	negative	negative
10971	Black Day Blue Night was actually good modern ...	La noche azul del día negro fue en realidad un...	positive	positive
48213	*THIS REVIEW MAY CONTAIN SPOILERS... OR MAYBE ...	* Esta revisión puede contener spoilers ... o ...	negative	negative
15325	Despite the mysteriously positive reviews and ...	A pesar de las críticas misteriosamente positi...	negative	negative
49798	..."Flight of the Living Dead" sports producti...	... "Vuelo de los muertos vivos" Valores de pr...	negative	negative
...	...	...	...	...
22812	Boogie Nights was without a doubt the best fil...	Boogie Nights fue sin duda la mejor película d...	positive	positive
8422	After watching Tipping the Velvet by Sarah wat...	After watching Tipping the Velvet by Sarah wat...	positive	positive
11356	I caught this movie right in my eye when I was...	Cogí esta película a la derecha en mi ojo cuan...	positive	positive
48106	"October Sky" is a film that will steal your h...	"Octubre Sky" es una película que le robará el...	positive	positive
42219	My main comment on this movie is how Zwick was...	Mi comentario principal en esta película es ¿c...	negative	negative

5000 rows × 4 columns

In [22]:

```
from sklearn.metrics import confusion_matrix
import seaborn as sns
import matplotlib.pyplot as plt

conf_matrix = confusion_matrix(test_df['sentiment'], test_df['bert_en_sentiment'])

conf_matrix_df = pd.DataFrame(conf_matrix,
                              index=['True Neg', 'True Pos'],
                              columns=['Predicted Neg', 'Predicted Pos'])

plt.figure(figsize=(10,7))
sns.heatmap(conf_matrix_df, annot=True, fmt="d")
plt.title('Confusion Matrix')
plt.ylabel('Actual label')
plt.xlabel('Predicted label')
plt.show()
```



```
In [23]: bert_en_values = []
         for i in range(15):
           sampled_df = test_df.sample(n=1000, replace=True)
           sampled_df['bert_en_sentiment'] = sampled_df['review_en'].progress_apply(predict)
           acc = sampled_df.query('sentiment == bert_en_sentiment').shape[0]/sampled_df.shape[0
           bert_en_values.append(acc)
```

```
  0%|          | 0/1000 [00:00<?, ?it/s]
  0%|          | 0/1000 [00:00<?, ?it/s]
  0%|          | 0/1000 [00:00<?, ?it/s]
  0%|          | 0/1000 [00:00<?, ?it/s]
  0%|          | 0/1000 [00:00<?, ?it/s]
  0%|          | 0/1000 [00:00<?, ?it/s]
  0%|          | 0/1000 [00:00<?, ?it/s]
  0%|          | 0/1000 [00:00<?, ?it/s]
  0%|          | 0/1000 [00:00<?, ?it/s]
  0%|          | 0/1000 [00:00<?, ?it/s]
  0%|          | 0/1000 [00:00<?, ?it/s]
  0%|          | 0/1000 [00:00<?, ?it/s]
  0%|          | 0/1000 [00:00<?, ?it/s]
  0%|          | 0/1000 [00:00<?, ?it/s]
  0%|          | 0/1000 [00:00<?, ?it/s]
```

```
In [24]: bert_en_values
```

```
Out[24]: [0.895,
           0.877,
           0.897,
           0.903,
           0.891,
           0.889,
           0.89,
           0.877,
           0.887,
           0.89,
           0.899,
           0.886,
           0.895,
           0.881,
           0.887]
```

```
In [ ]:
```

Just Spanish

March 10, 2024

0.1 Sentiment Analysis: Spanish

```
[11]: # Splits train between train(40%) and validation(20%)
x_train_es, x_val_es, y_train_es, y_val_es = train_test_split(
                                                    train_df['review_es'],
                                                    train_df['sentiment'],
                                                    test_size=0.25,
                                                    stratify =␣
 ↪train_df['sentiment'],
                                                    random_state=42
                                                )
```

```
[12]: # Tokenizes and encodes the sentences for training
# tried google-bert/bert-base-multilingual-uncased and dccuchile/
 ↪bert-base-spanish-wwm-uncased
tokenizer = BertTokenizer.from_pretrained('dccuchile/
 ↪bert-base-spanish-wwm-uncased', do_lower_case=True)
max_len= 128

X_train_encoded_es = tokenizer.batch_encode_plus(x_train_es.tolist(),
                                              padding=True,
                                              truncation=True,
                                              max_length = max_len,
                                              return_tensors='tf')

X_val_encoded_es = tokenizer.batch_encode_plus(x_val_es.tolist(),
                                              padding=True,
                                              truncation=True,
                                              max_length = max_len,
                                              return_tensors='tf')
```

```
tokenizer_config.json:   0%|          | 0.00/310 [00:00<?, ?B/s]

vocab.txt:   0%|          | 0.00/248k [00:00<?, ?B/s]

special_tokens_map.json:   0%|          | 0.00/134 [00:00<?, ?B/s]

tokenizer.json:   0%|          | 0.00/486k [00:00<?, ?B/s]

config.json:   0%|          | 0.00/650 [00:00<?, ?B/s]
```

```
2024-03-09 23:21:21.216496: I
tensorflow/core/common_runtime/gpu/gpu_device.cc:1639] Created device
/job:localhost/replica:0/task:0/device:GPU:0 with 10174 MB memory:  -> device:
0, name: NVIDIA A30 MIG 2g.12gb, pci bus id: 0000:44:00.0, compute capability:
8.0
```

[13]:
```
# Gets model and fine-tunes
LEARNING_RATE = 5e-5

model = TFBertForSequenceClassification.from_pretrained('dccuchile/
 ↪bert-base-spanish-wwm-uncased', num_labels=2)

optimizer = tf.keras.optimizers.legacy.Adam(learning_rate=LEARNING_RATE)
loss = tf.keras.losses.SparseCategoricalCrossentropy(from_logits=True)
metric = tf.keras.metrics.SparseCategoricalAccuracy('accuracy')

model.compile(optimizer=optimizer,
              loss=loss,
              metrics=[metric])
```

```
tf_model.h5:   0%|          | 0.00/537M [00:00<?, ?B/s]
```

[14]:
```
# Cleans datasets for model fitting
train_dataset_es = tf.data.Dataset.from_tensor_slices((
    dict(X_train_encoded_es),
    y_train_es.apply(lambda x: True if x=='positive' else False)
))

test_dataset_es = tf.data.Dataset.from_tensor_slices((
    dict(X_val_encoded_es),
    y_val_es.apply(lambda x: True if x=='positive' else False)
))
```

[15]:
```
# Fits model based on data
model.fit(train_dataset_es.batch(16),
              epochs=4,
              batch_size=16,
              validation_data=test_dataset_es.batch(16))
```

```
Epoch 1/4

2024-03-09 23:21:43.811042: I
tensorflow/compiler/xla/stream_executor/cuda/cuda_blas.cc:606] TensorFloat-32
will be used for the matrix multiplication. This will only be logged once.

235/235 [==============================] - 57s 180ms/step - loss: 0.5030 -
accuracy: 0.7525 - val_loss: 0.4043 - val_accuracy: 0.8136
Epoch 2/4
235/235 [==============================] - 40s 171ms/step - loss: 0.2674 -
```

```
accuracy: 0.8891 - val_loss: 0.4028 - val_accuracy: 0.8224
Epoch 3/4
235/235 [==============================] - 40s 171ms/step - loss: 0.2015 -
accuracy: 0.9224 - val_loss: 0.6820 - val_accuracy: 0.7320
Epoch 4/4
235/235 [==============================] - 40s 172ms/step - loss: 0.1330 -
accuracy: 0.9528 - val_loss: 0.4745 - val_accuracy: 0.7920
```

```
[15]: <keras.src.callbacks.History at 0x7f3383b263d0>
```

```
[18]: #running 15 trials to get test accuracies
for i in range(15):
    sample_df = test_df.sample(n=1000, random_state=i)

    sample_df['es_sentiment'] = sample_df['review_es'].progress_apply(predict)


    correct_pred = sample_df.query('sentiment == es_sentiment').shape[0]
    accuracy = correct_pred / sample_df.shape[0]

    accuracies.append(accuracy)
    print("trial accuracy for", i, "=", accuracy)

average_accuracy_es = np.mean(accuracies)
print("mean spanish accuracy =",average_accuracy_es)
```

```
  0%|          | 0/1000 [00:00<?, ?it/s]

trial accuracy for 0 = 0.858

  0%|          | 0/1000 [00:00<?, ?it/s]

trial accuracy for 1 = 0.865

  0%|          | 0/1000 [00:00<?, ?it/s]

trial accuracy for 2 = 0.85

  0%|          | 0/1000 [00:00<?, ?it/s]

trial accuracy for 3 = 0.858

  0%|          | 0/1000 [00:00<?, ?it/s]

trial accuracy for 4 = 0.863

  0%|          | 0/1000 [00:00<?, ?it/s]

trial accuracy for 5 = 0.856

  0%|          | 0/1000 [00:00<?, ?it/s]

trial accuracy for 6 = 0.862

  0%|          | 0/1000 [00:00<?, ?it/s]
```

```
trial accuracy for 7 = 0.869

  0%|          | 0/1000 [00:00<?, ?it/s]

trial accuracy for 8 = 0.851

  0%|          | 0/1000 [00:00<?, ?it/s]

trial accuracy for 9 = 0.848

  0%|          | 0/1000 [00:00<?, ?it/s]

trial accuracy for 10 = 0.849

  0%|          | 0/1000 [00:00<?, ?it/s]

trial accuracy for 11 = 0.843

  0%|          | 0/1000 [00:00<?, ?it/s]

trial accuracy for 12 = 0.85

  0%|          | 0/1000 [00:00<?, ?it/s]

trial accuracy for 13 = 0.862

  0%|          | 0/1000 [00:00<?, ?it/s]

trial accuracy for 14 = 0.873
```

[18]:
```
0.8571333333333333
```

[26]:
```
# Save data
accuracies_df = pd.DataFrame(accuracies, columns=['accuracy'])
accuracies_df.to_csv('test_df_with_bert_only.csv')

test_df_es = pd.read_csv('test_df_with_bert_only.csv')
test_df_es
test_df_es = test_df_es.rename(columns={'Unnamed: 0': 'trials'})
```

[31]:
```
test_df
```

[31]:
```
                                                review_en  \
42187  I had high hopes when I went into the theatre-…
10971  Black Day Blue Night was actually good modern …
48213  *THIS REVIEW MAY CONTAIN SPOILERS… OR MAYBE …
15325  Despite the mysteriously positive reviews and …
49798  …"Flight of the Living Dead" sports producti…
…                                                    …
22812  Boogie Nights was without a doubt the best fil…
8422   After watching Tipping the Velvet by Sarah wat…
11356  I caught this movie right in my eye when I was…
48106  "October Sky" is a film that will steal your h…
42219  My main comment on this movie is how Zwick was…
```

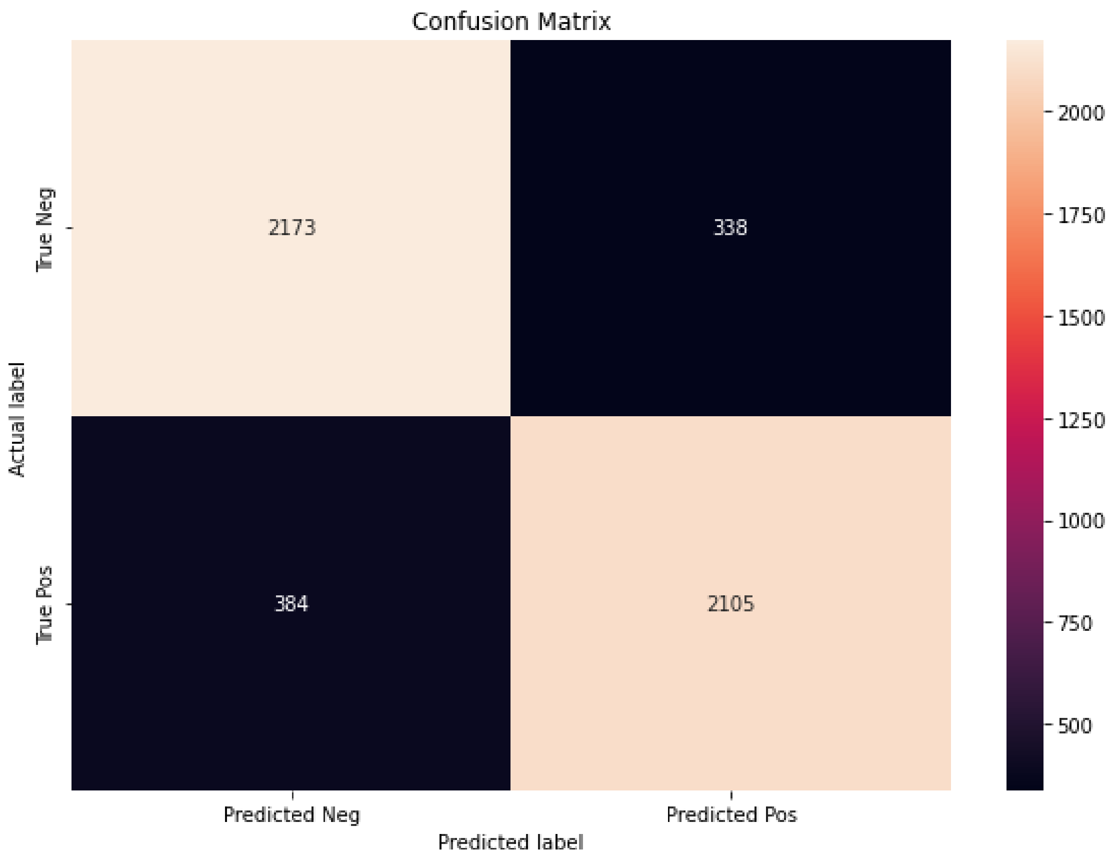
```
                                               review_es sentiment
42187  Tenía grandes esperanzas cuando entré en el te…  negative
10971  La noche azul del día negro fue en realidad un…  positive
48213  * Esta revisión puede contener spoilers … o …  negative
15325  A pesar de las críticas misteriosamente positi…  negative
49798  … "Vuelo de los muertos vivos" Valores de pr…  negative
…                                                    …        …
22812  Boogie Nights fue sin duda la mejor película d…  positive
8422   After watching Tipping the Velvet by Sarah wat…  positive
11356  Cogí esta película a la derecha en mi ojo cuan…  positive
48106  "Octubre Sky" es una película que le robará el…  positive
42219  Mi comentario principal en esta película es ¿c…  negative

[5000 rows x 3 columns]
```

[30]:
```python
test_df_es
```

[30]:
```
    trials  accuracy
0        0     0.858
1        1     0.865
2        2     0.850
3        3     0.858
4        4     0.863
5        5     0.856
6        6     0.862
7        7     0.869
8        8     0.851
9        9     0.848
10      10     0.849
11      11     0.843
12      12     0.850
13      13     0.862
14      14     0.873
```

[33]:
```python
from sklearn.metrics import confusion_matrix
import seaborn as sns
import matplotlib.pyplot as plt

# Gets BERT sentiment and accuracy
test_df['bert_es_sentiment'] = test_df['review_es'].progress_apply(predict)
test_df.query('sentiment == bert_es_sentiment').shape[0]/test_df.shape[0]
conf_matrix = confusion_matrix(test_df['sentiment'],␣
 ↪test_df['bert_es_sentiment'])

conf_matrix_df = pd.DataFrame(conf_matrix,
                              index=['True Neg', 'True Pos'],
                                    columns=['Predicted Neg', 'Predicted Pos'])

plt.figure(figsize=(10,7))
sns.heatmap(conf_matrix_df, annot=True, fmt="d")
plt.title('Confusion Matrix')
plt.ylabel('Actual label')
plt.xlabel('Predicted label')
plt.show()
```

```
  0%|          | 0/5000 [00:00<?, ?it/s]
```



[27]:
```
test_df_es
```

[27]:
```
   trials  accuracy
0       0     0.858
1       1     0.865
2       2     0.850
3       3     0.858
4       4     0.863
5       5     0.856
```

```
6          6      0.862
7          7      0.869
8          8      0.851
9          9      0.848
10        10      0.849
11        11      0.843
12        12      0.850
13        13      0.862
14        14      0.873
```

```
In [1]: import numpy as np
        import pandas as pd
        import scipy.stats as stats
        import statsmodels

        import nltk
        from nltk.sentiment import SentimentIntensityAnalyzer

        from nltk.sentiment import SentimentIntensityAnalyzer
        import seaborn as sns
        import matplotlib.pyplot as plt
```

Statistical Tests:

- For our question, we can perform a hypothesis test between two groups - difference in means of the accuracies of NLTK on english and spanish text, and the difference in means of accuracies of BERT on the same.
- One sample will be from directly passing reviews in Spanish and English into NLTK's Sentiment Intensity Analyzer and taking the difference in accuracy, and our second sample will use the same method but instead using BERT's pretrained model. To get a good estimate of the accuracy of both models, we will run replacement sampling of size 1000, with 15 different samples.
- We believe that BERT will perform better than NLTK because the BERT model will be trained and fine tuned to our testing data, whereas we will only use the given NLTK model.

```
In [22]: nltk_en_values
```

```
Out[22]: [0.688,
          0.678,
          0.69,
          0.704,
          0.705,
          0.675,
          0.683,
          0.698,
          0.708,
          0.679,
          0.698,
          0.691,
          0.678,
          0.69,
          0.712]
```

```
In [23]: nltk_es_values
```

```
Out[23]: [0.649,
 0.666,
 0.648,
 0.667,
 0.645,
 0.646,
 0.656,
 0.65,
 0.657,
 0.658,
 0.662,
 0.686,
 0.648,
 0.657,
 0.648]
```

the accuracy results from bert

```
In [25]: bert_en_values = [0.895,
 0.877,
 0.897,
 0.903,
 0.891,
 0.889,
 0.89,
 0.877,
 0.887,
 0.89,
 0.899,
 0.886,
 0.895,
 0.881,
 0.887]
bert_es_values = [0.8581,0.8652,0.853,0.8584,0.8635,0.8566,0.8627,0.8698,0.8519,0.8481
```

```
In [26]: import statistics
```

EDA on the variance of each group

```
In [27]: print(statistics.variance(nltk_en_values))
print(statistics.variance(nltk_es_values))
print(statistics.variance(bert_en_values))
print(statistics.variance(bert_es_values))

0.00014259999999999992
0.00011860000000000021
5.725714285714296e-05
7.086881238095256e-05
```

```
In [28]: statistics.variance(bert_es_values) * 2

Out[28]: 0.00014173762476190512
```

```
In [29]: levene_statistic = stats.levene(nltk_en_values, nltk_es_values, bert_en_values, bert_e
         levene_statistic.pvalue
```

```
Out[29]: 0.31540351933199584
```

```
In [30]: levene_statistic.statistic
```

```
Out[30]: 1.207684518369143
```

The levene statistic shows that there isn't significant enough evidence to show that there are differences in the variance, since the p-value is greater than 0.05, we fail to reject the null hypothesis of the levene's test. this means that the groups have high homoscedasticity, which means the variances are similar

```
In [31]: print('average accuracy of nltk on english : ' + str(np.mean(nltk_en_values)))
         print('average accuracy of nltk on spanish : ' + str(np.mean(nltk_es_values)))
         print('average accuracy of bert on english : ' + str(np.mean(bert_en_values)))
         print('average accuracy of bert on spanish : ' + str(np.mean(bert_es_values)))
```

```
average accuracy of nltk on english : 0.6918000000000001
average accuracy of nltk on spanish : 0.6562
average accuracy of bert on english : 0.8896000000000001
average accuracy of bert on spanish : 0.8577313333333333
```

Two-way ANOVA test with the independent variables being the model and the language the the dependent variable being the accuracy

```
In [32]: import statsmodels.api as sm
         from statsmodels.formula.api import ols
```

```
In [33]: # Example data: NLP model, Language, and Performance scores
         data = {
             'NLP_model': ['NLTK'] * 30 + ['BERT'] * 30,
             'Language': ['English'] * 15 + ['Spanish'] * 15 + ['English'] * 15 + ['Spanish'] *
             'Performance': nltk_en_values + nltk_es_values + bert_en_values + bert_es_values
         }

         # Convert data to pandas DataFrame
         df = pd.DataFrame(data)
         print(df)

         # Perform two-way ANOVA
         formula = 'Performance ~ C(NLP_model) + C(Language) + C(NLP_model):C(Language)'
         model = ols(formula, data=df).fit()
         anova_table = sm.stats.anova_lm(model, typ=2)
```

```
print(anova_table)
```

```
   NLP_model Language  Performance
0       NLTK  English      0.68800
1       NLTK  English      0.67800
2       NLTK  English      0.69000
3       NLTK  English      0.70400
4       NLTK  English      0.70500
5       NLTK  English      0.67500
6       NLTK  English      0.68300
7       NLTK  English      0.69800
8       NLTK  English      0.70800
9       NLTK  English      0.67900
10      NLTK  English      0.69800
11      NLTK  English      0.69100
12      NLTK  English      0.67800
13      NLTK  English      0.69000
14      NLTK  English      0.71200
15      NLTK  Spanish      0.64900
16      NLTK  Spanish      0.66600
17      NLTK  Spanish      0.64800
18      NLTK  Spanish      0.66700
19      NLTK  Spanish      0.64500
20      NLTK  Spanish      0.64600
21      NLTK  Spanish      0.65600
22      NLTK  Spanish      0.65000
23      NLTK  Spanish      0.65700
24      NLTK  Spanish      0.65800
25      NLTK  Spanish      0.66200
26      NLTK  Spanish      0.68600
27      NLTK  Spanish      0.64800
28      NLTK  Spanish      0.65700
29      NLTK  Spanish      0.64800
30      BERT  English      0.89500
31      BERT  English      0.87700
32      BERT  English      0.89700
33      BERT  English      0.90300
34      BERT  English      0.89100
35      BERT  English      0.88900
36      BERT  English      0.89000
37      BERT  English      0.87700
38      BERT  English      0.88700
39      BERT  English      0.89000
40      BERT  English      0.89900
41      BERT  English      0.88600
42      BERT  English      0.89500
43      BERT  English      0.88100
44      BERT  English      0.88700
45      BERT  Spanish      0.85810
46      BERT  Spanish      0.86520
47      BERT  Spanish      0.85300
48      BERT  Spanish      0.85840
49      BERT  Spanish      0.86350
50      BERT  Spanish      0.85660
51      BERT  Spanish      0.86270
52      BERT  Spanish      0.86980
53      BERT  Spanish      0.85190
54      BERT  Spanish      0.84810
55      BERT  Spanish      0.84911
```

```
56      BERT  Spanish      0.84312
57      BERT  Spanish      0.85130
58      BERT  Spanish      0.86214
59      BERT  Spanish      0.87300
                            sum_sq    df            F        PR(>F)
C(NLP_model)              0.597996   1.0  6143.907629  6.170040e-59
C(Language)               0.017070   1.0   175.380844  6.789181e-19
C(NLP_model):C(Language)  0.000052   1.0     0.536421  4.669741e-01
Residual                  0.005451  56.0          NaN           NaN
```

- The results of this ANOVA test indicate that there is significant evidence that both the NLP model and the language individually impact the accuracy levels to a significant extent, since both p-values were extremely low.
- With the choice of model explaining about 61% of the variance in our accuracy outcomes, and choice of language explaining around 19%. However, the p-value of the joint model and language choice was relatively large, thusly not significant.
- This can be interpreted as neither model performing better than the other on only one language.

T-test between Spanish and English performance within each NLP model and between each NLP model within each language.

In [35]:
```
def ttest(X_1, X_2):
    t_statistic, p_value = stats.ttest_rel(X_1, X_2)

    # Output results
    print("Paired t-test results:")
    print("T-statistic:", t_statistic)
    print("P-value:", p_value)

    # Interpret results
    alpha = 0.05
    if p_value < alpha:
        print("Reject null hypothesis: There is a significant difference in accuracies
    else:
        print("Fail to reject null hypothesis: There is no significant difference in a

print(ttest(nltk_en_values, nltk_es_values))
print(ttest(bert_en_values, bert_es_values))
print(ttest(nltk_en_values, bert_en_values))
print(ttest(nltk_es_values, bert_es_values))
```

```
Paired t-test results:
T-statistic: 8.470247564897582
P-value: 7.00010101220569e-07
Reject null hypothesis: There is a significant difference in accuracies.
None
Paired t-test results:
T-statistic: 9.1067718153829
P-value: 2.9383496627066053e-07
Reject null hypothesis: There is a significant difference in accuracies.
None
Paired t-test results:
T-statistic: -57.41522397544252
P-value: 5.076521071604921e-18
Reject null hypothesis: There is a significant difference in accuracies.
None
Paired t-test results:
T-statistic: -47.19061047330478
P-value: 7.801600968013823e-17
Reject null hypothesis: There is a significant difference in accuracies.
None
```

- **These results show that there is a significant difference between each pair of categories.**
- **As the large positive T-value and small p-value in the first test indicates that NLTK performed significantly better on the English over Spanish text.**
- **The large positive T-value and small p-value in the second test indicates that BERT performed significantly better on the English over Spanish text.**
- **The large negative T-value and small p-value in the third test indicates that NLTK performed significantly worse than BERT on the English text.**
- **The large negative T-value and small p-value in the fourth test indicates that NLTK performed significantly worse than BERT on the Spanish text.**

Conclusion:

- **The trained BERT model significantly outperformed the NLTK model on both the English and Spanish texts, while both BERT and NLTK did statistically better on the English over Spanish.**

Future work:

- For future work, we can explore other languages aside from English and Spanish while testing the performance of both the BERT and NLTK model. If we wanted to continue analyzing the performance of the models in the context of languages, we could potentially do an analysis that includes more languages, even comparing those autotranslated versus manually done so.
- Similarly there is an opportunity to open up this exploration to include more NLP models. The Friedman hypothesis test would work well for this as it has less assumptions than

ANOVA, however, in order to use it, we would then need the testing rows to be held constant for each model, for each sample.