Technical Report: Final Project DS 5110: Introduction to Data Management and Processing

Team Members: Colin Johnson
Khoury College of Computer Sciences
Data Science Program
johnson.colin@northeastern.edu

December 10, 2024

Contents

1 Introduction

In this project I am investigating the Stanford Movie Dataset which includes a split of positive and negative train and test data. The goal of this project is to make a model that can predict the sentiment score of reviews by using PyTorch Machine Learning.

2 Literature Review

Other projects of the same time use varying types of models. Some use RNN model which is more standard however a lot use CNN too. Models similar to mine found similar accuracy results depending on how well the data is preprocessed.

3 Methodology

I found the RNN models to be lacking is depth when I applied it to my dataset, I would have lower train accuracy. After applying CNN the model fit much better.

3.1 Data Collection

I collected the data by using os and extracted the tarfile using the tarfile import. I split the data into two separate dataset, one with the training data and one with the testing data, contain both all the negative and positive labels.

3.2 Data Preprocessing

Processed by using re to remove HTML labels and punctuations. I lemmanized the data to generalize the words filter specific words that were popular but useless in order to make the runtime faster (words like "movie", "film", etc.). Then I tokenized the reviews to only include important words.

After, I then took every token and matched it with vocab index values to encode words essentially. Words like ['love'] would turn into [200] for example here. I finally padded to the 90th percentile as for the model to work all data has to have equal length.

3.3 Analysis Techniques

Used a CNN model to relate the text to a sentiment score using the label. An example of how this model would work is if you had a tokenized review like (['love'], ['great], ['happy']). The model then would detect the label is 1 therefore it relates these terms to positive. After, if it sees this more and more then it relates it more to positive.

Parameters

- **vocab_size**: Size of the vocabulary.
- **embedding_dim**: Dimensionality of the word embeddings.
- **conv_config**: Configuration of convolutional layers, including:

  - **kernel_sizes**: List of kernel sizes (window sizes for the filters).
  - **num_channels**: Number of filters per kernel size.
- **output_size**: Number of output classes (pos, neg)
- **dropout**: Dropout probability to prevent overfitting the padded sequences

4 Results

Results found that the model worked with 85 percent accuracy after 10 epochs. Here

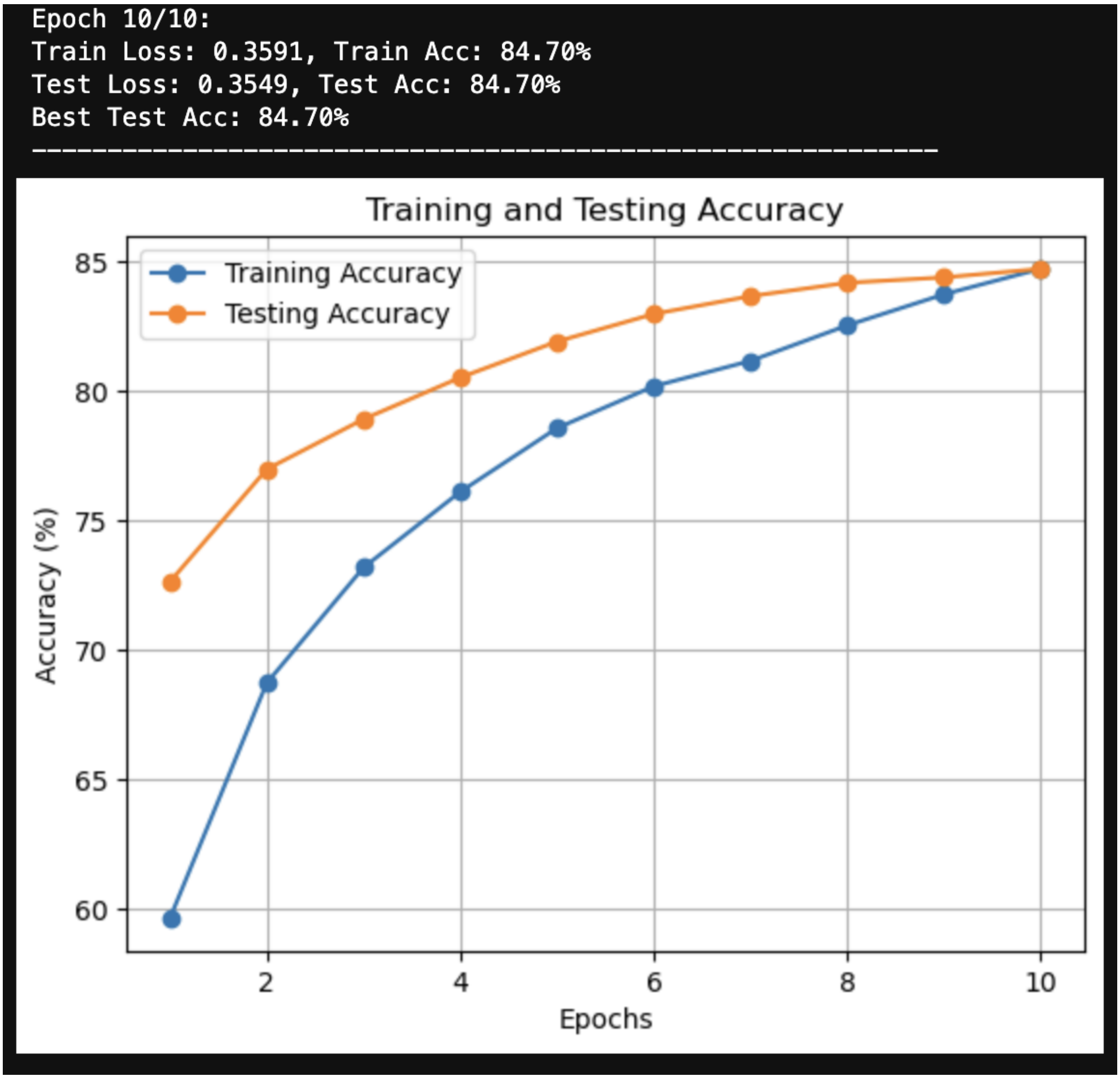
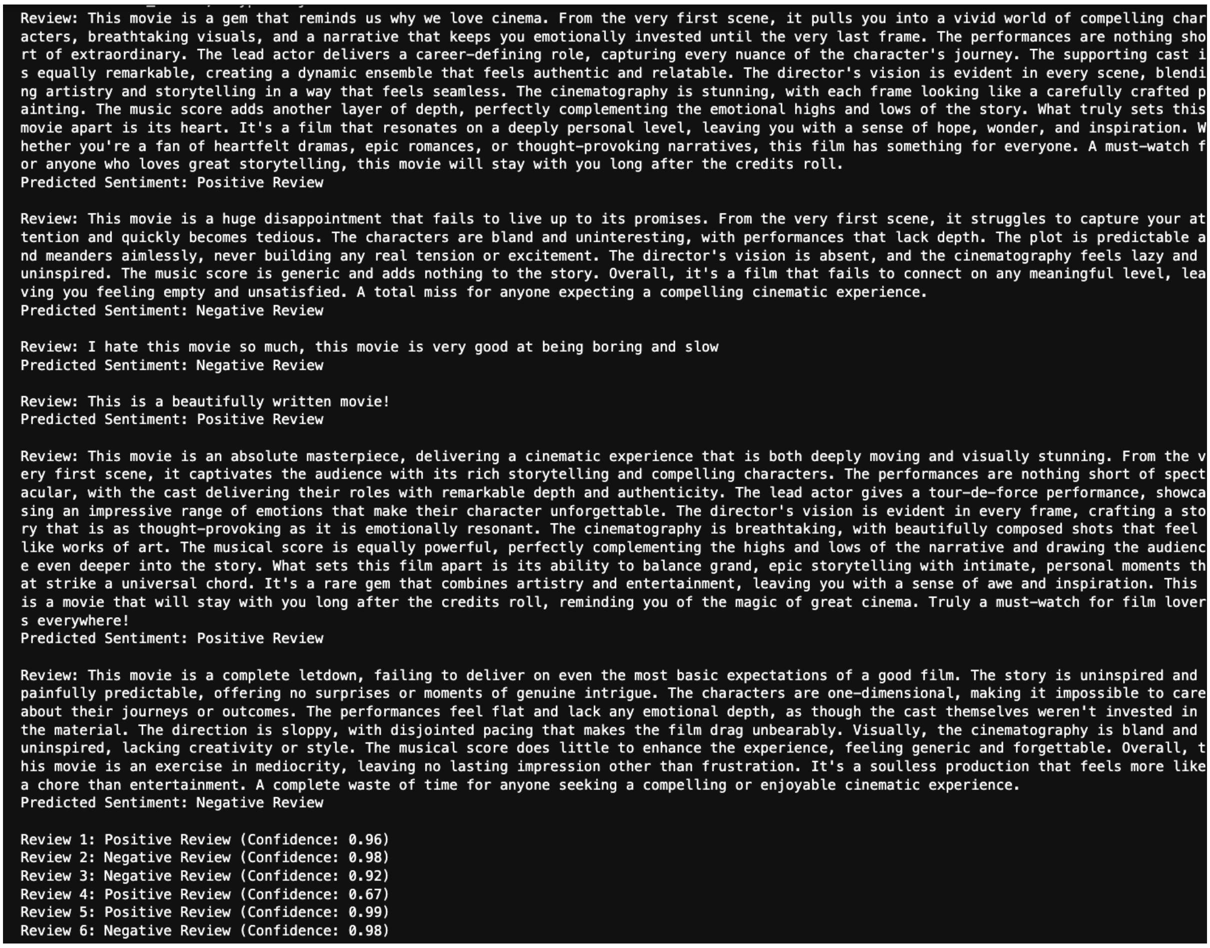


Figure 1: Accuracy after 10 epochs

shows my model is 85 percent accurate. I ran a test case for the model to figure how accurate it was and here they are



```
Review: This movie is a gem that reminds us why we love cinema. From the very first scene, it pulls you into a vivid world of compelling char
acters, breathtaking visuals, and a narrative that keeps you emotionally invested until the very last frame. The performances are nothing sho
rt of extraordinary. The lead actor delivers a career-defining role, capturing every nuance of the character's journey. The supporting cast i
s equally remarkable, creating a dynamic ensemble that feels authentic and relatable. The director's vision is evident in every scene, blendi
ng artistry and storytelling in a way that feels seamless. The cinematography is stunning, with each frame looking like a carefully crafted p
ainting. The music score adds another layer of depth, perfectly complementing the emotional highs and lows of the story. What truly sets this
movie apart is its heart. It's a film that resonates on a deeply personal level, leaving you with a sense of hope, wonder, and inspiration. W
hether you're a fan of heartfelt dramas, epic romances, or thought-provoking narratives, this film has something for everyone. A must-watch f
or anyone who loves great storytelling, this movie will stay with you long after the credits roll.
Predicted Sentiment: Positive Review

Review: This movie is a huge disappointment that fails to live up to its promises. From the very first scene, it struggles to capture your at
tention and quickly becomes tedious. The characters are bland and uninteresting, with performances that lack depth. The plot is predictable a
nd meanders aimlessly, never building any real tension or excitement. The director's vision is absent, and the cinematography feels lazy and
uninspired. The music score is generic and adds nothing to the story. Overall, it's a film that fails to connect on any meaningful level, lea
ving you feeling empty and unsatisfied. A total miss for anyone expecting a compelling cinematic experience.
Predicted Sentiment: Negative Review

Review: I hate this movie so much, this movie is very good at being boring and slow
Predicted Sentiment: Negative Review

Review: This is a beautifully written movie!
Predicted Sentiment: Positive Review

Review: This movie is an absolute masterpiece, delivering a cinematic experience that is both deeply moving and visually stunning. From the v
ery first scene, it captivates the audience with its rich storytelling and compelling characters. The performances are nothing short of spect
acular, with the cast delivering their roles with remarkable depth and authenticity. The lead actor gives a tour-de-force performance, showca
sing an impressive range of emotions that make their character unforgettable. The director's vision is evident in every frame, crafting a sto
ry that is as thought-provoking as it is emotionally resonant. The cinematography is breathtaking, with beautifully composed shots that feel
like works of art. The musical score is equally powerful, perfectly complementing the highs and lows of the narrative and drawing the audienc
e even deeper into the story. What sets this film apart is its ability to balance grand, epic storytelling with intimate, personal moments th
at strike a universal chord. It's a rare gem that combines artistry and entertainment, leaving you with a sense of awe and inspiration. This
is a movie that will stay with you long after the credits roll, reminding you of the magic of great cinema. Truly a must-watch for film lover
s everywhere!
Predicted Sentiment: Positive Review

Review: This movie is a complete letdown, failing to deliver on even the most basic expectations of a good film. The story is uninspired and
painfully predictable, offering no surprises or moments of genuine intrigue. The characters are one-dimensional, making it impossible to care
about their journeys or outcomes. The performances feel flat and lack any emotional depth, as though the cast themselves weren't invested in
the material. The direction is sloppy, with disjointed pacing that makes the film drag unbearably. Visually, the cinematography is bland and
uninspired, lacking creativity or style. The musical score does little to enhance the experience, feeling generic and forgettable. Overall, t
his movie is an exercise in mediocrity, leaving no lasting impression other than frustration. It's a soulless production that feels more like
a chore than entertainment. A complete waste of time for anyone seeking a compelling or enjoyable cinematic experience.
Predicted Sentiment: Negative Review

Review 1: Positive Review (Confidence: 0.96)
Review 2: Negative Review (Confidence: 0.98)
Review 3: Negative Review (Confidence: 0.92)
Review 4: Positive Review (Confidence: 0.67)
Review 5: Positive Review (Confidence: 0.99)
Review 6: Negative Review (Confidence: 0.98)
```

Figure 2: Test case for model

5 Discussion

The results show there is some work in progress still with my model. For example, notice how review 4 (the shortest one) has confidence value of .67. This implies shorter reviews are less accurate which is why the accuracy for the model is 85 percent. The other figures represent the model working well especially for longer more informed reviews. Compared to other works well others have better accuracy due to stronger preprocessing so if I had the time to change any part it would be the preprocessing and make it clearer which review is positive or not for shorter reviews. However, there is also just error with the model in finding reviews here. This is because there is sometimes so little info it will be very hard to fine-tune to the point that you can get it perfect every time.

6 Conclusion

Found that CNN works well for Sentiment Analysis compared to RNN models at least for my dataset. Model accuracy is highly dataset related as despite there being more work for tokenization, if the dataset is not large enough you can get poor performance due to lack there of references for the model to work with. This is also a limitation as smaller datasets will not be able to get all the nuances of reviews that come their way, causing for lower accuracy. For future research, I can implement a neutral section where

I investigate with neutral reviews too, hopefully making for better accuracy due to more categories for the model to reference when building. Dealing with smaller less nuanced reviews as well, with more research and time it is possible to make the model be able to investigate vague reviews as well.

7 References

References

here

here

A Appendix A: Code

```python
def tokenize_with_vocab(text, vocab):
    '''
    Tokenizing the reviews, removing common and useless words, etc.
    '''
    words_remove = ['movie', 'film', 'director', 'plays', 'horror', 'comedy', 'watching', 'seen', 'people', 'guy', 's', 'time', 'second', 'book']
    vocab = filter_vocab(vocab, words_remove)
    tokenized_reviews = []
    for doc in nlp.pipe(text, batch_size=1000, disable=["parser", "ner", "tagger"]):
        cleaned_text = remove_html_tags(doc.text)
        cleaned_text = remove_punctuation(doc.text)
        lemmas = spacy_lemmatization(cleaned_text)
        tokens = [
            lemma.lower() for lemma in lemmas
            if lemma.lower() in vocab and lemma.lower() not in nlp.Defaults.stop_words
        ]
        tokenized_reviews.append(tokens)

    return tokenized_reviews
```

Listing 1: Preprocessing

```python
class  TextCNN(nn.Module):
    def __init__(self, vocab_size, embedding_dim, conv_config, output_size, dropout=0.5):
        super(TextCNN, self).__init__()


        self.conv_config = conv_config
        self.output_size = output_size
        self.dropout_p = dropout

        self.embedding = nn.Embedding(vocab_size, embedding_dim)
        #configuring the hidden layers, extracting features from the input nodes (reviews)
        self.convolutions = nn.ModuleList([
            nn.Sequential(
```

```
                nn.Conv1d(embedding_dim, self.conv_config['num_channels'], kernel_size=kernel),
                nn.ReLU(),
                nn.AdaptiveMaxPool1d(1)
            )
            for kernel in self.conv_config['kernel_sizes']
        ])

        self.dropout = nn.Dropout(self.dropout_p)
        self.linear = nn.Linear(
            self.conv_config['num_channels'] * len(self.conv_config['kernel_sizes']),
            self.output_size
        )

    def forward(self, input_seq):
        '''
        Forward pass for the TextCNN model.
        Returns the log probability with shape of output and batch size
        '''

        emb_out = self.embedding(input_seq).permute(0, 2, 1)


        conv_out = [conv(emb_out).squeeze(2) for conv in self.convolutions]

        concat_out = torch.cat(conv_out, dim=1)

        concat_out = self.dropout(concat_out)
        out = self.linear(concat_out)

        return F.log_softmax(out, dim=-1)

class SentimentDataset(Dataset):
    def __init__(self, sequences, labels):
        self.sequences = sequences
        self.labels = torch.tensor(labels, dtype=torch.long)

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

    def __getitem__(self, idx):
        return self.sequences[idx], self.labels[idx]
```

Listing 2: CNN Model

```
def train_model(model, train_loader, test_loader, criterion, optimizer, num_epochs=10):
    device='cpu'
    model = model.to(device)
    train_losses = []
    test_losses = []
    train_accuracies = []
    test_accuracies = []
    best_acc = 0.0

    for epoch in range(num_epochs):
        model.train()
```

```
        running_loss = 0.0
        correct = 0
        total = 0

        for inputs, labels in train_loader:
            inputs, labels = inputs.to(device), labels.to(device)
            optimizer.zero_grad()
            outputs = model(inputs)
            #comparing how much you lose compared to output target values
            loss = criterion(outputs, labels)
            loss.backward()
            optimizer.step()

            running_loss += loss.item()
            _, predicted = outputs.max(1)
            total += labels.size(0)
            correct += predicted.eq(labels).sum().item()


        epoch_loss = running_loss / len(train_loader)
        epoch_acc = 100. * correct / total
        train_losses.append(epoch_loss)
        train_accuracies.append(epoch_acc)


        model.eval()
        test_loss = 0.0
        correct = 0
        total = 0

        with torch.no_grad():
            for inputs, labels in test_loader:
                inputs, labels = inputs.to(device), labels.to(device)
                outputs = model(inputs)
                loss = criterion(outputs, labels)
                #collect test loss & test values
                test_loss += loss.item()
                _, predicted = outputs.max(1)
                total += labels.size(0)
                correct += predicted.eq(labels).sum().item()


        test_loss = test_loss / len(test_loader)
        test_acc = 100. * correct / total
        test_losses.append(test_loss)
        test_accuracies.append(test_acc)

        if test_acc > best_acc:
            best_acc = test_acc
            torch.save(model.state_dict(), 'best_model.pth')

        print(f'Epoch {epoch+1}/{num_epochs}:')
        print(f'Train Loss: {epoch_loss:.4f}, Train Acc: {epoch_acc:.2f}%')
        print(f'Test Loss: {test_loss:.4f}, Test Acc: {test_acc:.2f}%')
        print(f'Best Test Acc: {best_acc:.2f}%')
        print('-' * 60)
```

```python
        epochs = range(1, len(train_accuracies) + 1)
        '''
        Plot using matplotlib to display the change in accuracy over multiple epoch's
        '''
        plt.plot(epochs, train_accuracies, label='Training Accuracy', marker='o')
        plt.plot(epochs, test_accuracies, label='Testing Accuracy', marker='o')
        plt.xlabel('Epochs')
        plt.ylabel('Accuracy (%)')
        plt.title('Training and Testing Accuracy')
        plt.legend()
        plt.grid(True)
        plt.show()
```

Listing 3: Training Model

```python
review = ["This movie is a gem that reminds us why we love cinema. From the very first scene, it pulls you into a vivid world of compelling characters, breathtaking visuals, and a narrative that keeps you emotionally invested until the very last frame. \
The performances are nothing short of extraordinary. The lead actor delivers a career-defining role, capturing every nuance of the character's journey. The supporting cast is equally remarkable, creating a dynamic ensemble that feels authentic and relatable. \
The director's vision is evident in every scene, blending artistry and storytelling in a way that feels seamless. The cinematography is stunning, with each frame looking like a carefully crafted painting. The music score adds another layer of depth, perfectly \
complementing the emotional highs and lows of the story. What truly sets this movie apart is its heart. It's a film that resonates on a deeply personal level, leaving you with a sense of hope, wonder, and inspiration. Whether you're a fan of heartfelt dramas, \
epic romances, or thought-provoking narratives, this film has something for everyone. A must-watch for anyone who loves great storytelling, this movie will stay with you long after the credits roll.",
"This movie is a huge disappointment that fails to live up to \
its promises. From the very first scene, it struggles to capture your attention and quickly becomes tedious. The characters are bland and uninteresting, with performances that lack depth. The plot is predictable and meanders aimlessly, never building any real tension or \
excitement. The director's vision is absent, and the cinematography feels lazy and uninspired. The music score is generic and adds nothing to the story. Overall, it's a film that fails to connect on any meaningful level, leaving you feeling empty and unsatisfied. \
A total miss for anyone expecting a compelling cinematic experience.",
"I hate this movie so much, this movie is very good at being boring and slow",
"This is a beautifully written movie!",
"This movie is an absolute masterpiece, delivering a cinematic experience that is both deeply moving and visually stunning. From the very first scene, it captivates the audience with its rich storytelling and compelling characters. The performances are nothing short of spectacular, with the cast delivering their roles with remarkable depth and authenticity. The lead actor gives a tour-de-force performance, showcasing an impressive range of emotions that make their character unforgettable. \
```

```
The director's vision is evident in every frame, crafting a story that is as thought-provoking as it is emotionally resonant. The cinematography is breathtaking, with beautifully composed shots that feel like works of art. The musical score is equally powerful, perfectly complementing the highs and lows of the narrative and drawing the audience even deeper into the story. \
What sets this film apart is its ability to balance grand, epic storytelling with intimate, personal moments that strike a universal chord. It's a rare gem that combines artistry and entertainment, leaving you with a sense of awe and inspiration. This is a movie that will stay with you long after the credits roll, reminding you of the magic of great cinema. Truly a must-watch for film lovers everywhere!",
"This movie is a complete letdown, failing to deliver on even the most basic expectations of a good film. The story is uninspired and painfully predictable, offering no surprises or moments of genuine intrigue. The characters are one-dimensional, making it impossible to care about their journeys or outcomes. \
The performances feel flat and lack any emotional depth, as though the cast themselves weren't invested in the material. The direction is sloppy, with disjointed pacing that makes the film drag unbearably. Visually, the cinematography is bland and uninspired, lacking creativity or style. The musical score does little to enhance the experience, feeling generic and forgettable. \
Overall, this movie is an exercise in mediocrity, leaving no lasting impression other than frustration. It's a soulless production that feels more like a chore than entertainment. A complete waste of time for anyone seeking a compelling or enjoyable cinematic experience."]
'''
Test case
'''
df = pd.DataFrame(review, columns=['review'])

df['tokenized_review'] = tokenize_with_vocab(df['review'],vocab)
print(df['tokenized_review'])
df['review_ids'] = df['tokenized_review'].apply(lambda tokens: tokenize_id(tokens,vocab))
df['review_length'] = df['tokenized_review'].apply(len)
sequence_df = [torch.tensor(ids, dtype=torch.long) for ids in df['review_ids']]
max_length = int(df['review_length'].max())
padded = pad_sequences(sequence_df, max_length)

model.eval()
with torch.no_grad():
    output = model(padded)

    predicted_class = torch.argmax(output, dim=1).tolist()

label_map = {0: "Negative Review", 1: "Positive Review"}
predicted_labels = [label_map[p] for p in predicted_class]

for i, review in enumerate(review):
    print(f"Review: {review}")
    print(f"Predicted Sentiment: {predicted_labels[i]}\n")
probabilities = torch.softmax(output, dim=1)
for i, prob in enumerate(probabilities):
```

```
    if prob[0] > prob[1]:
        sentiment = "Negative Review"
    else:
        sentiment = "Positive Review"
    print(f"Review {i + 1}: {sentiment} (Confidence: {prob.max().item()
   :.2f})")
```

Listing 4: Test Cases