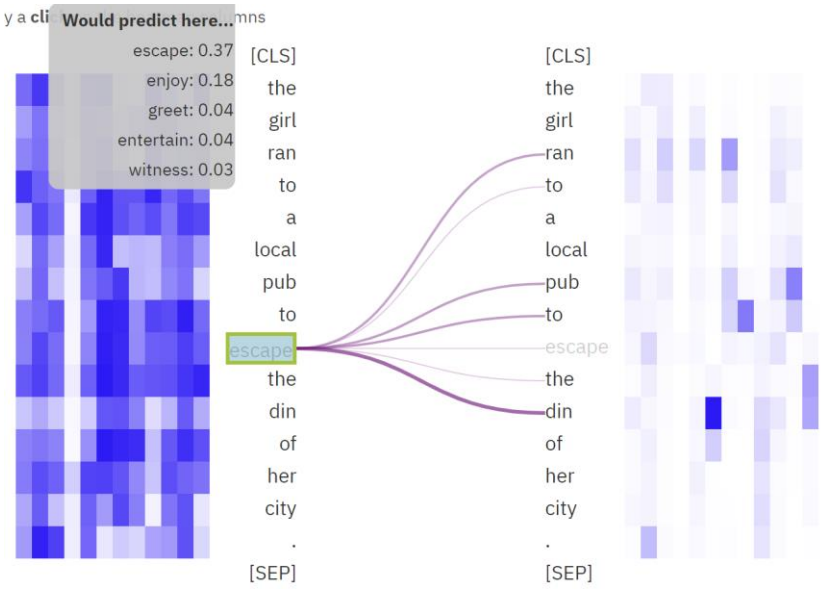
Intro_to_AI HW4

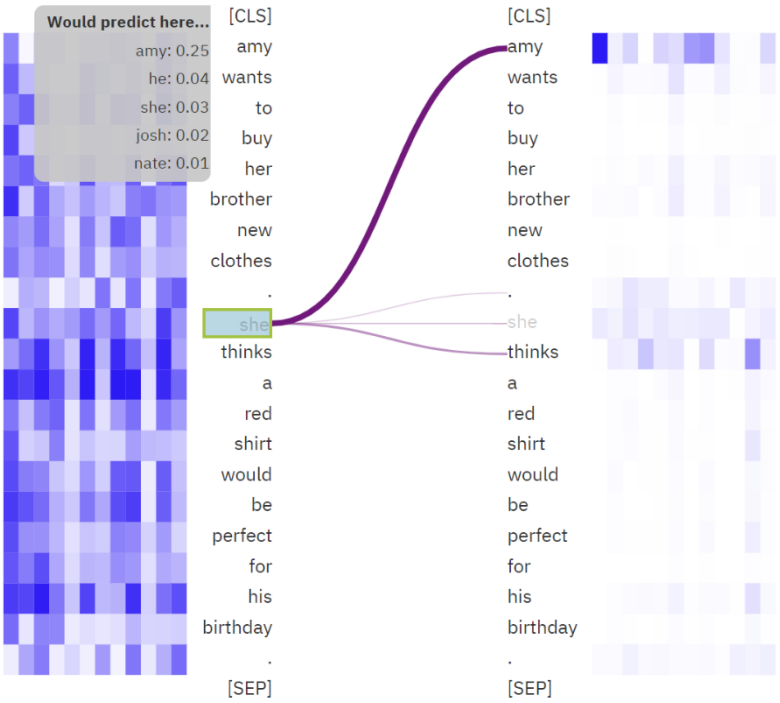
Part 1: Attention Visualization – exBERT

1. ExBERT (Explainable BERT) is a tool designed to visualize and interpret the inner workings of BERT models. The core of BERT is the attention mechanism, which allows the model to focus on specific sections of the input.
   The pretrained process of BERT includes MLM (Masked Language Modeling) and NSP (Next Sentence Prediction). We can use exBERT to understand the attention patterns and relationships across different tokens, layers, and heads that BERT learns during these tasks.
2. First, we choose “distilbert-base-uncased” as the model and observe how MLM works.
   1) The input sentence is: “The girl ran to a local pub to escape the din of her city.” And the masked word is “escape”.
      - There’re only 6 layers.
      - In this multi-head attention mechanism, there’re 12 heads. Different attention heads are for capturing various aspects of the input data, which allow the model to extract richer features.
      - I choose layer 3 because other layers seem to pay more attention on the previous or next word but not the content. Words that “escape” pay attention to are “ran” and “din”, which meaning are truly relevance with “escape”. It shows the attention mechanism.

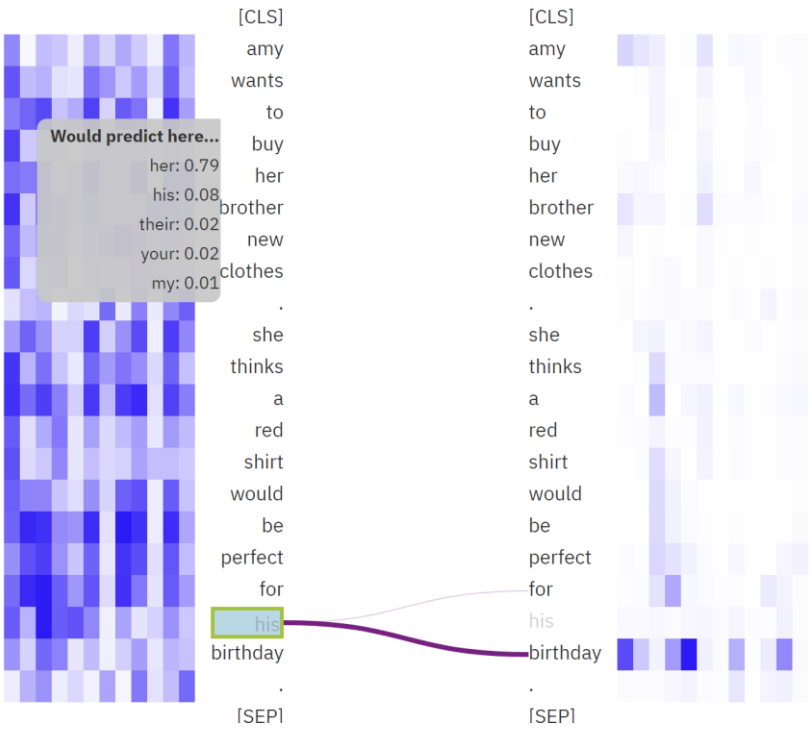


2) The input sentence is: “Amy wants to buy her brother new clothes. She thinks a red shirt would be perfect for his birthday.” And the masked word is “She”.

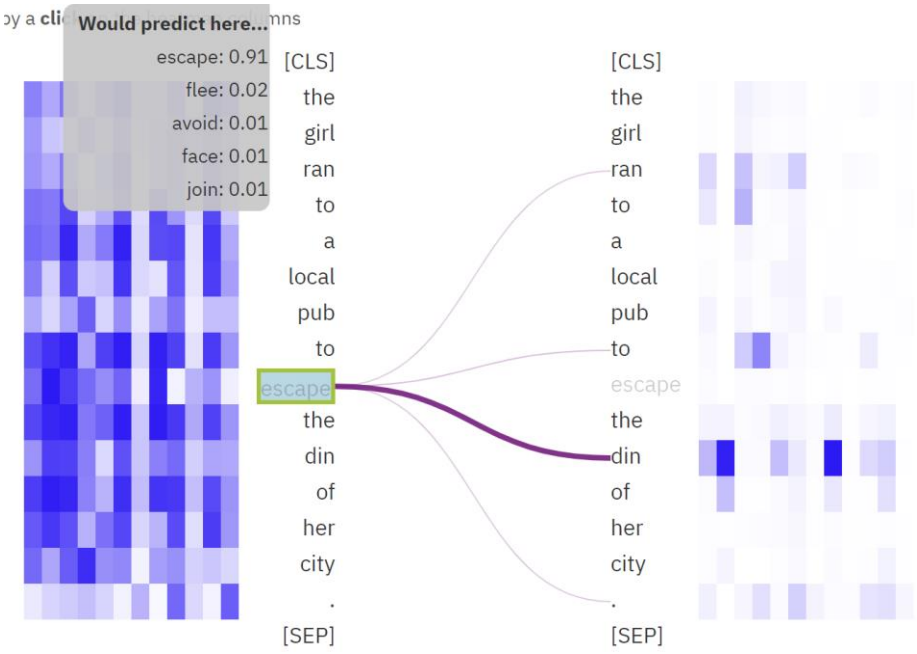
- I choose layer 5, which pay the most attention to the word “Amy”.



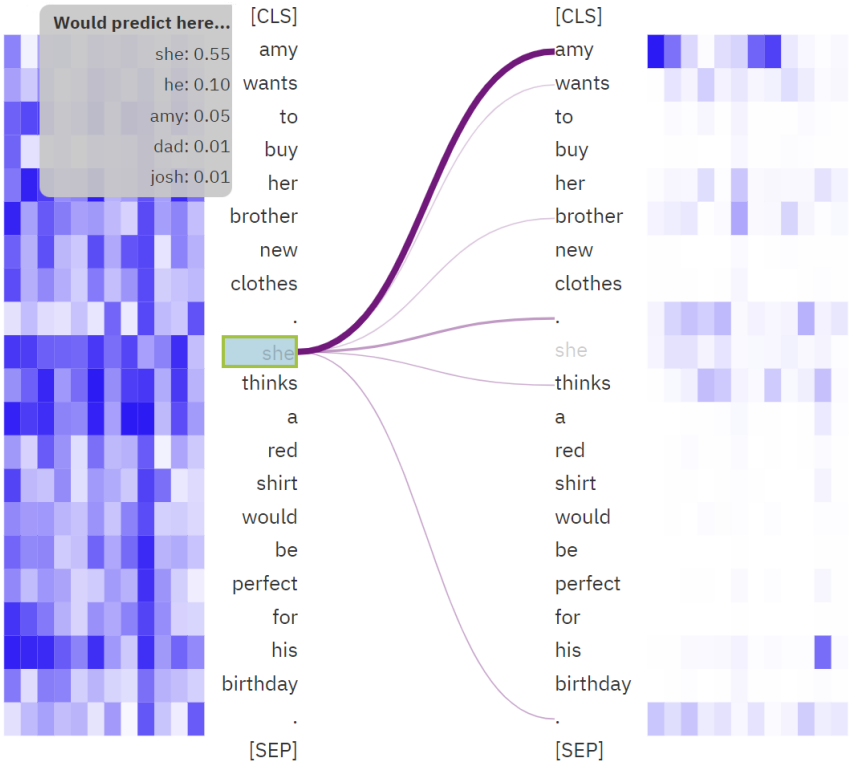
- However, if I choose to mask the word “his”, the attention pattern doesn't seem ideal. And the word it predicts is “her” which is incorrect.



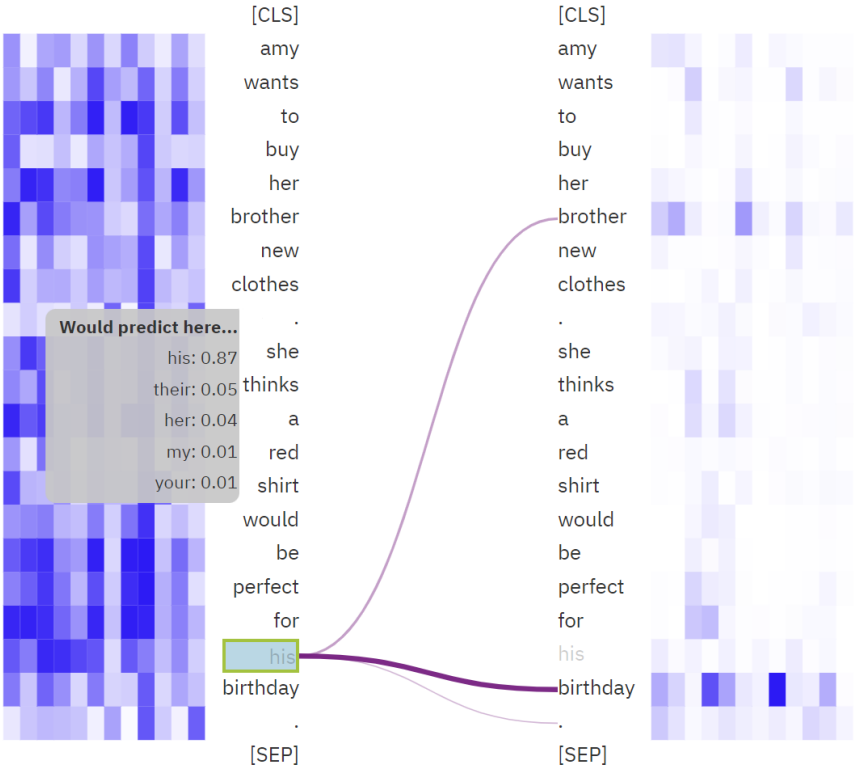
3. Due to the issue of not paying enough attention, I use the “bert-base-uncased” model and compare it with the “distilbert-base-uncased”.
    1) The input sentence is: “The girl ran to a local pub to escape the din of her city.” And the masked word is “escape”.
        - There’re 12 layers. And I choose layer 10. It pays most attention to the word “din”.
        - If choosing layer 11 or 12, it’ll only pay attention to the [SEP]. If choosing lower layer, it’ll pay more attention to the nearby words.
        - Compared to the DistilBERT version, the BERT model can focus more on conceptually related words rather than just nearby words.
        - Also, the predicted probability of "escape" is 0.91, which is significantly higher than that of DistilBERT. And words like "flee" and "avoid" are also predicted as similar words.



    2) The input sentence is: “Amy wants to buy her brother new clothes. She thinks a red shirt would be perfect for his birthday.” And the masked word is “She”.
        - Compared with DistilBERT, the predicted probability of word “she” is higher than word “Amy”.



- If the masked word is “his”, the attention words include “brother”, and the predict probability of “his” is 0.87, which is quite high.



3) Overall, the performance of BERT model is better than DistilBERT, but DistilBERT still show a certain level of effectiveness.

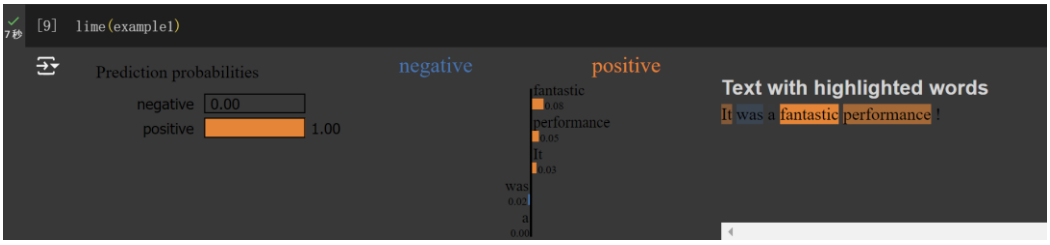
Part 2&3: Explanation Techniques – LIME&SHAP

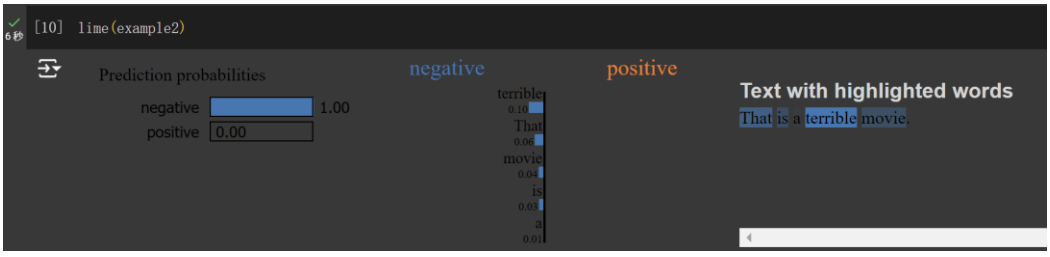
1. LIME (Local Interpretable Model-agnostic Explanations) and SHAP (SHapley Additive Explanations) are used to explain the predictions of machine learning models. They provide the contribution proportion of each word to the predicted result.
2. Test with example 1 and example 2:

```
[ ]  example1 = 'It was a fantastic performance !'
     example2 = 'That is a terrible movie.'
```

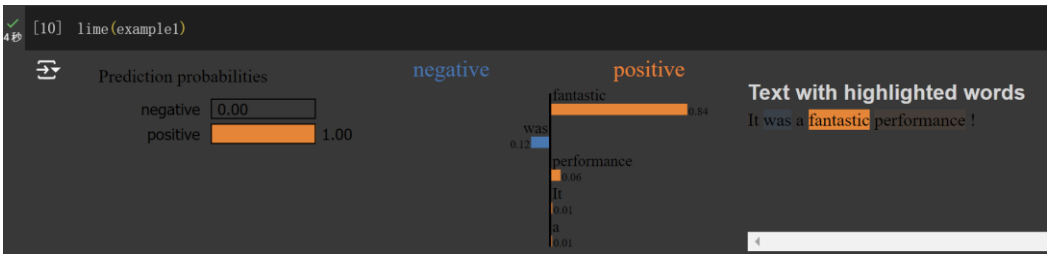
1) LIME

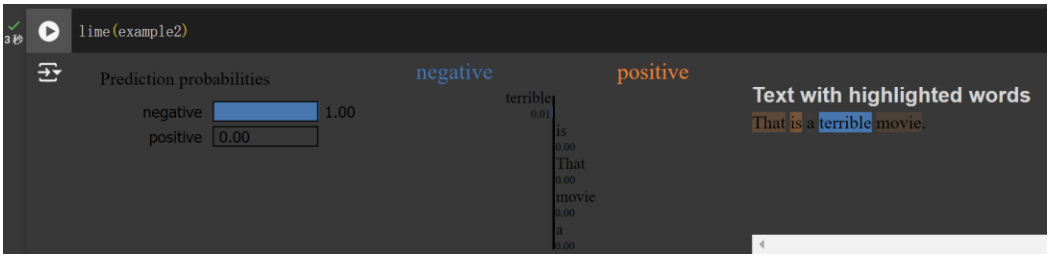
- Using model 1(distilbert-base-uncased)





- Using model 2(bert-small)

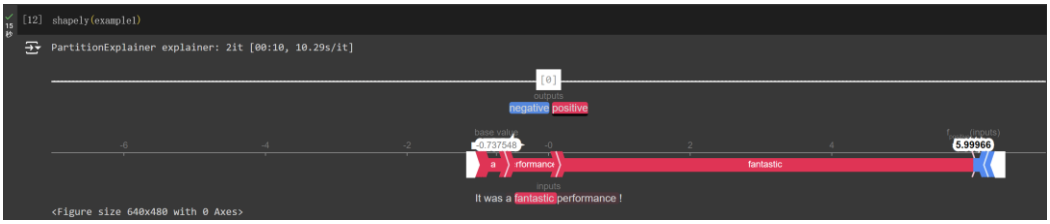




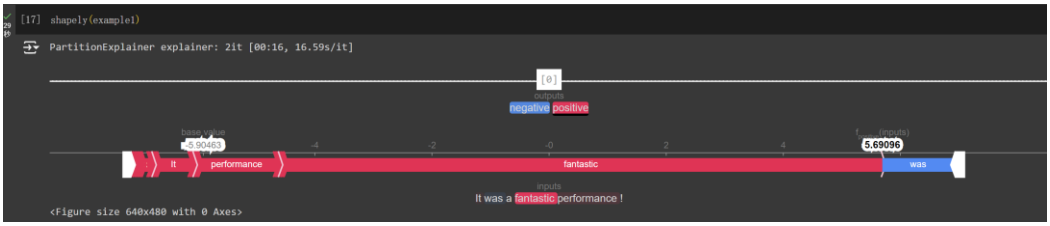
- With simple input, the result seems correct, and LIME correctly highlighted the words that contributes to positive or negative result like “fantastic” or “terrible”. Model_2 seems to be better since the probability of “fantastic” rise from 0.08 to 0.84.

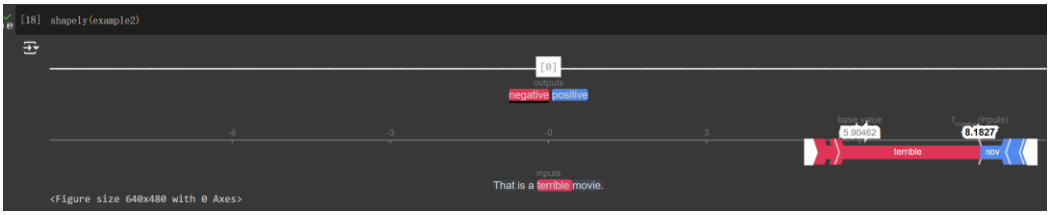
2) SHAP

- Using model 1



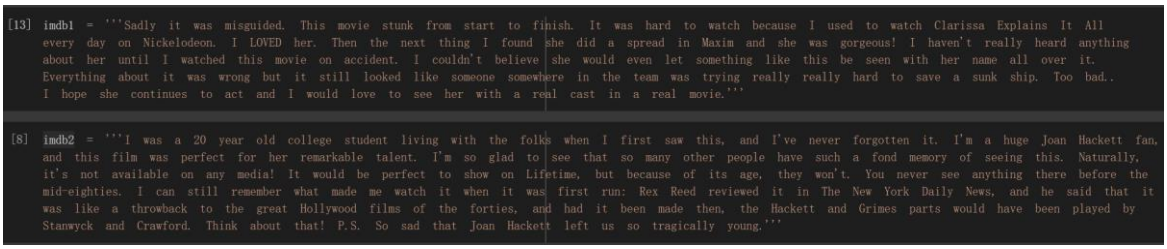
- Using model 2





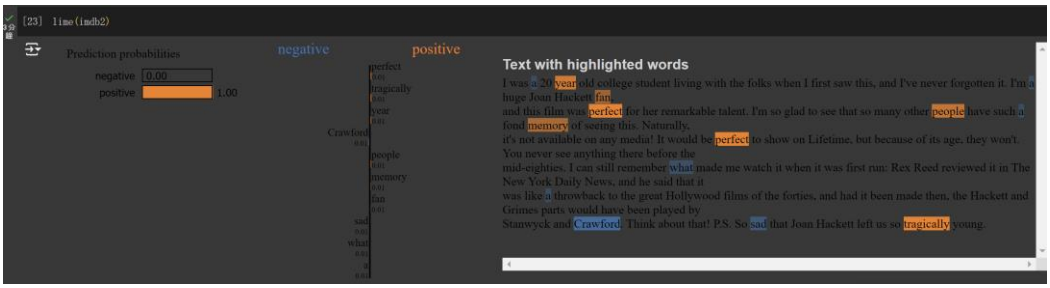
- With the positive result cases, the base value of model 2 is lower; with the negative result cases, the base value of model 2 is higher.

3. Test with IMDB movie review datasets input (imdb1&imdb2):
“imdb1” should be negative and “imdb2” should be positive.



1) LIME

- Using model 1

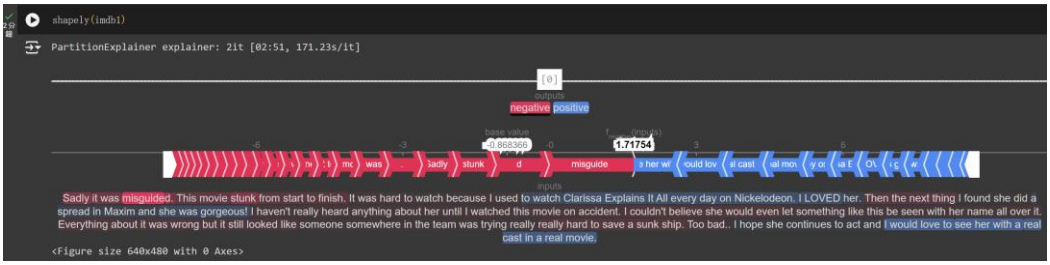


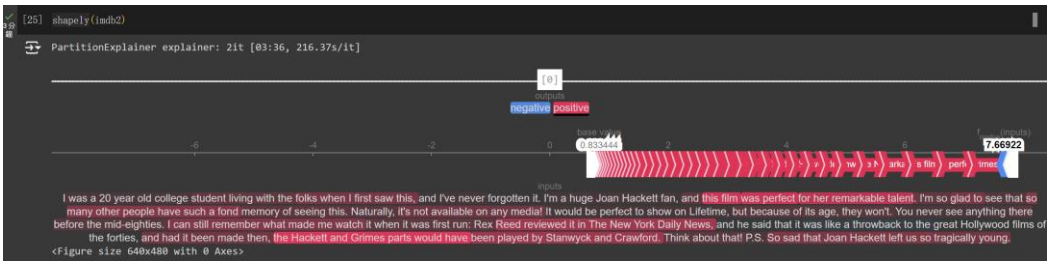
- Using model 2

- Both models give the correct prediction, but in the case of imdb1, model2 performs better with a 1.00 probability of being negative compared to model1's 0.85 probability of being negative. And the word it mark as positive like “loved”, “gorgeous” are more reasonable than in model1’s marked word: “continues” or “was”.

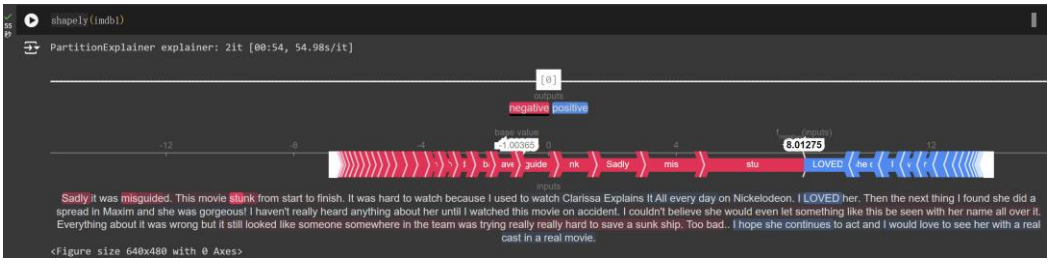
2) SHAP

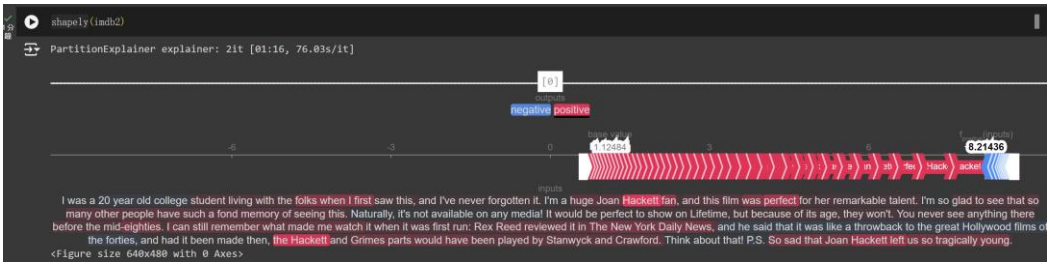
- Using model 1





- Using model 2





- I think in these two input example, model1 highlighted the words more correctly, and the base values doesn't differ significantly.

4. Compare the explanation of LIME and SHAP.
    1) SHAP takes longer time to run since its computation is more complicated than LIME.
    2) LIME explains the prediction locally while SHAP provides local explanations but can also be aggregated to give global insights.
    3) LIME generate a set of perturbed samples around the instance to be explained. And it generates randomly, so if I run the command again, the

result can be different. On the other hand, SHAP is based on game theory. It calculates the contribution of each feature to the prediction by considering all possible combinations of features.

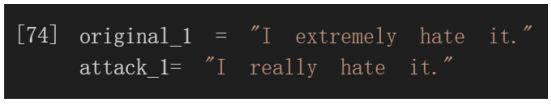
4) SHAP not only represents the contribution of each feature but also calculate a base value which serves as the baseline prediction for the model when no features are considered.
5) LIME's representations are more straightforward and easy to understand compared to SHAP.

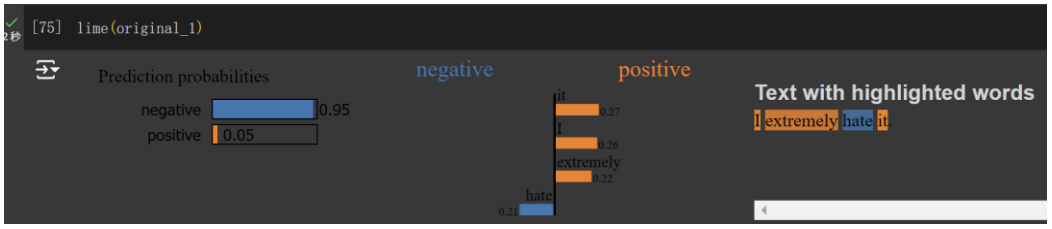
Part 4: Try some input sentences for attack

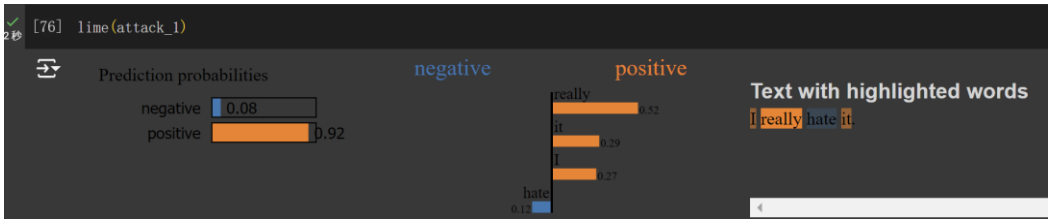
Attacks make small changes that cause a model to give wrong results. These changes are usually not noticeable to humans but can greatly affect the model's outcomes.

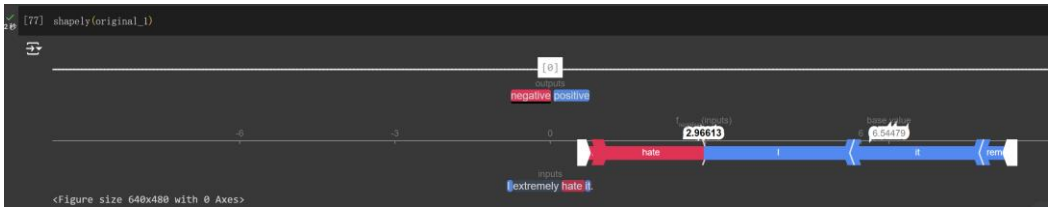
The model used in this part is “TA_model_2.pt”.

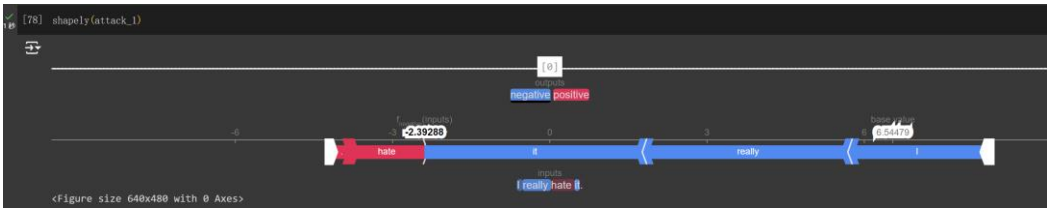
1. Replace the word with synonym.







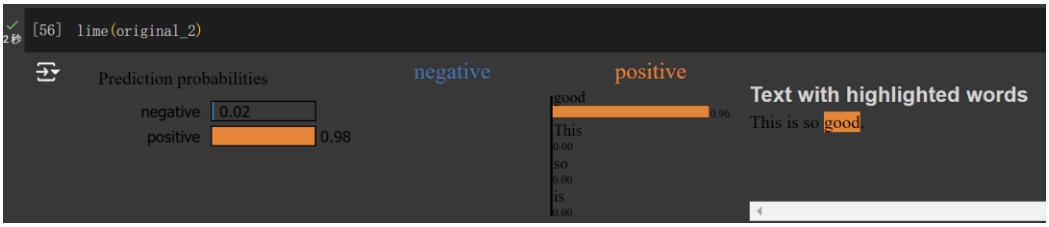


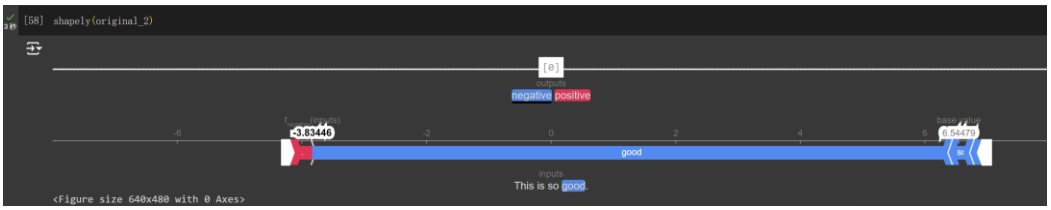


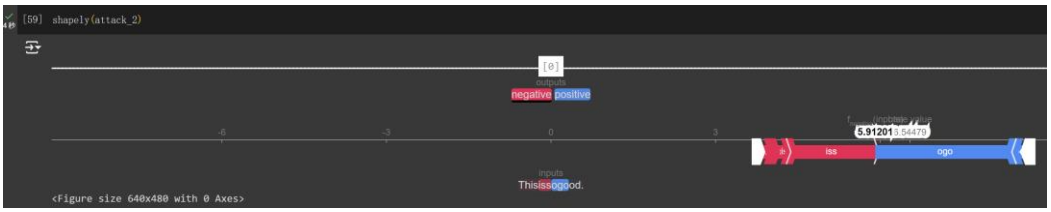
- The result shows that if replace “extremely” with “really”, “really” will be considered as a positive word that contributes significantly and affect the prediction.

2. Remove the space between words.

```
[55] original_2= "This is so good."
     attack_2  = "Thisissogood."
```



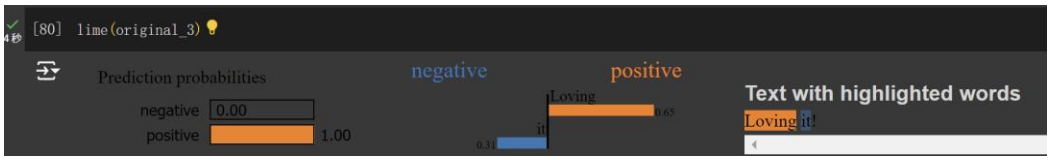


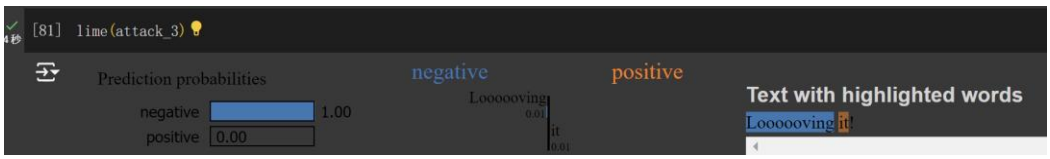


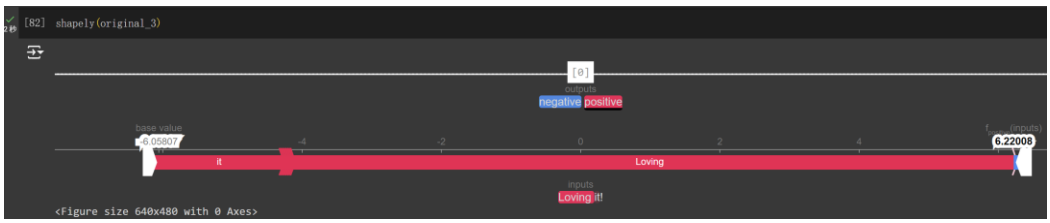
- When removing the space, LIME interprets the entire string as a single word, does not highlight any specific word, and predicts it as negative.
- In SHAP, even if it is a single string, it still extracts several letters as words, with the negative and positive components being about equal.

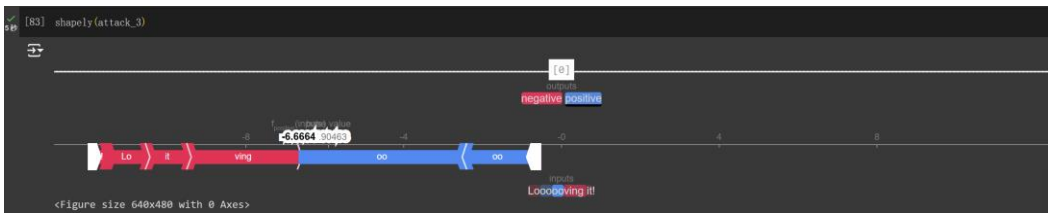
3. Misspelling the words

```
[79]  original_3=  "Loving  it!"
      attack_3  =  "Looooving  it!"
```





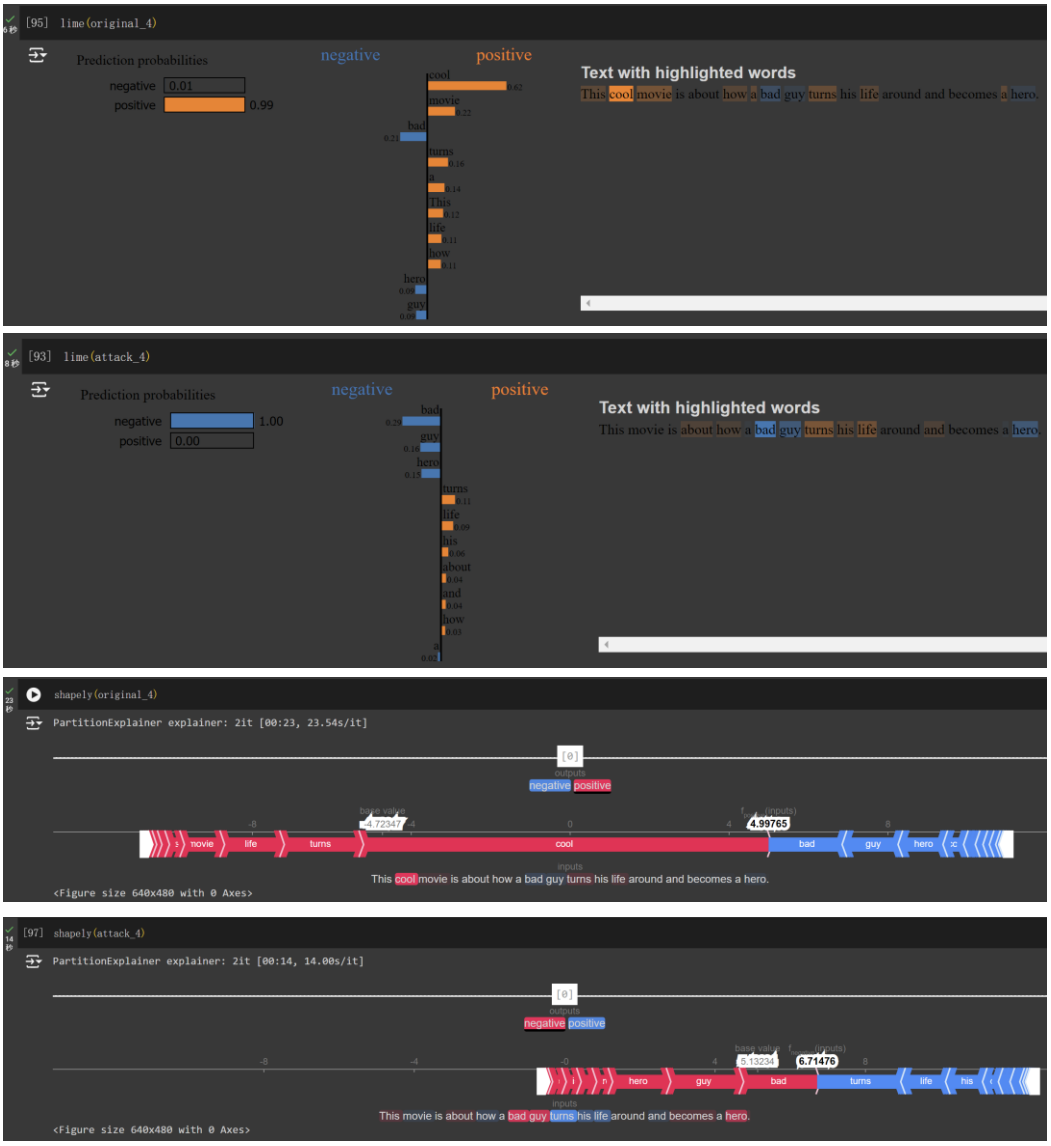




- The original text is obviously positive. However, in human typing, we often add multiple repeated letters to emphasize. Clearly, with LIME's explanation, the AI model cannot understand this and predicts it as negative.
- In SHAP, we can see that the letters in the word "looooooving" are highlighted individually, resulting in a roughly balanced prediction of positive and negative outcomes.

4. Others: delete the adjective
   - In this case, "cool" is the key word for determining the sentiment of the text. And we describe the movie's content with a negative word "bad".

```
[94]  original_4=  "This  cool  movie  is  about  how  a  bad  guy  turns  his  life  around  and  becomes  a  hero."
      attack_4  =  "This  movie  is  about  how  a  bad  guy  turns  his  life  around  and  becomes  a  hero."
```



- The result shows that if we remove "cool", the text lacks any positive or negative distinction and simply becomes a statement.
- However, LIME's explanation incorrectly identifies it as negative due to the presence of words like "bad" and "guy", whereas SHAP's explanation is more balanced and aligns better with the actual context.

5. How to prevent the attack:
   1) Add more content in input: For example, in the case of original_1, adding more reasoning or explanation sentence can help.
   2) Adjust the model to make it more sensitive to words with multiple meanings.
   3) Perform preprocessing to text, such as converting uppercase letters to lowercase or standardizing exaggerated words, to ensure fairer results.

4) Data Augmentation: Introduce perturbed data during training process so the model can handle disturbance better.
5) Check the Input Text: Perform tasks such as spell-checking and correction.

Problems

1. Transformer version:
   - Initially, in the hw4.ipynb file, an error was encountered when executing the code `‘path = 'model/TA_model_1.pt'’` and `‘model = torch.load(path, map_location=torch.device('cpu'))’`, which resulted in the error message: `'DistilBertModel' object has no attribute '_use_flash_attention_2'.`
   - Referring to the discussion on Teams, the issue was resolved by modifying the command `‘!pip install transformers’` to install a specific version, `‘!pip install transformers==4.30.0’`, which allowed the code to run.
2. The URL link “https://exbert.net/exBERT.html” in PART1 could not be accessed, so I can only use hugging-face version. However, the page is slightly difference from the tutorial video. I can’t find the “search by context” and “search by embedding” buttons, causing me a little bit confused. And I ignored that function at last.
3. Understanding how to operate exBERT and the underlying significance of each operation button took me some time to comprehend. I searched for information and articles online to get familiar with it.
4. When attempting to attack, initial trials such as transforming "I hate this film" into "eye h8 this film" or converting lowercase to uppercase did not yield significant differences in predicting results. Therefore, I try more diverse alterations to the text to achieve successful attacks.