**CPSC5900**

**Project Report:**

**Categorizing Sentiments on YouTube Comments**

**Mark B. Humphrey**

**April 27, 2025**

***Data is like clay. With the right tools, you can shape it into something meaningful.***

**--- John Tukey**

**Executive Summary**

I was given the task to use UnSloth, Colab, and related tools to complete a Machine Learning task.

**Introduction**

After some struggle with learning to use these tools, I found a large CSV file1. This is a CSV file with many records: 1,032,226, and 12 attributes.

**Objective: *I set my goal to create my code in Colab, to see if I could train on data to see if I could predict if the user input of comment-text had the sentiment of Positive, Neutral, or Negative.***

Thus, one attribute (column) in the CSV file among the 12 was Comment-Text and another attribute (column) was Sentiment.

**Methodology**

**Tools, Techniques, Technologies Used**

**Data Preparation**

I downloaded the CSV file. I created a new file with only the Comment Text column and the Sentiment column. Thereafter, I uploaded the new data file into Colab directory.

**Code Creation**

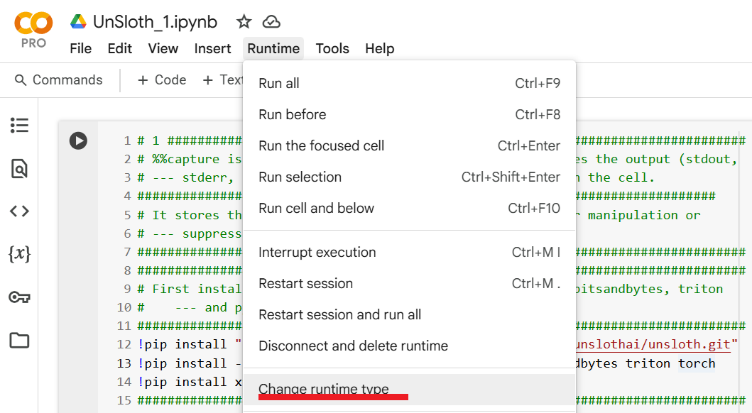
The code used to train the model in this project was created by modifying code from an online source2 to work with the aforementioned CSV file that was uploaded into Colab.

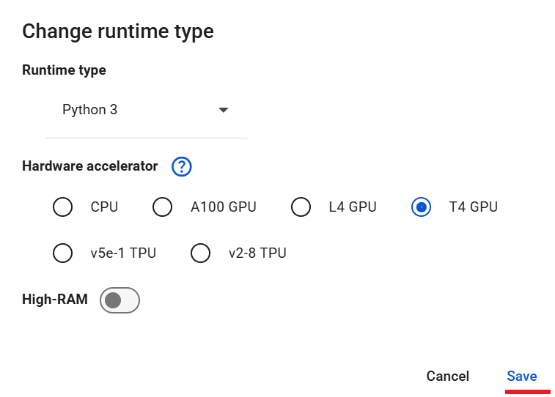
**Implementation And Development**

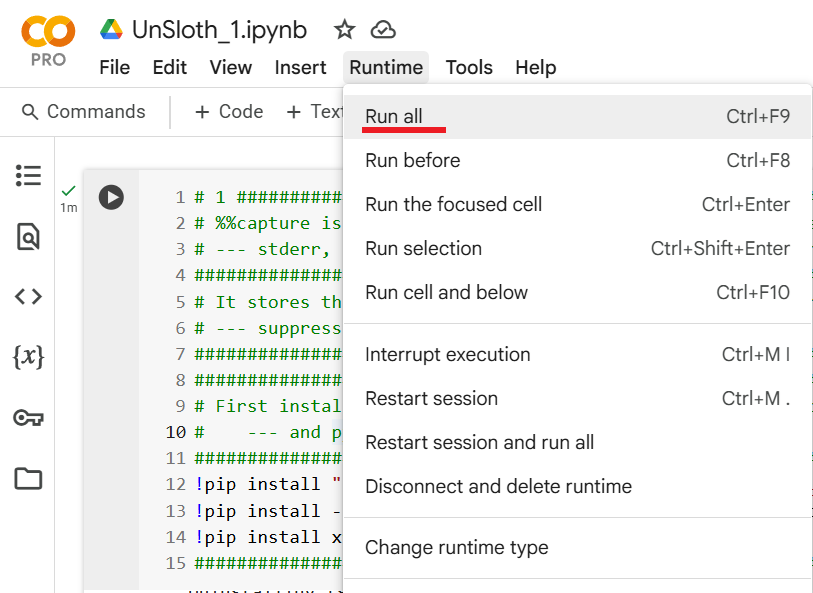
**How The Project Was Implemented and Development**

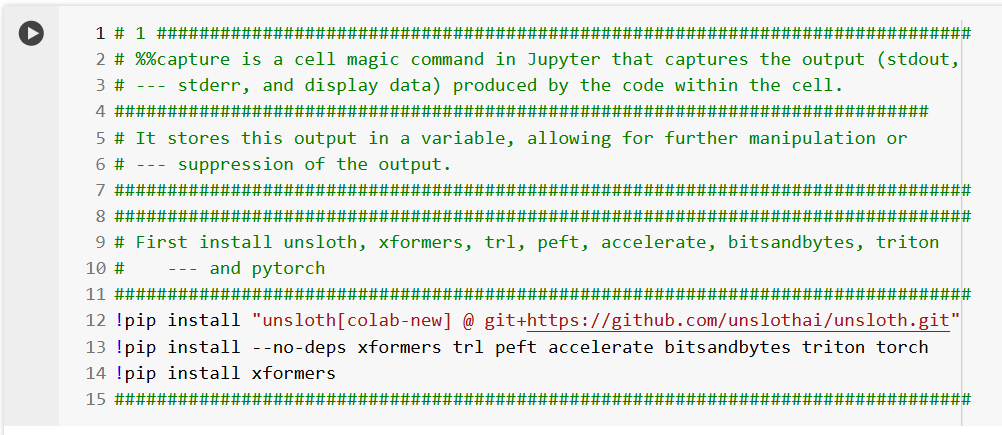
1. The CSV file was modified to have only CommentText and Sentiment attributes (columns).
2. The modified CSV file was uploaded into Colab folder.
3. The Colab file was created to contain code to be run, and the code was added to the cells in Colab.
4. The code was modified and run, continually, with debugging to have the final application in Colab.
5. To debug the code and test it in Colab, code in the final cell was run repeatedly to check comments to see if they would be rated Positive, Neutral, or Negative.

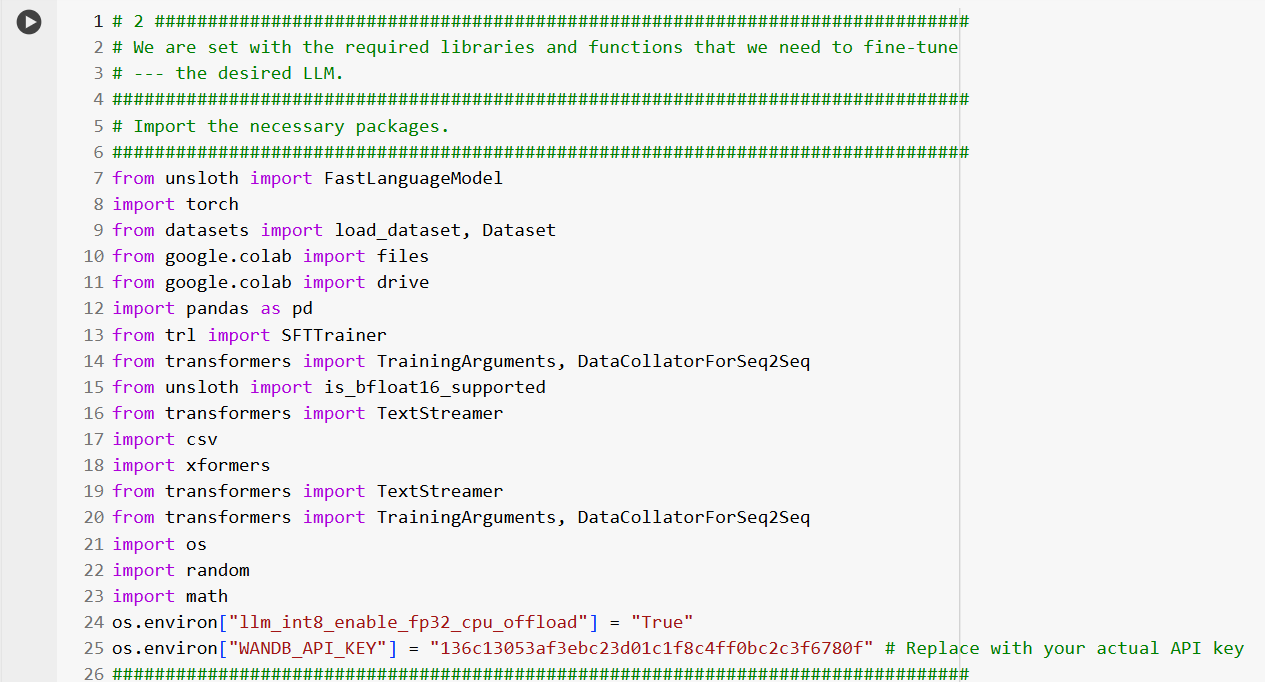
The following slides are from logging into Colab then running, and viewing the code:

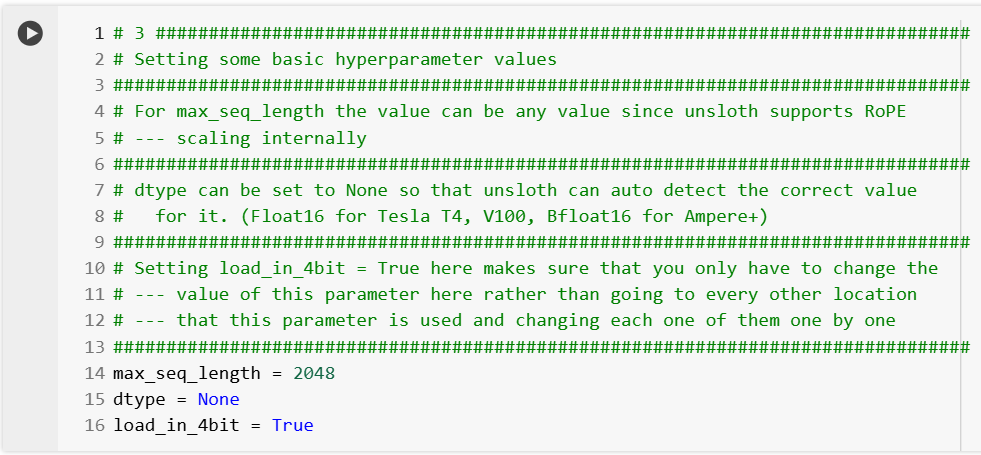


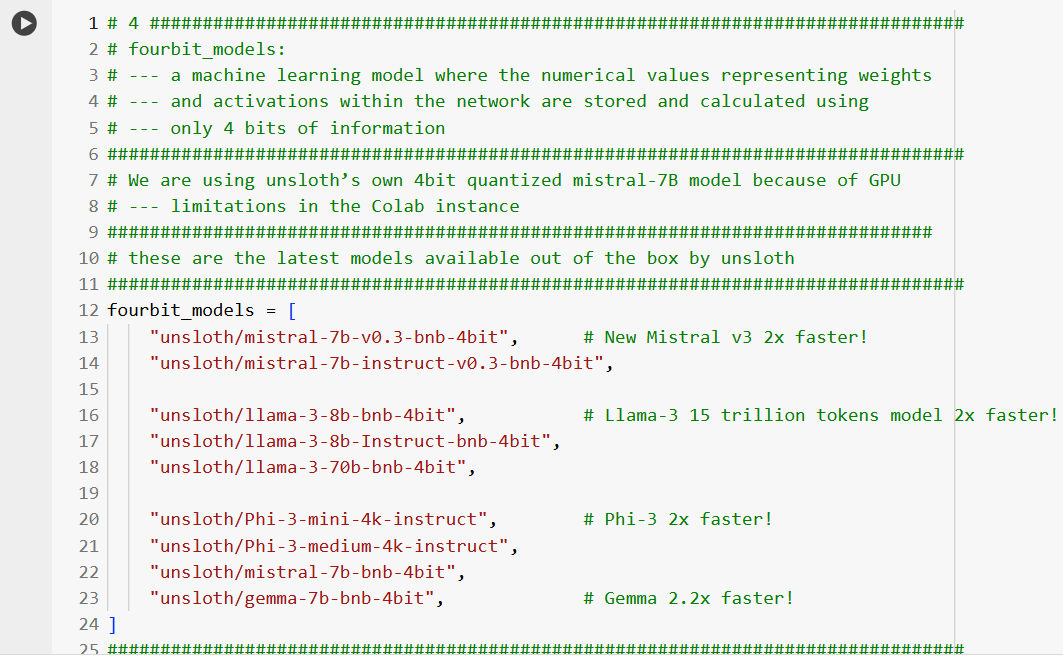


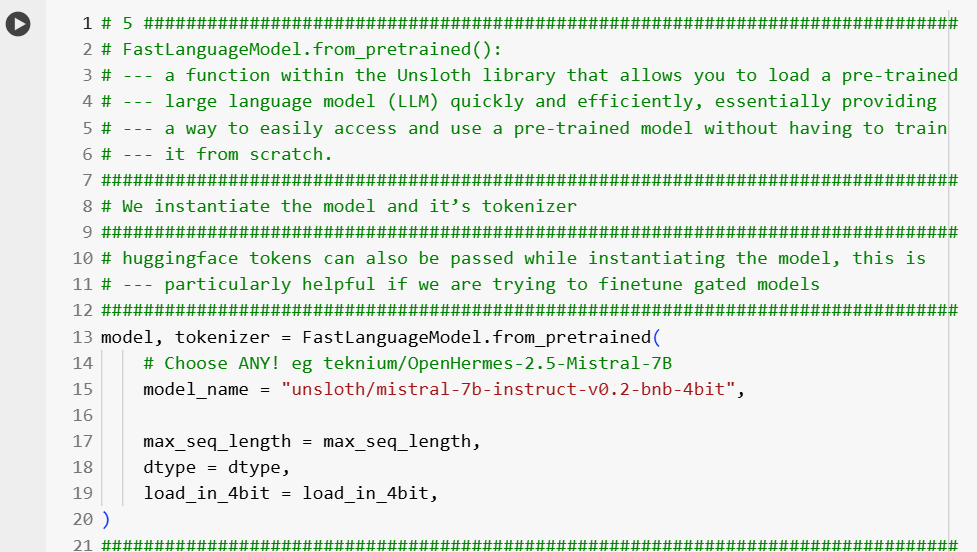




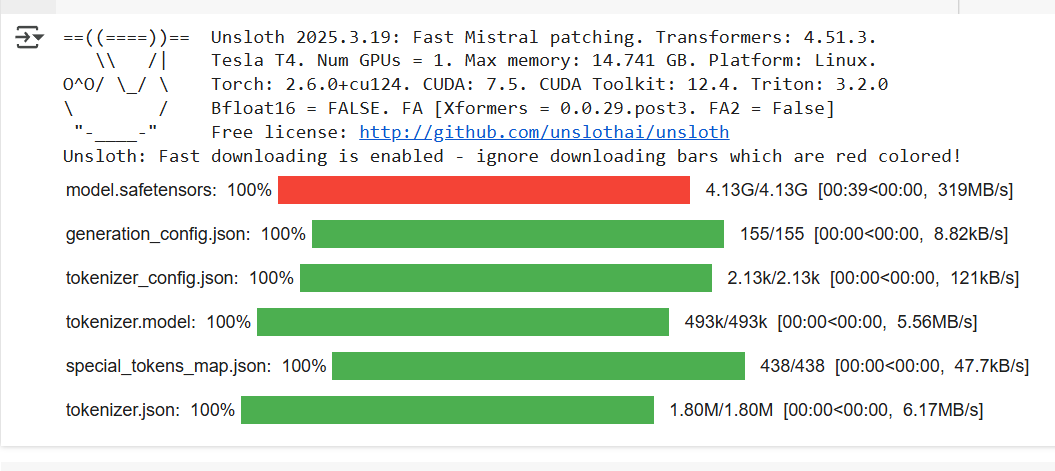


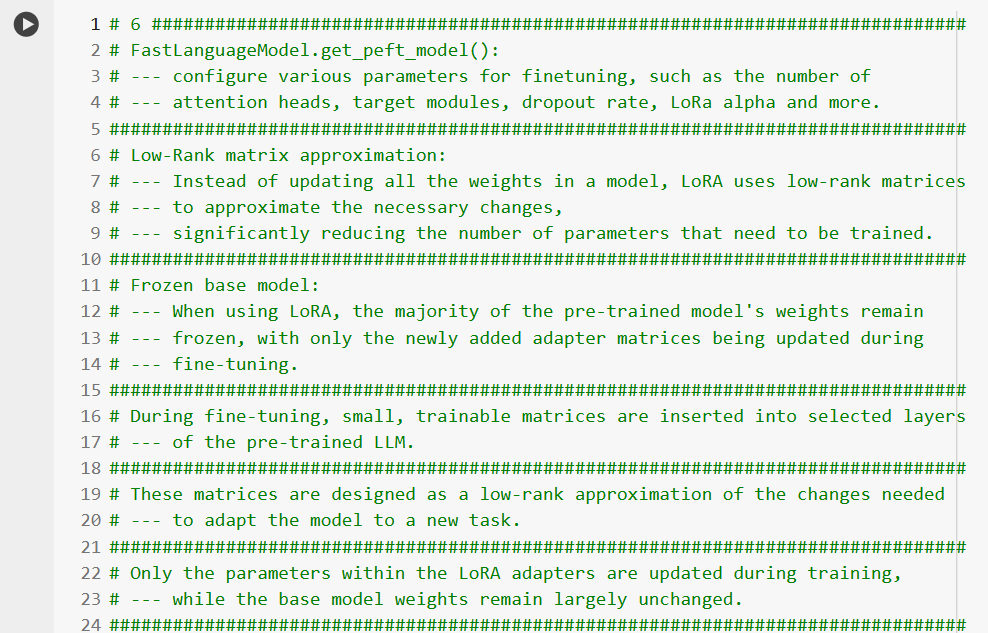


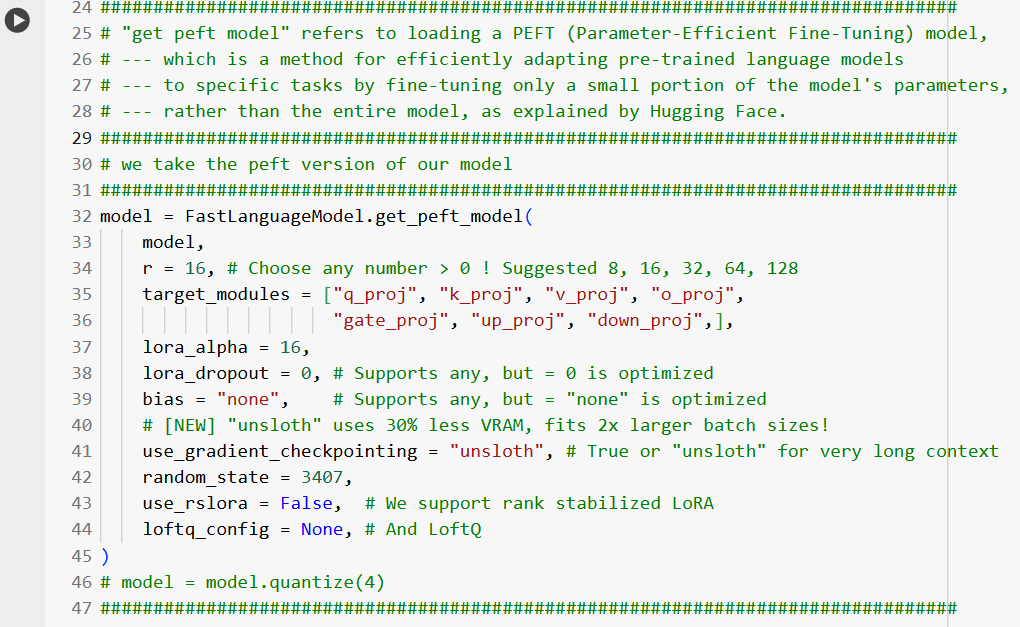


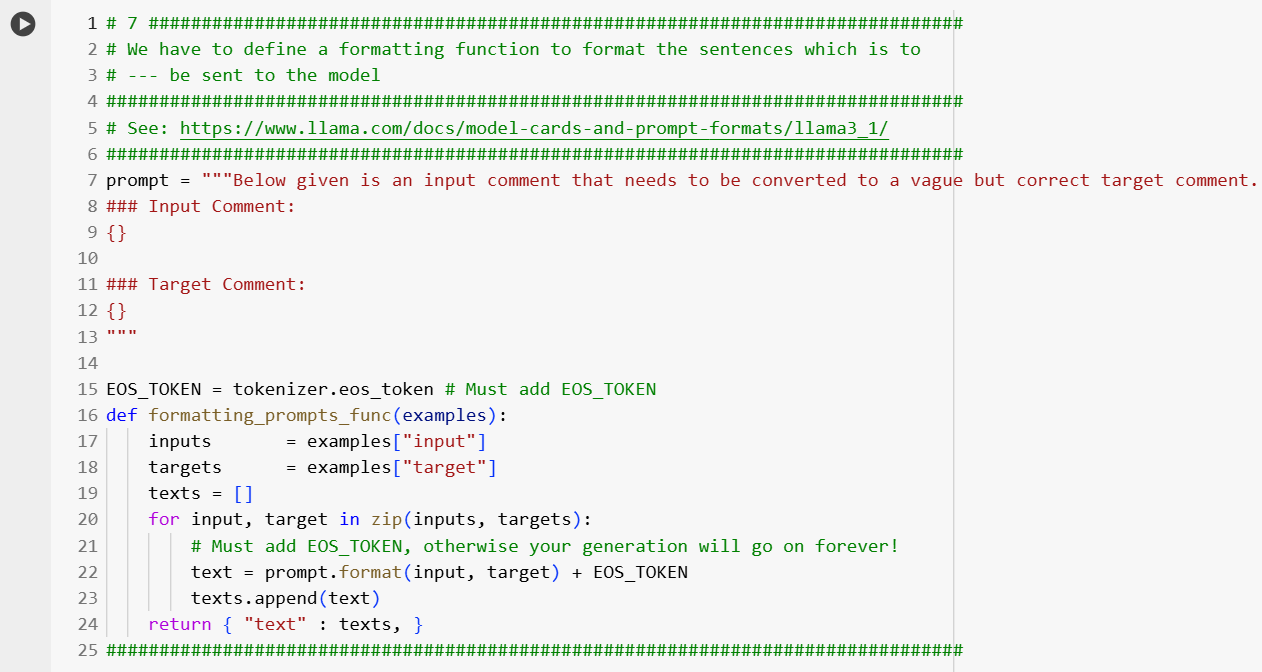


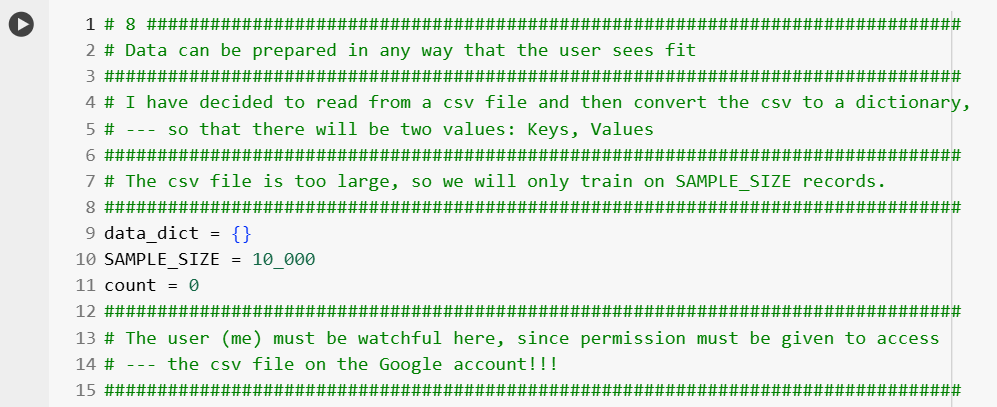
(Execution Details)

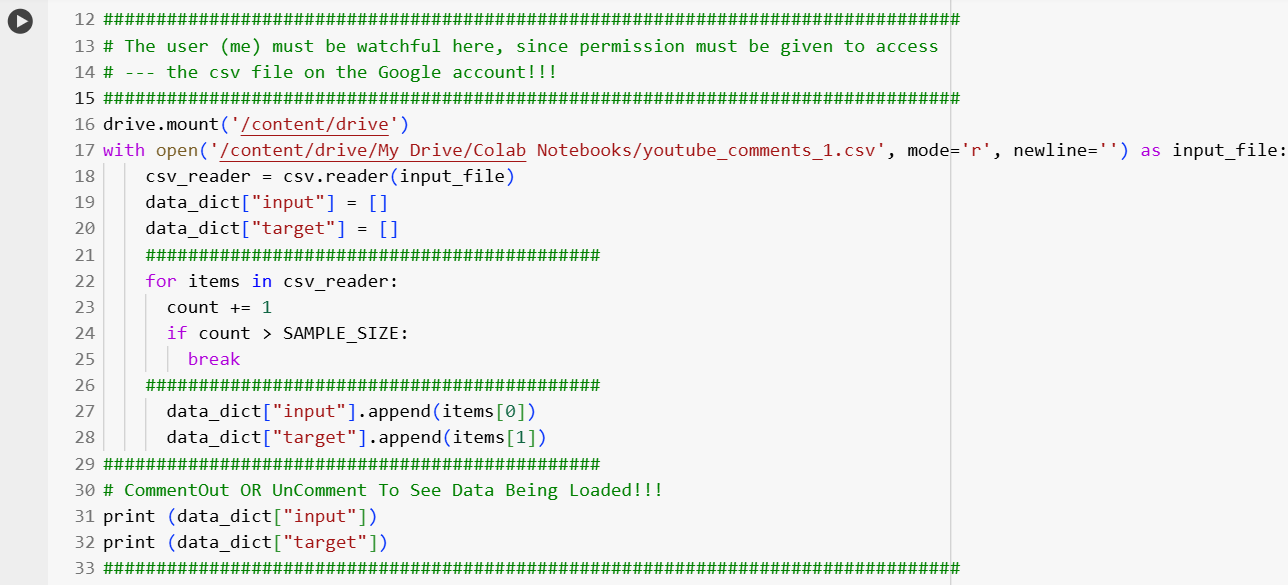




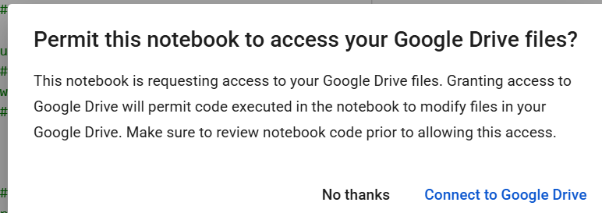


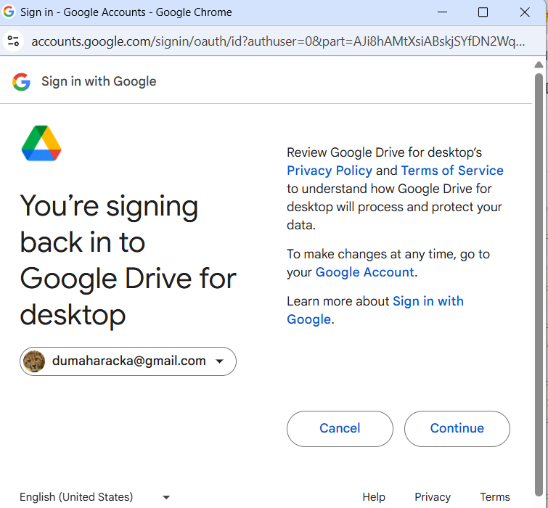
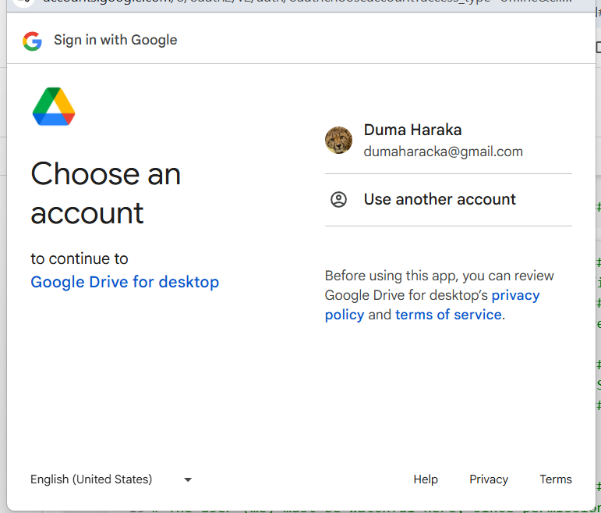


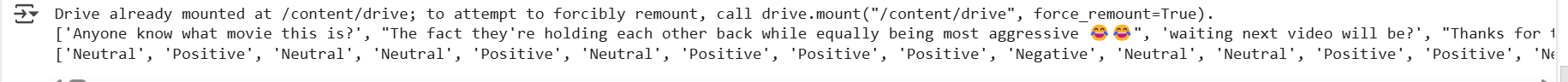


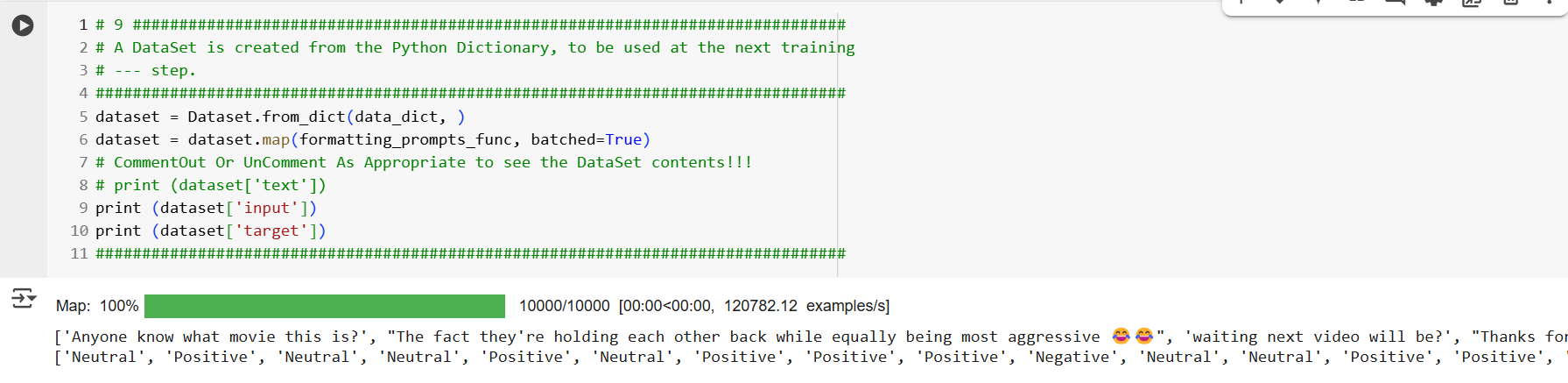


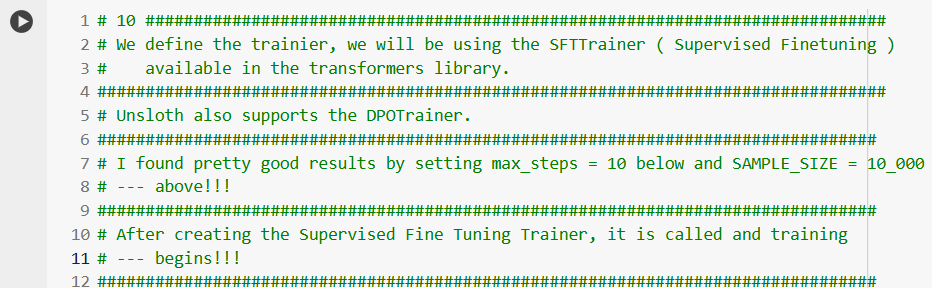
(Execution Details)





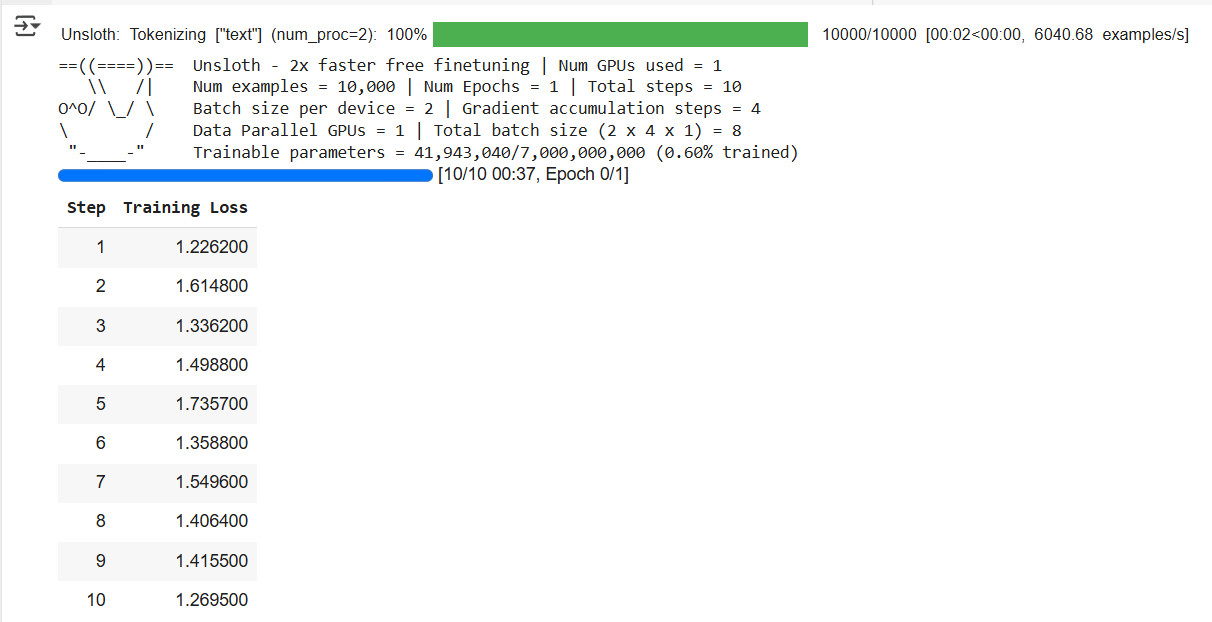


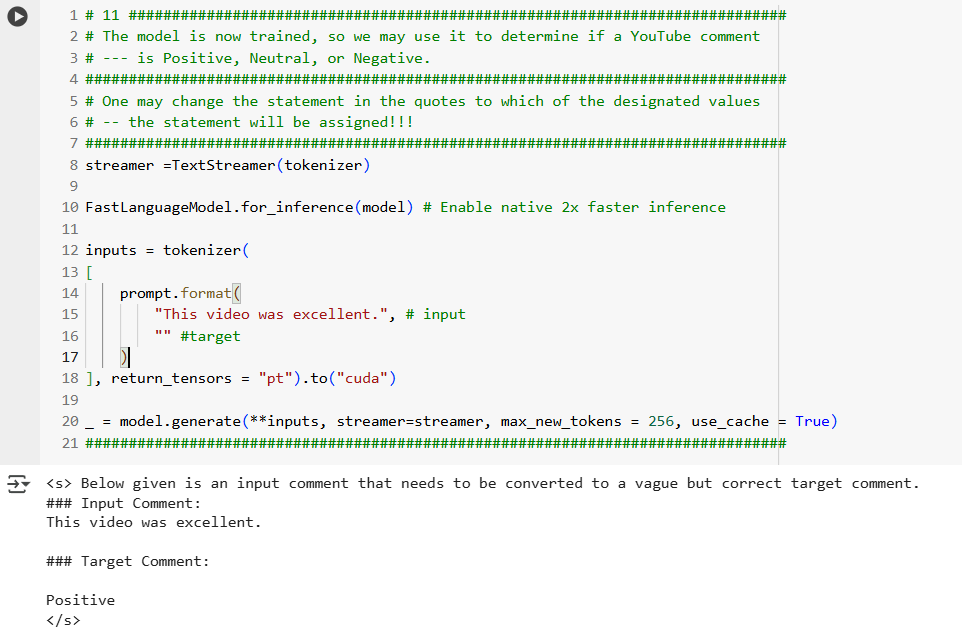






(Execution Details)





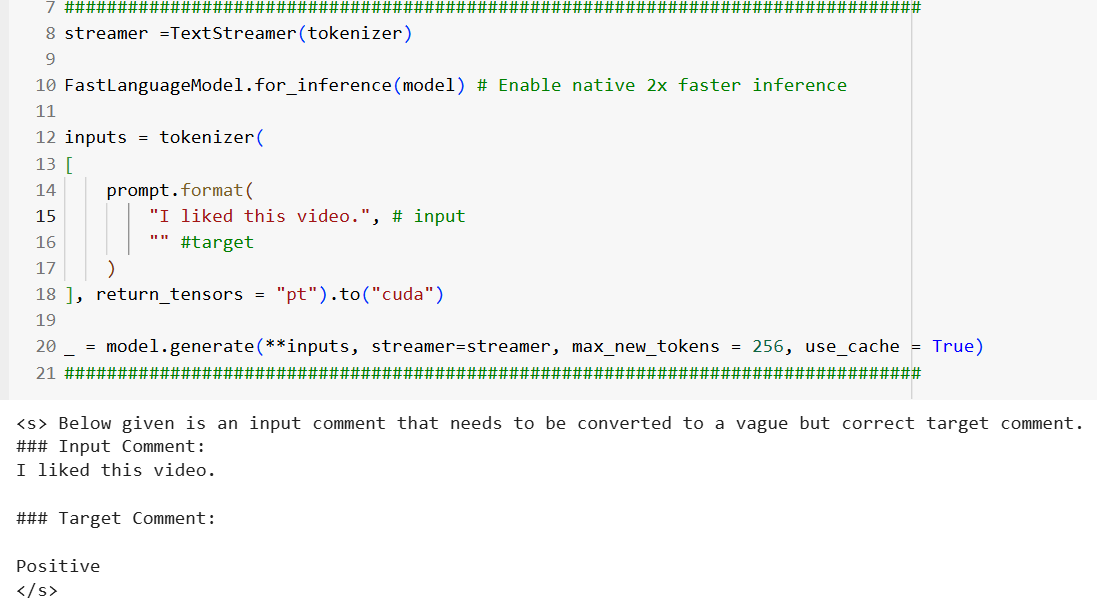
**Results And Discussion**

The following are some images from rerunning the last cell in Colab, which yielded the sentiments of the comments. ***We are the testers, noting if the answers make intuitive sense.***

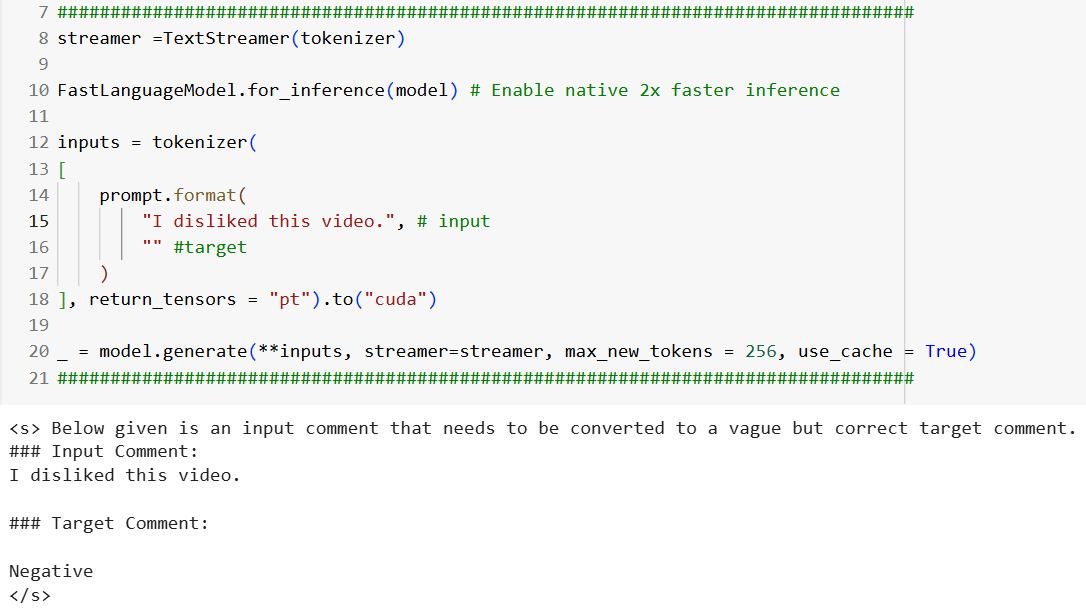
I used data that I was sure should be of positive or be negative sentiment, and indeed in these cases I found the sentiments to be as expected.

Sometimes I combined a positive and negative statement, to attempt to obtain a neutral sentiment. In some of these cases the sentiment was indeed neutral, but sometimes it was positive or negative instead, maybe this because the model saw it as more positive than negative or vice versa. Let us look now at the model’s responses.

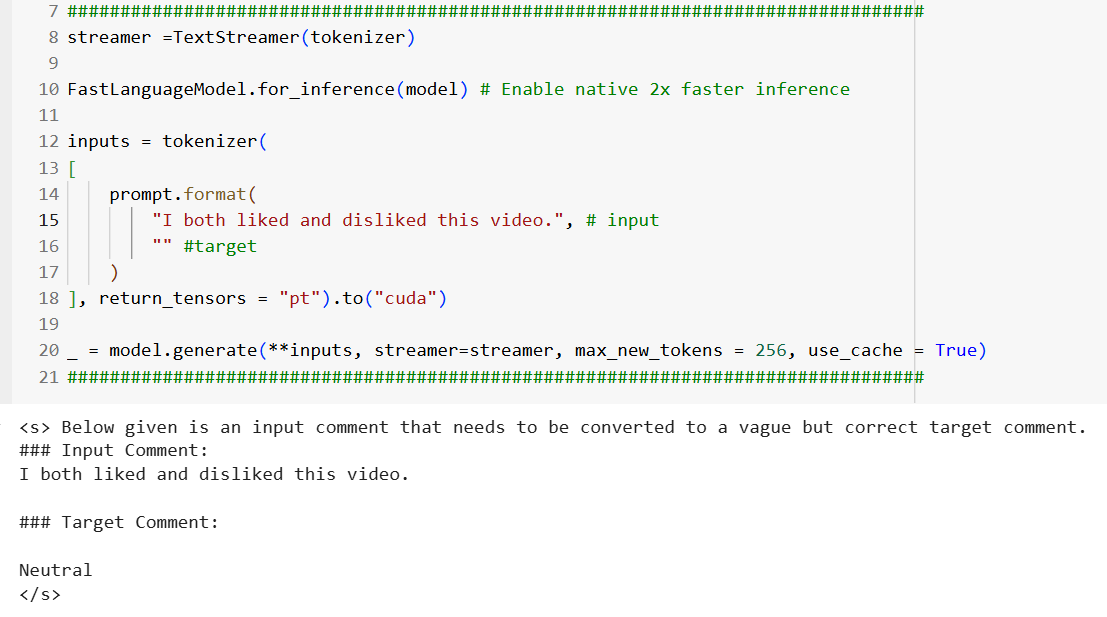
This was sure to return positive, if the model was working correctly.



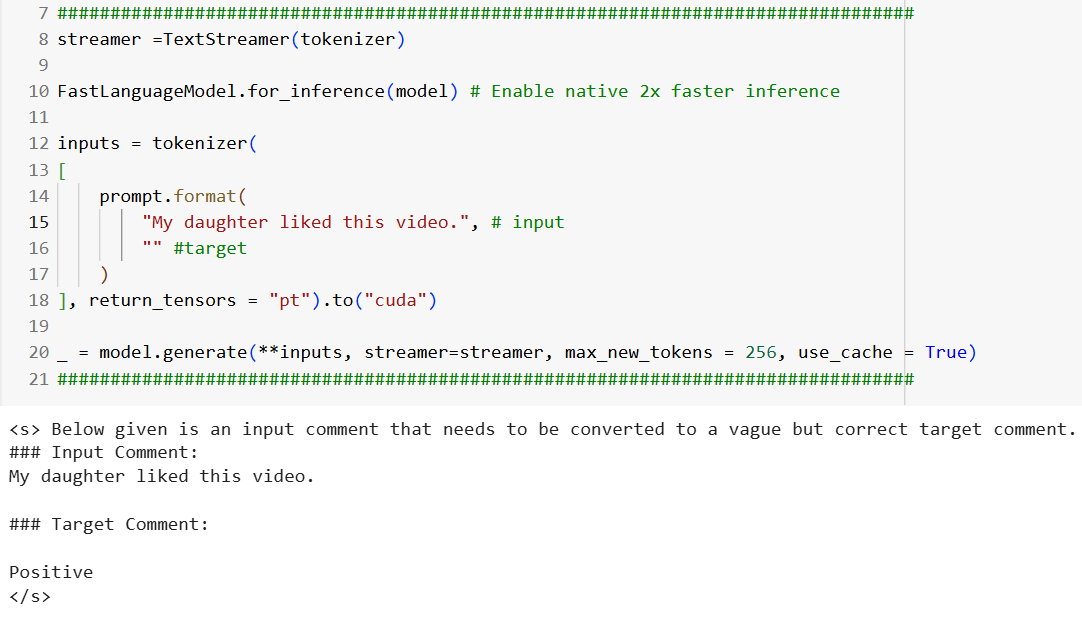
Similarly, if the model was working correctly, this should return negative, and it did.



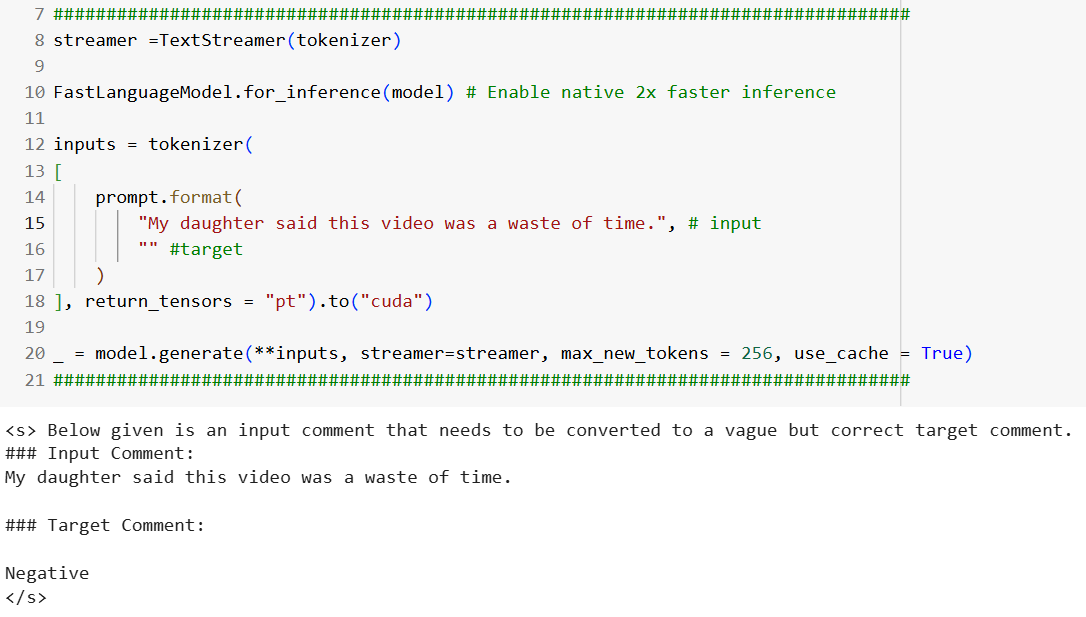
This example was an attempt to combine positive and negative statements to return a neutral sentiment, and it did.



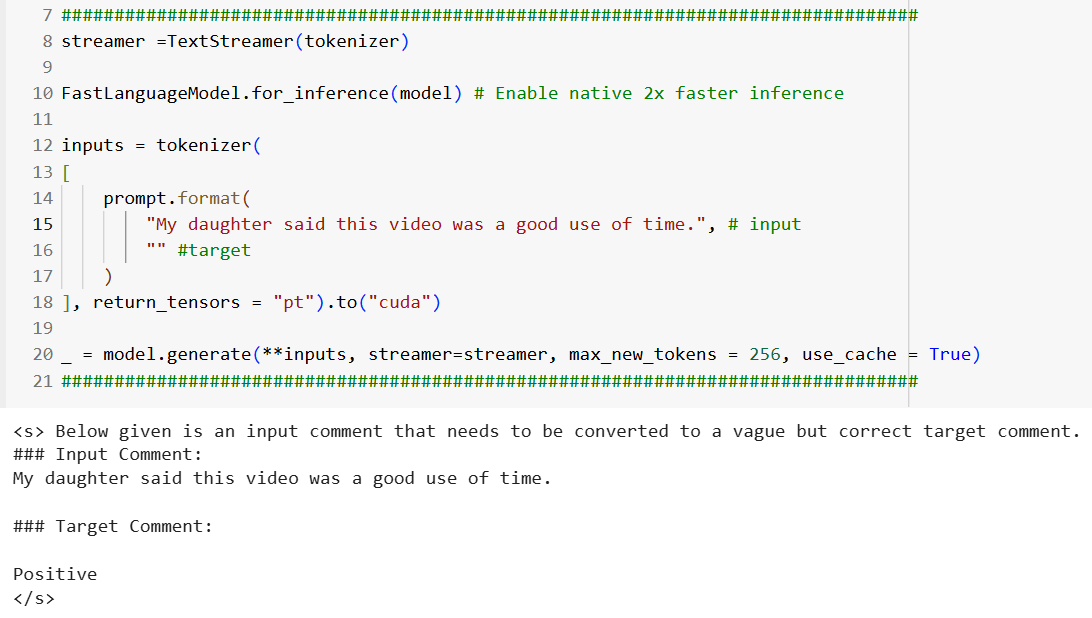
Another attempt to return positive; it did.



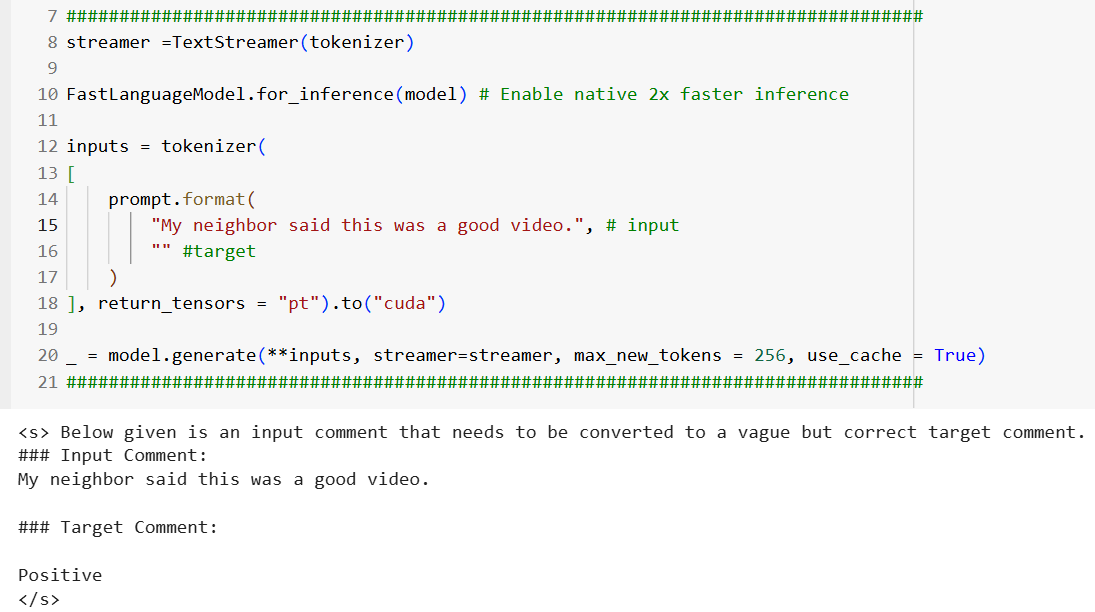
An attempt to return negative, and indeed it did.



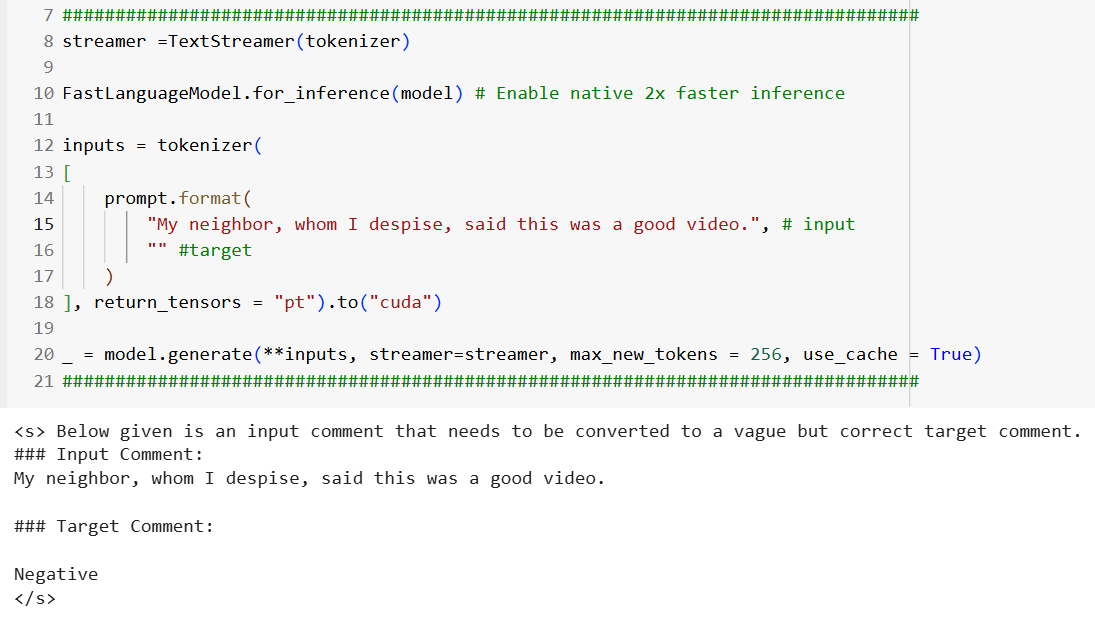
Again, a successful positive return.



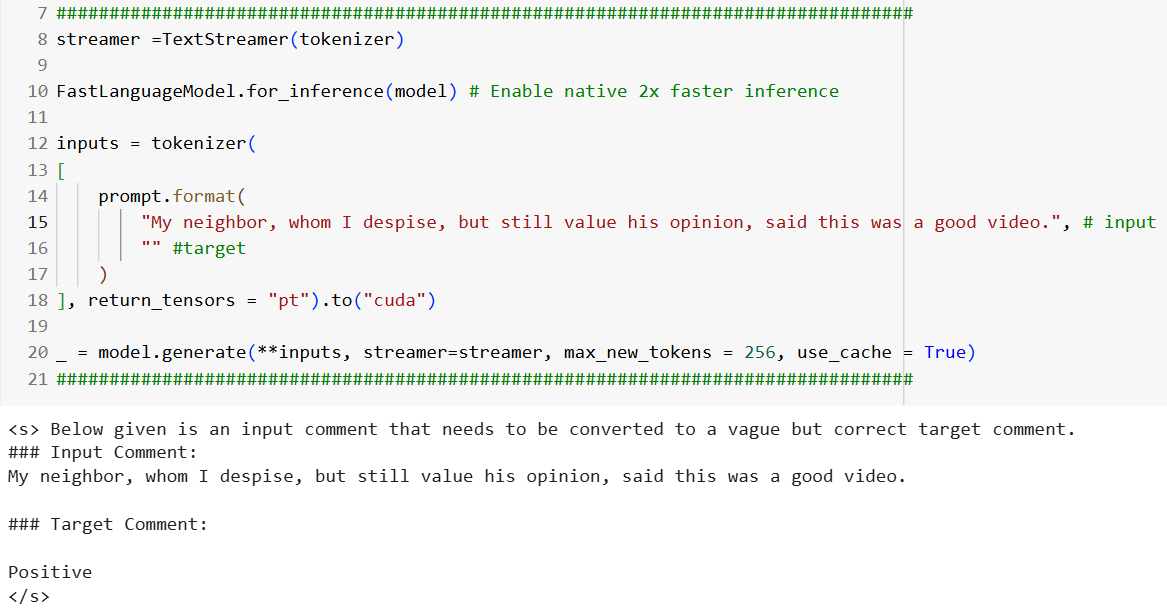
Unsurprisingly, this returned a positive sentiment.



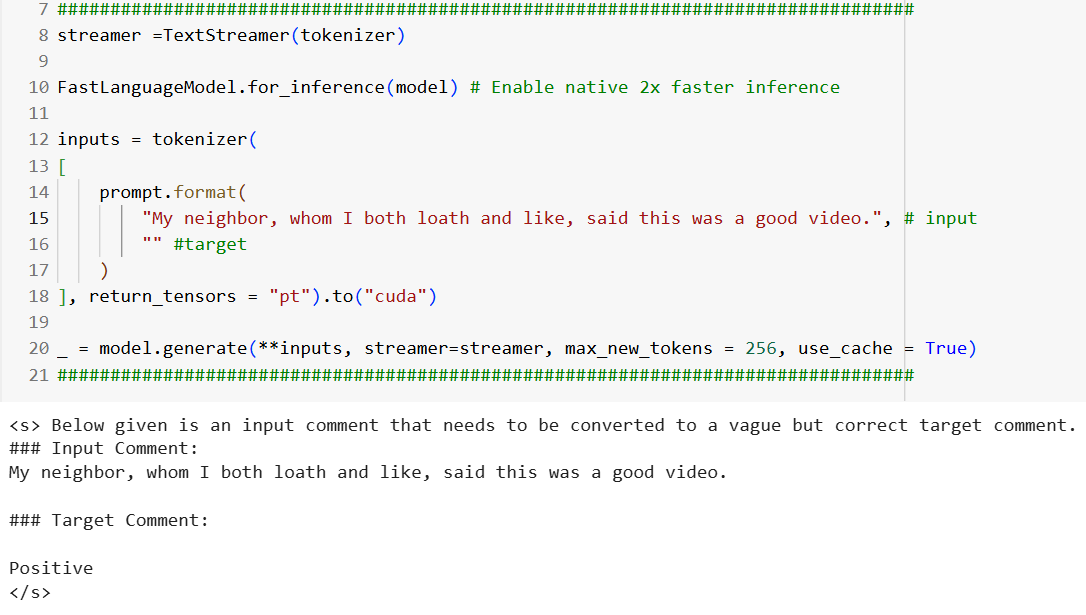
In this case I changed the previous example to include my bad feelings about my neighbor; note that this changed the sentiment to be negative instead of positive, which I sort of expected.



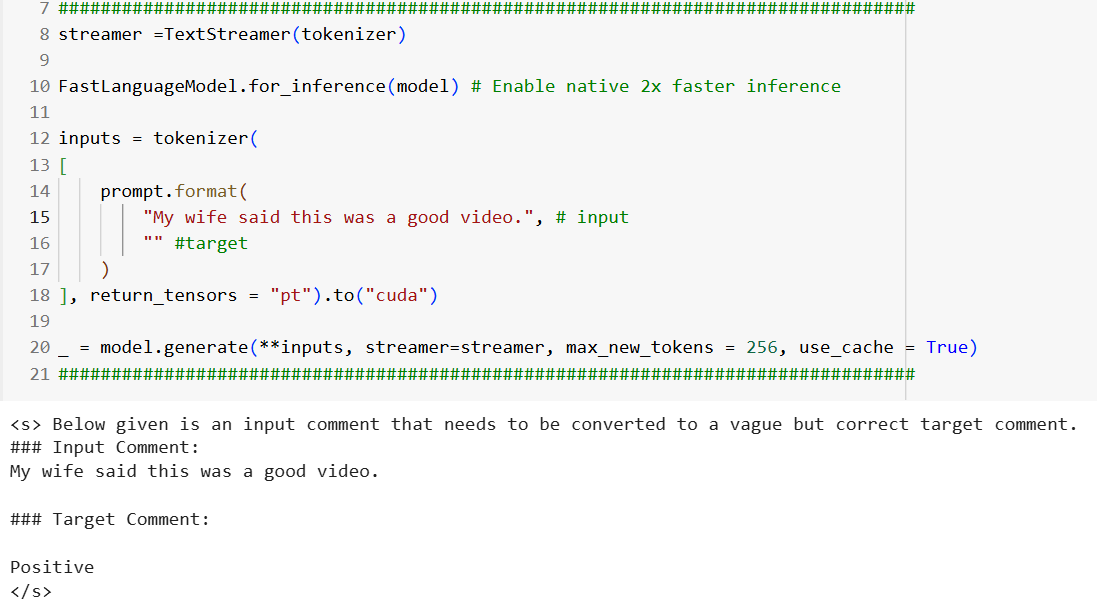
Here I added more positive information to the previous case, which changed the sentiment back to positive. ***I found this to be not unexpected.***



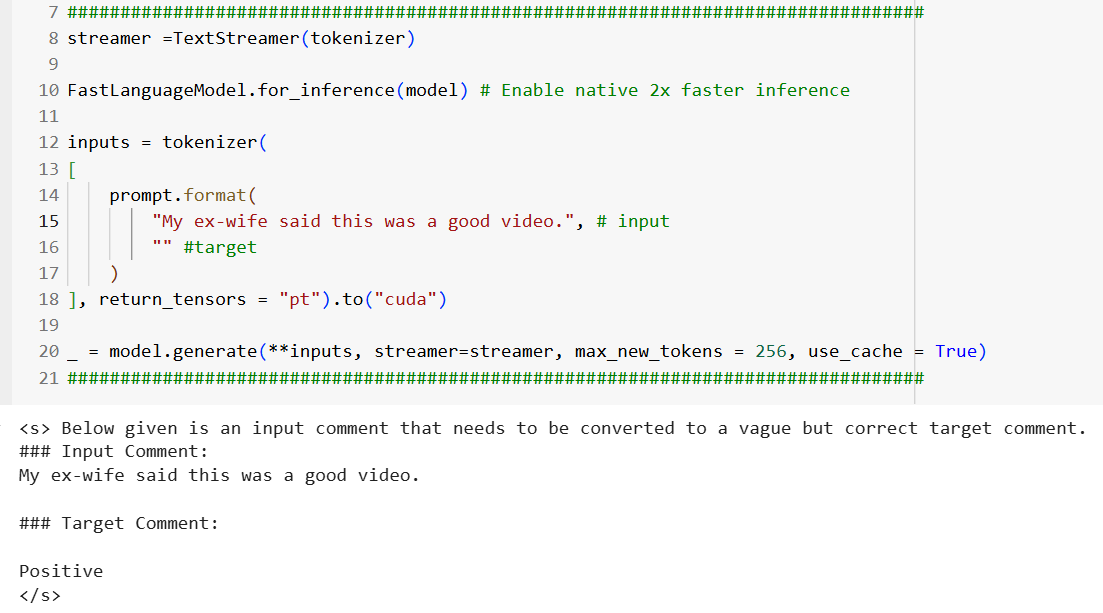
This is an ***attempt*** at a neutral, but it returned positive. Interesting, but not unexpected.



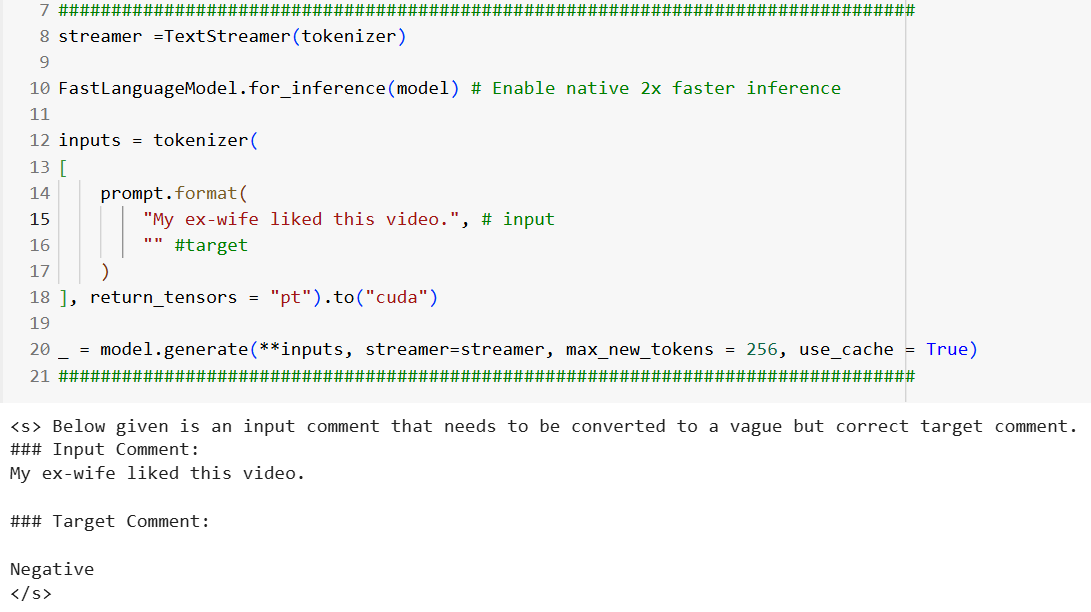
Positive, as expected.



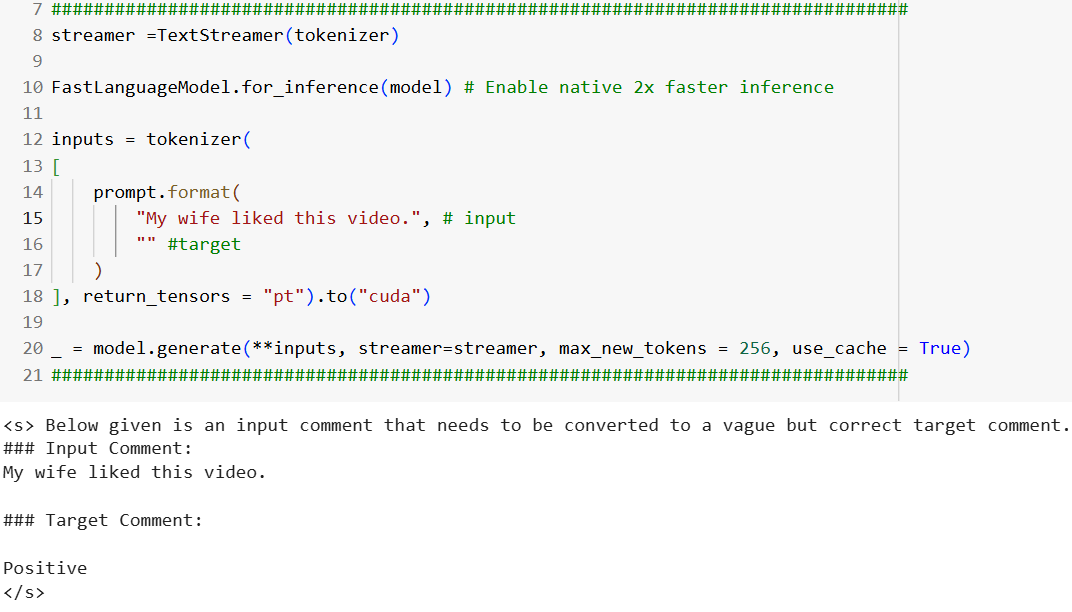
This was an attempt at a neutral or negative, but it returned positive; still not too far from expectations.



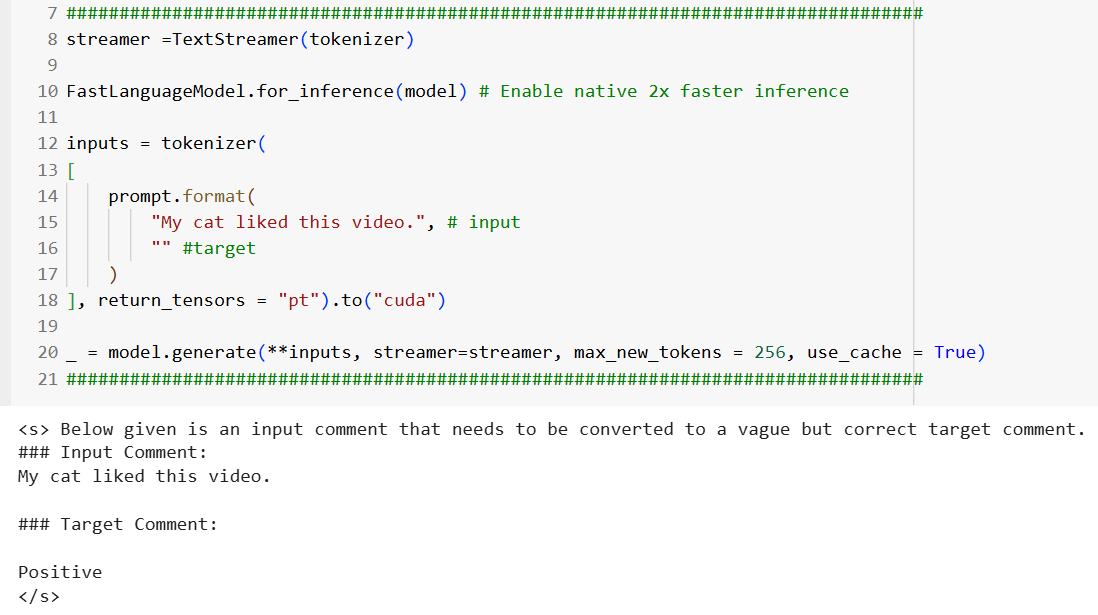
I modified the above example to get a negative. This illustrates the difference in the ex-wife saying the “video was good” and “liking the video”. Strange, but not unexpected.



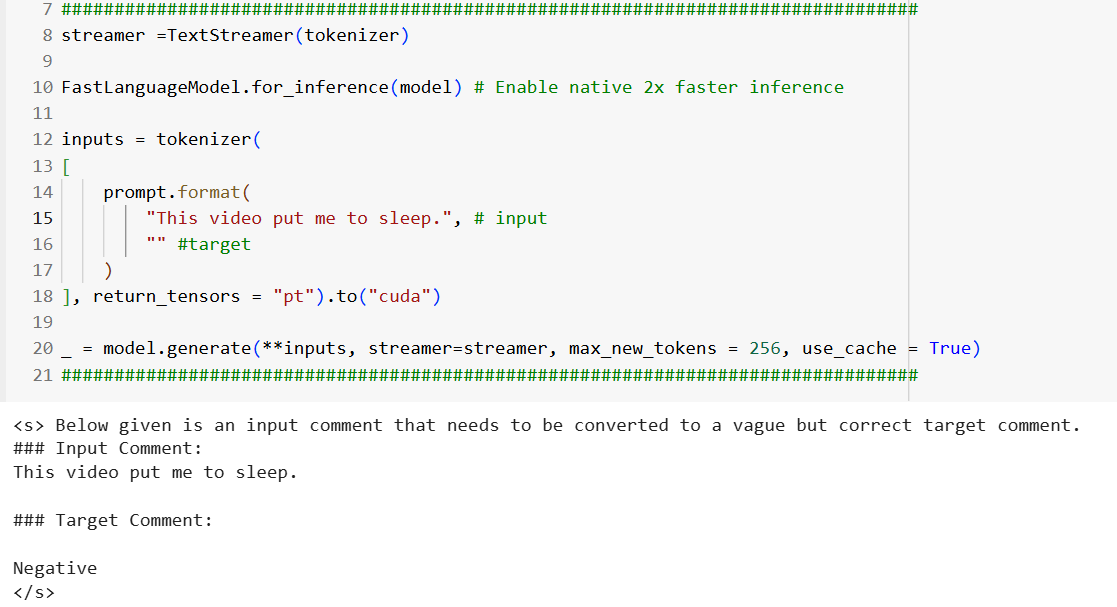
I changed the previous example from ‘ex-wife’ to ‘wife’, and the result was positive again. As to be expected.



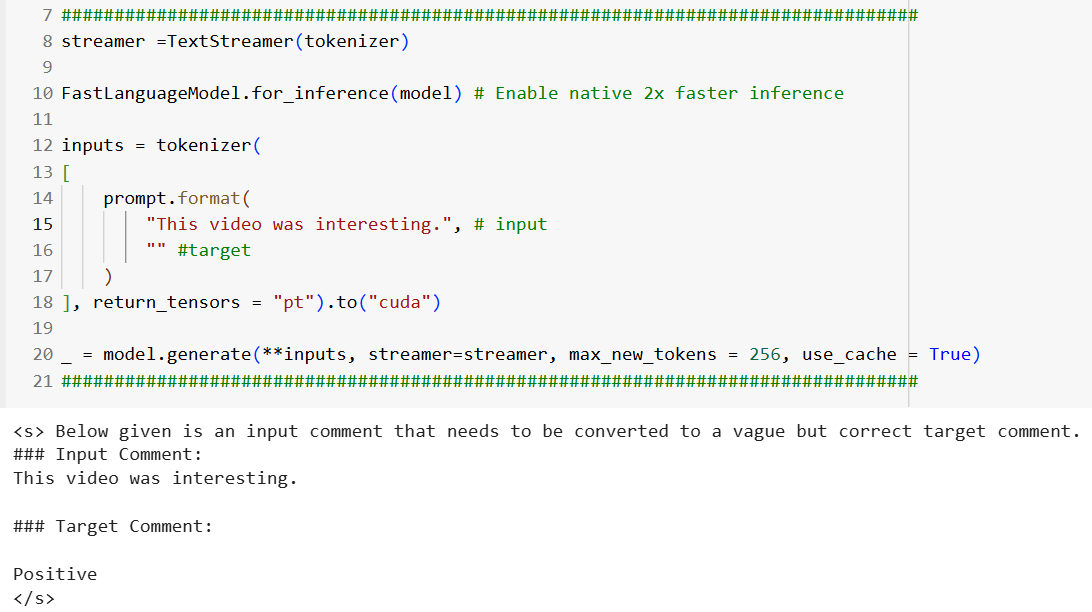
In this example, I wanted to see if it would matter if I used the cat as a YouTube commenter.



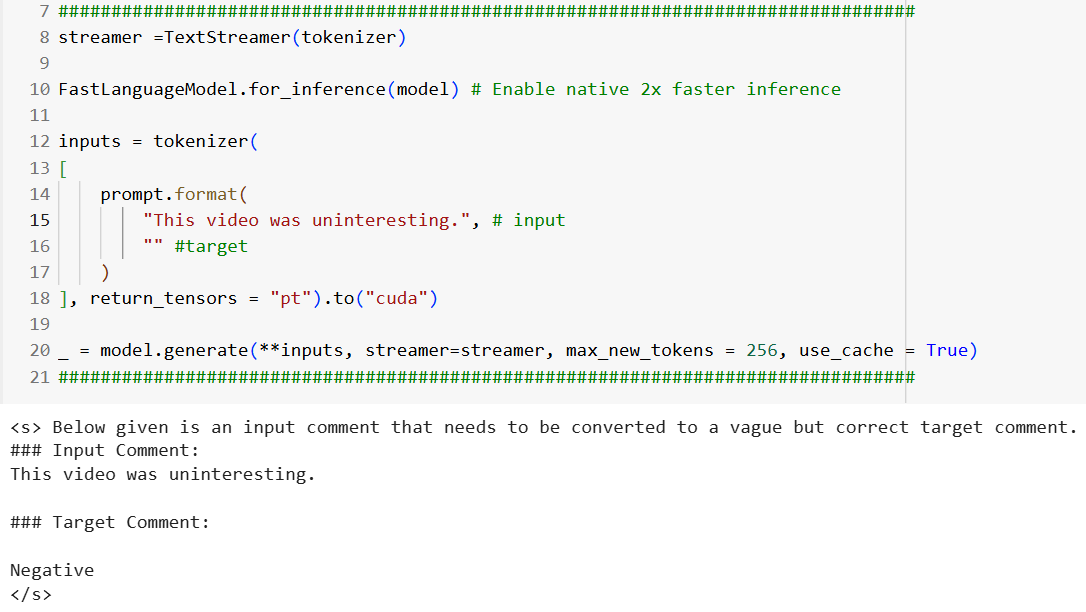
I expected this to return negative sentiment and it did.



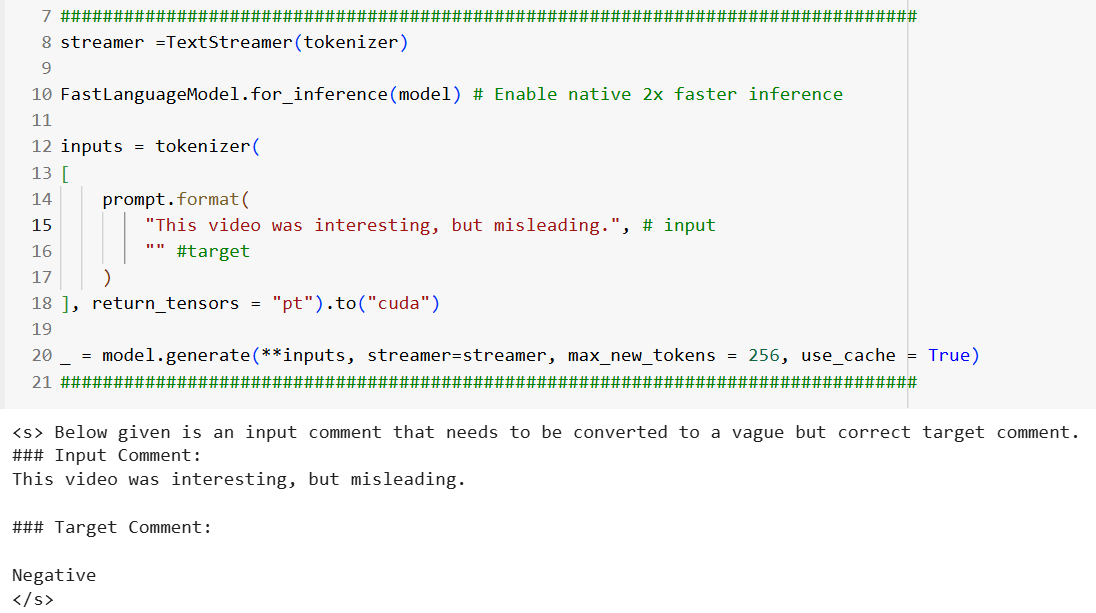
Positive as expected.



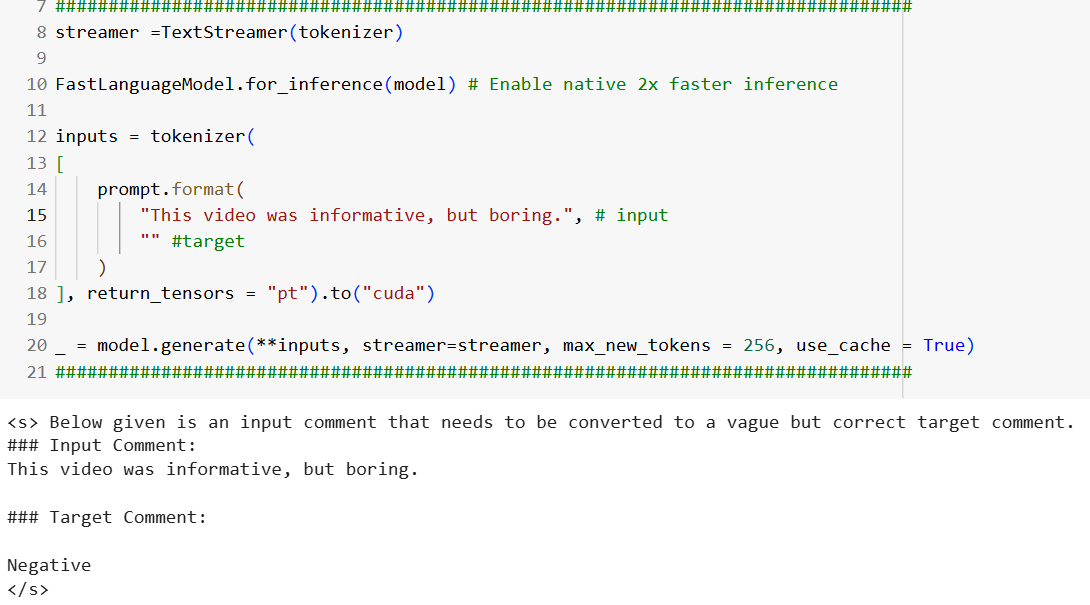
Negative as expected.



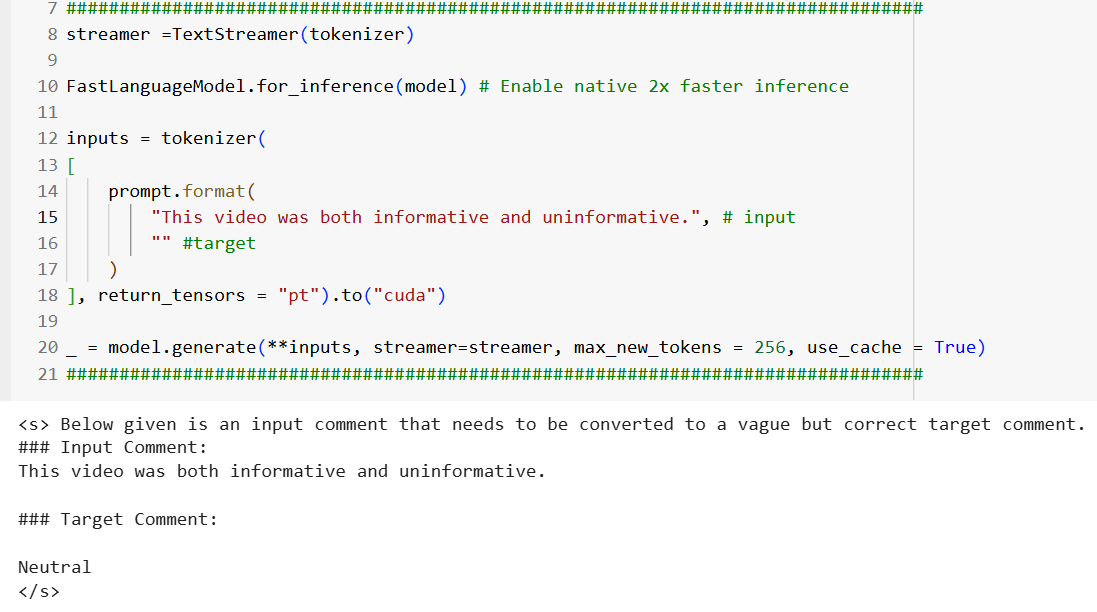
Here was an attempt at a neutral that was negative; still not out of range. Apparently, “interesting” does not outweigh “misleading”!



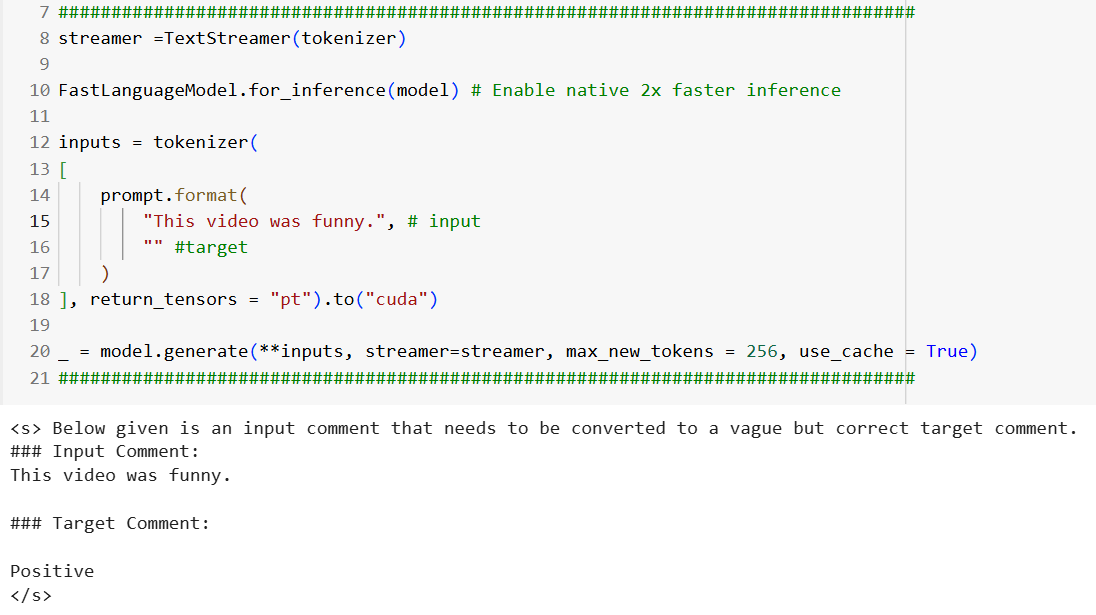
This is another attempt at a neutral that was negative.



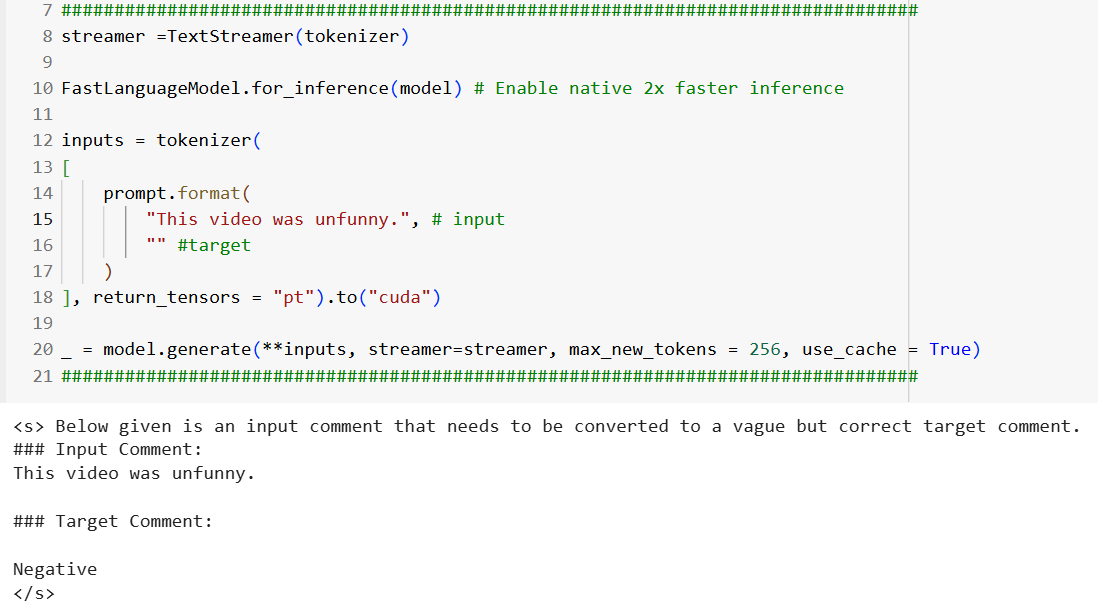
This attempt at neutral succeeded.



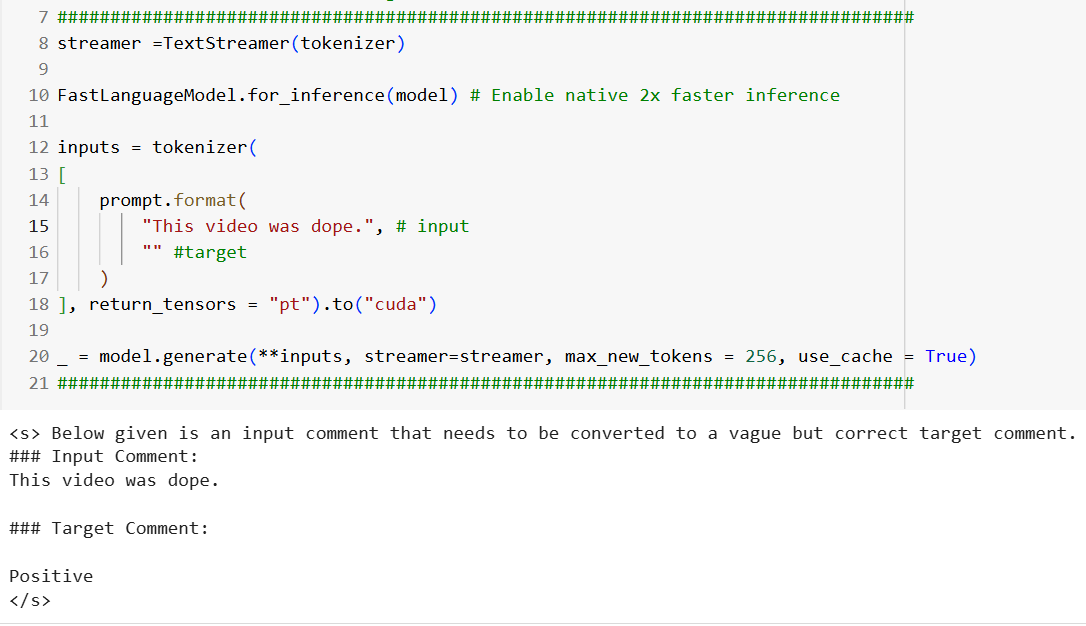
Positive.



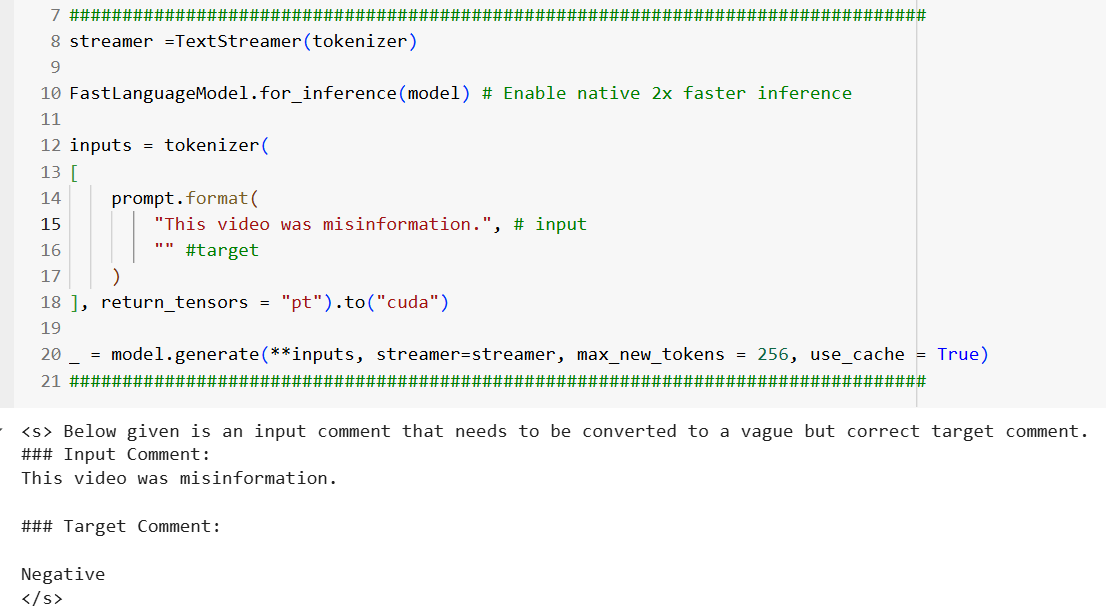
Negative.



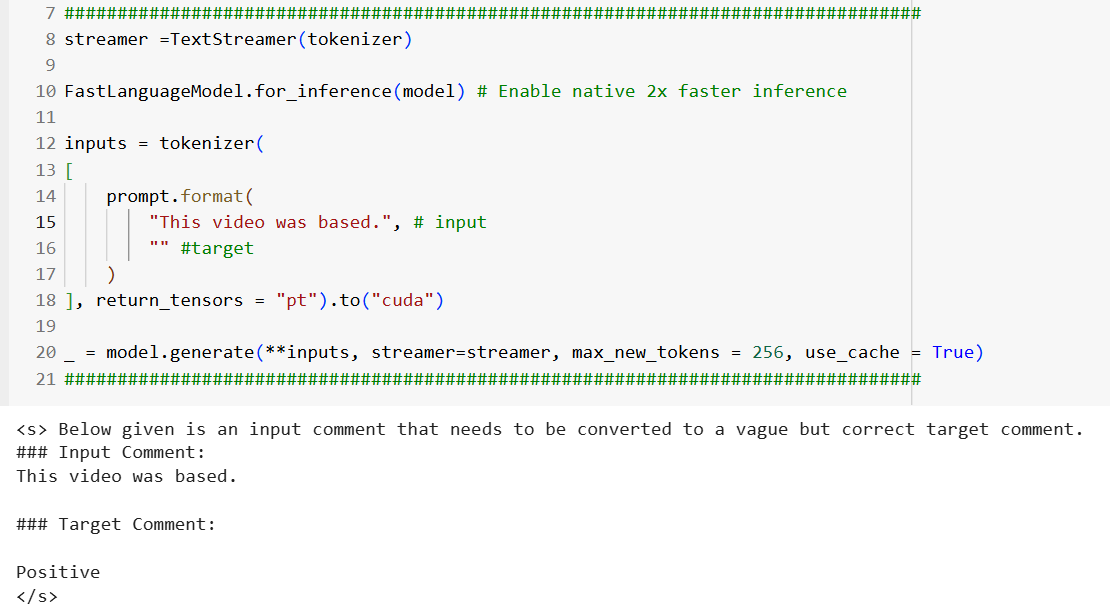
In this case I use some slang, and it returned positive as expected.



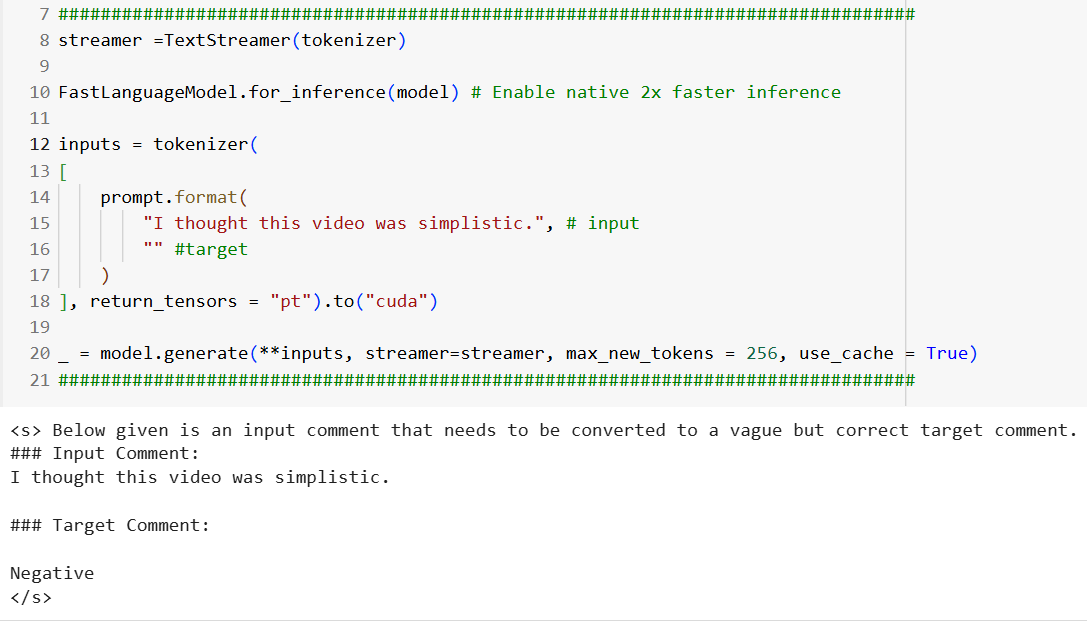
Negative as expected.



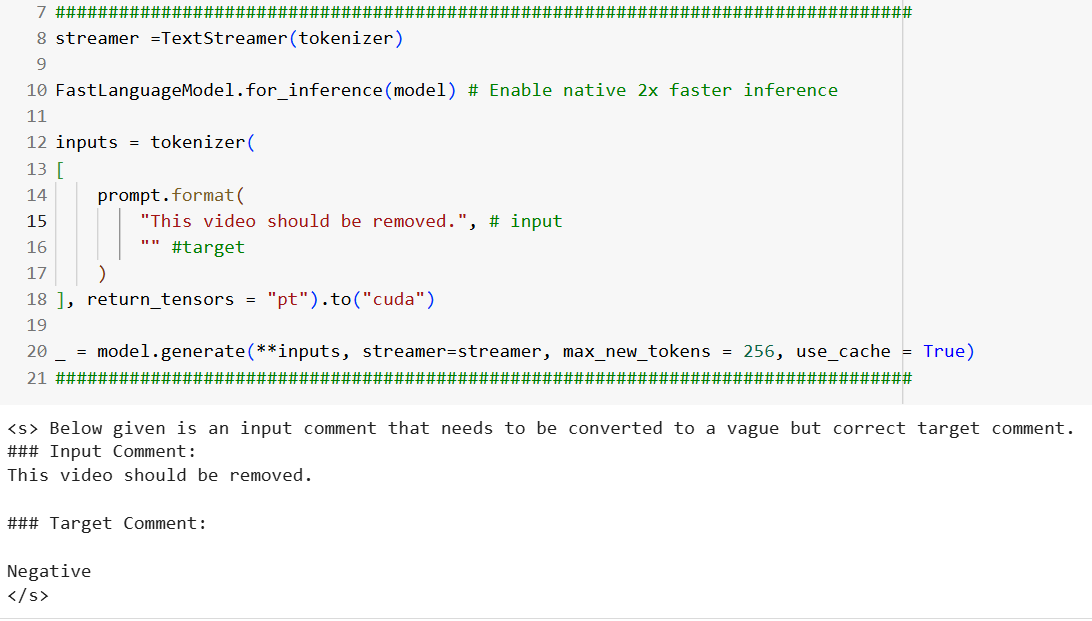
More positive slang.



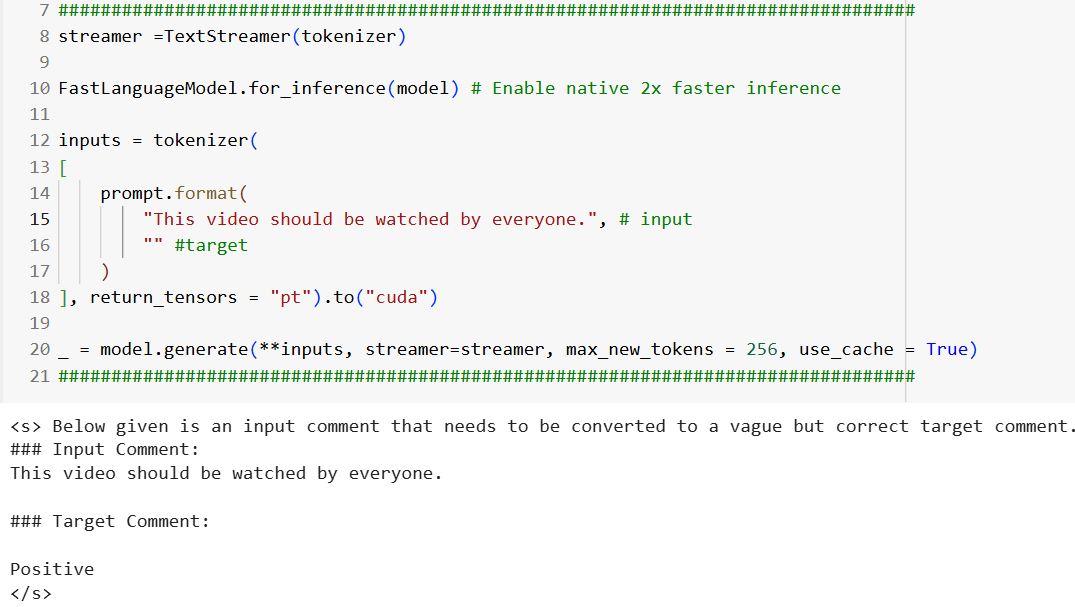
Negative.



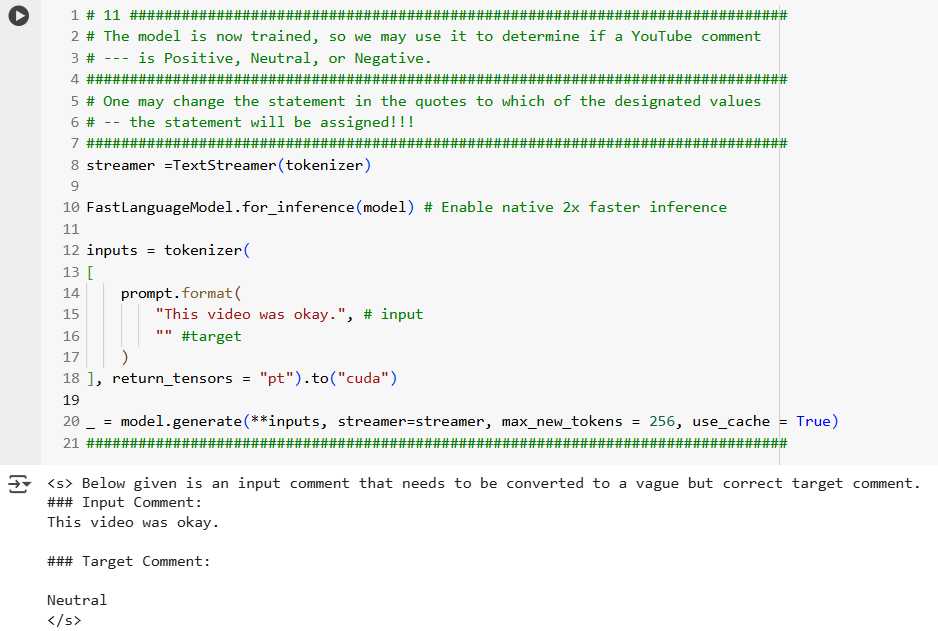
Definitely negative!!!



Definitely positive!!!



A final neutral comment, indeed!



**Conclusion**

In the testing examples that just followed, sometimes there were results that were unanticipated, but still reasonable. If unanticipated, the result would still be not too far out of what one would expect.

If one used input that should ***definitely*** return positive sentiment then it would indeed return positive; input that should ***definitely*** return negative sentiment would indeed return negative.

Attempts to return neutral responses, sometimes returned positive or negative instead. I wondered how the trained model weighed combinations of comments that contained both positive and negative elements, and thereafter returned maybe unanticipated, but still reasonable results.

**Future Work**

I want to create some new code adjustments, including the following:

1. I would like to be to load another CSV file, which would have the Comment Text column filled in and the Sentiment column empty. The Colab application would then complete the data for the Sentiment column, and return this file as output back to the Colab directory.
2. I would also like to train on other similar files and on other parts of the given data file. The given file was so large that I only trained it on the only first 10,000 records.

**Potential Uses of This Project**

If I do the previous **Future Work**, this project would be of great advantage to a data analyst, or other person seeking to categorize text data.

An analyst would capture the Comment Text in a CSV file (able to be accessed by Excel), as mentioned above, after users placed their comments in a website form. A data analyst would then use the trained model to categorize the data into Positive, Negative, or Neutral Sentiment categories, and thus study the data using Excel or a data analysis tool, such as SAS or Tableau**.**

**The following is a video that illustrates the use of this Colab application:**

<https://youtu.be/BMTCNt-wF0M>

**References**

1. **Data Source:** Poonawala, Amaan; YouTube Comments Sentiment Dataset; 3 months ago;

<https://www.kaggle.com/datasets/amaanpoonawala/youtube-comments-sentiment-dataset?select=youtube_comments_cleaned.csv>

1. **Code Inspiration:** Deep Blue Research; Finetuning mistral 7b using Unsloth;   
   Jun 4, 2024; <https://dbrpl.medium.com/finetuning-mistral-7b-using-unsloth-2cf554159a03>