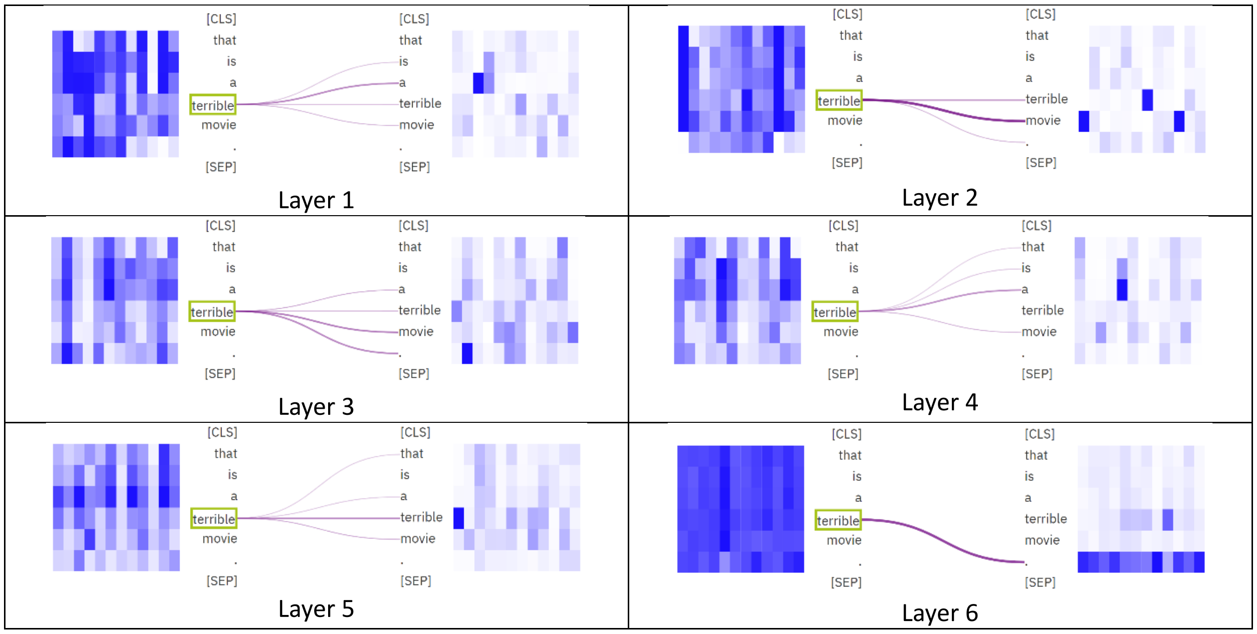
HW4 **111101013 蔡長頤**

Part 1: Attention Visualization:

1. DistilBert

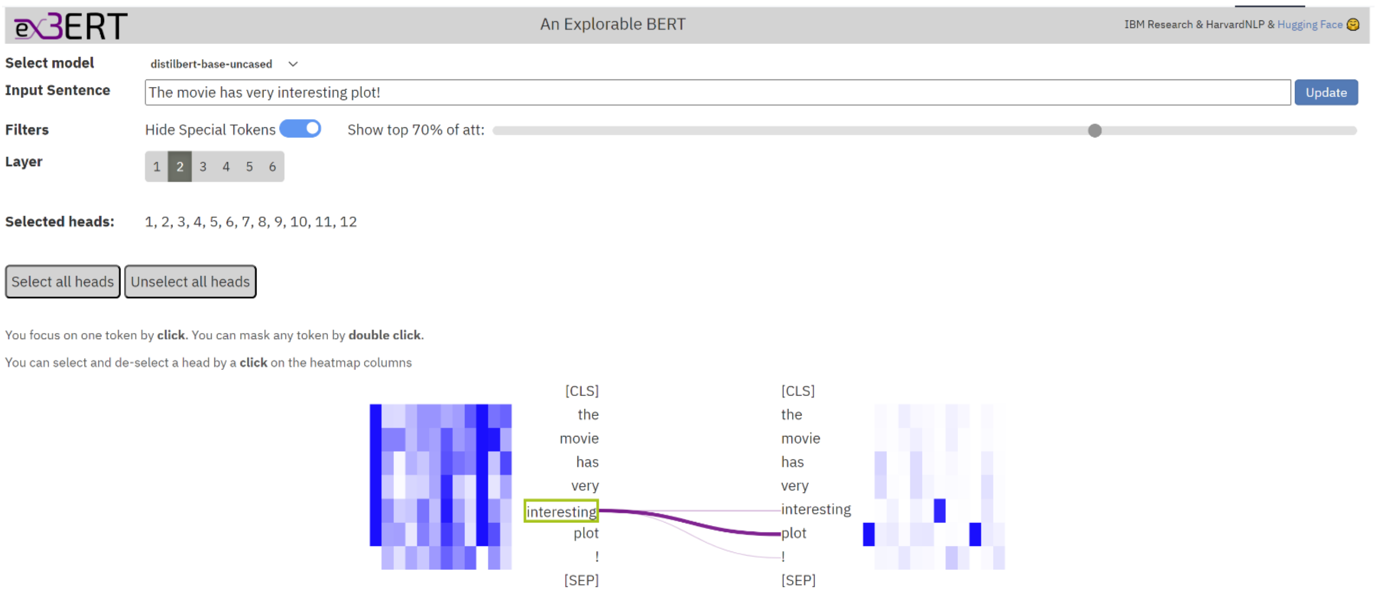
(1) Observation of the attention on different layers and different heads:
Input sequence: That is a terrible movie.



▲DistilBert

a. By observing how "terrible" distributes its attention to other tokens, we can reveal what contextual elements the model considers when interpreting the sentiment expressed by "terrible". In this example, the attentions between the sentiment keyword “terrible” and the subject “movie” are in the top 70% in all 6 layers. This implies that the word “movie” may be considered important when interpreting the sentiment of the word “terrible”.

b. Also, because different heads may learn to focus on different types of relationships. We can selectively look at individual heads to discover if some are particularly good at focusing on sentiment-relevant aspects.
We can observe that Head 1 at layer 2 in this example pays more attention to "terrible" and its relationship with "movie". This implies that head 2-1 (layer2, head1) may be particularly good at focusing on sentiment-relevant aspects. This implication was verified by trying multiple similar structured sentences, all with the structure “sentimental adjective (ex: interesting, worst, etc) + noun related to movie (ex: movie, plot, etc.)”. These similar structured sentences all have high attentions on head 2-1.


exBERT An Explorable BERT IBM Research & HarvardNLP & Hugging Face

Select model distilbert-base-uncased

Input Sentence The movie has very interesting plot! Update

Filters Hide Special Tokens Show top 70% of att:

Layer 1 2 3 4 5 6

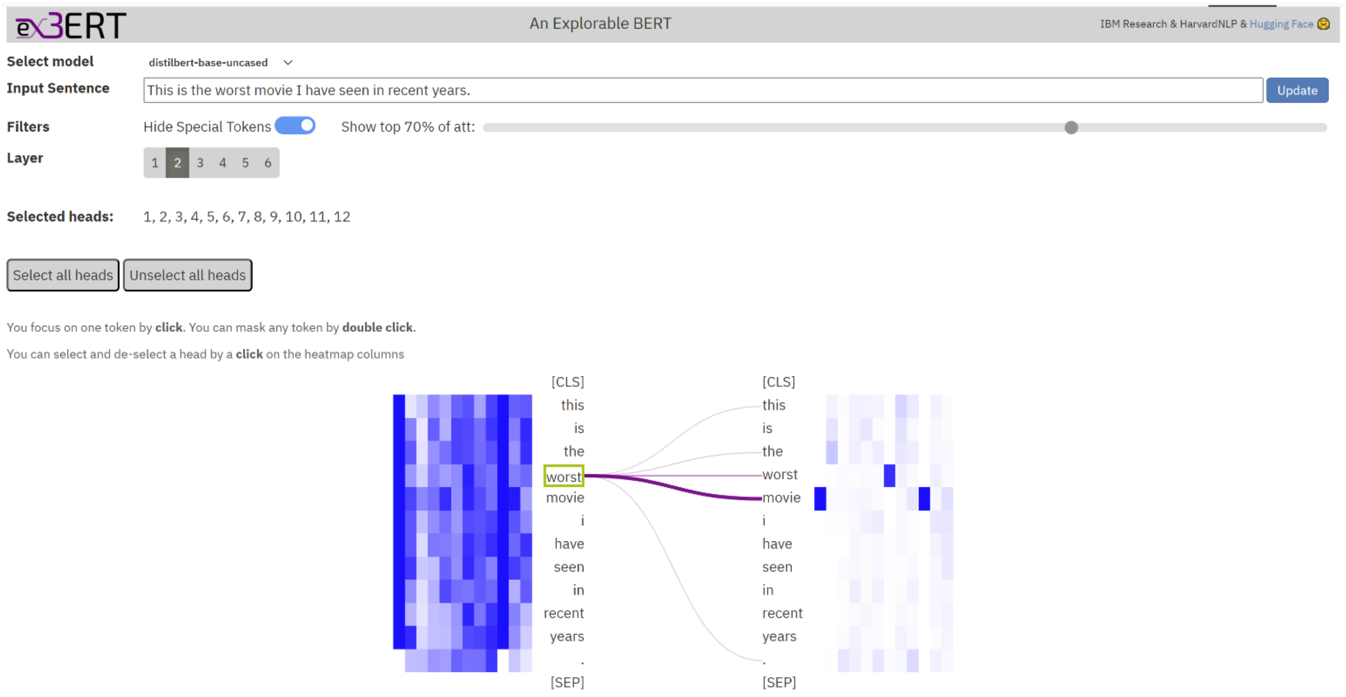
Selected heads: 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12

Select all heads | Unselect all heads

You focus on one token by **click**. You can mask any token by **double click**.

You can select and de-select a head by a **click** on the heatmap columns

▲Head 2-1 is very active with input sentence “The movie has very interesting plot!”


exBERT An Explorable BERT IBM Research & HarvardNLP & Hugging Face

Select model distilbert-base-uncased

Input Sentence This is the worst movie I have seen in recent years. Update

Filters Hide Special Tokens Show top 70% of att:

Layer 1 2 3 4 5 6

Selected heads: 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12

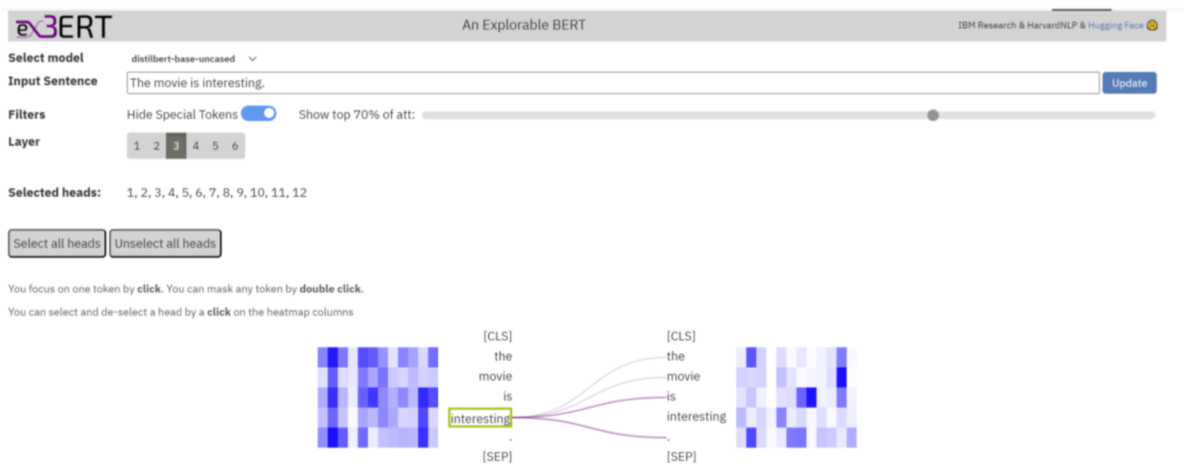
Select all heads | Unselect all heads

You focus on one token by **click**. You can mask any token by **double click**.

You can select and de-select a head by a **click** on the heatmap columns

▲Head 2-1 is very active with input sentence “This is the worst movie I have seen in recent years.”

c. However, we should note that, with different sentence structures, the focused head will be different. For example, rather than the “sentimental adjective + noun related to movie” structure, if the movie review has a structure “+ noun related to movie + ...is + sentimental adjective”, the most prominent head would be 3-11.



Select model distilbert-base-uncased

Input Sentence The movie is interesting.

Update

Filters Hide Special Tokens Show top 70% of att:

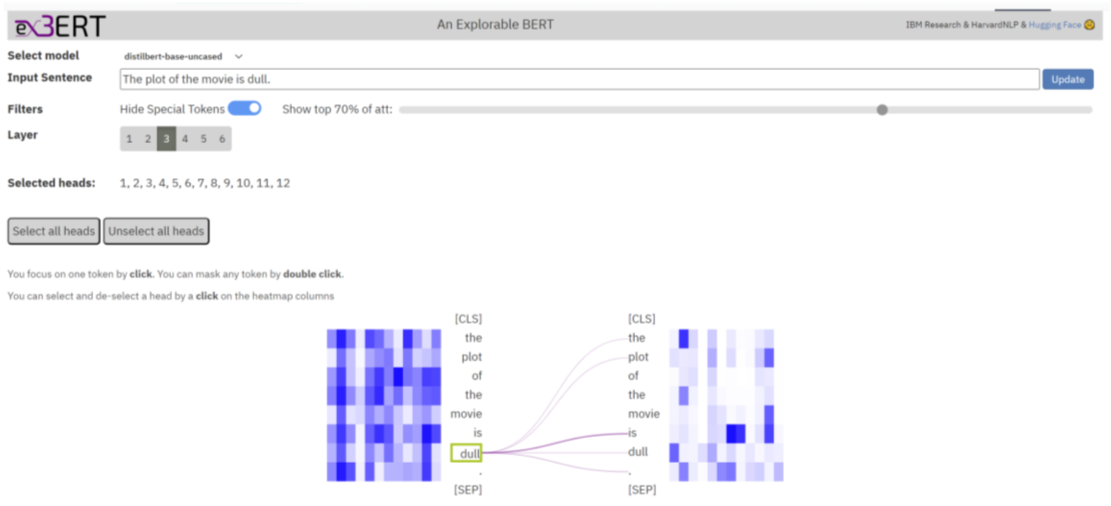
Layer 1 2 3 4 5 6

Selected heads: 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12

Select all heads Unselect all heads

You focus on one token by **click**. You can mask any token by **double click**.

You can select and de-select a head by a **click** on the heatmap columns



Select model distilbert-base-uncased

Input Sentence The plot of the movie is dull.

Update

Filters Hide Special Tokens Show top 70% of att:

Layer 1 2 3 4 5 6

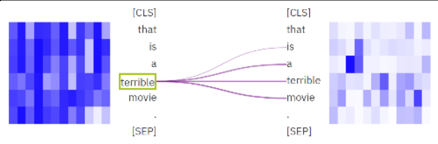
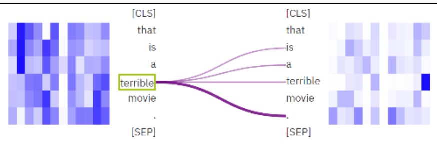
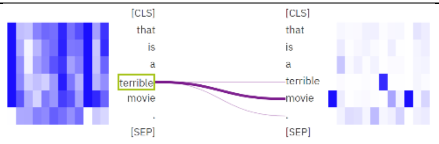
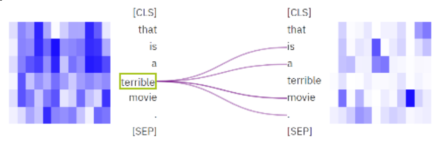
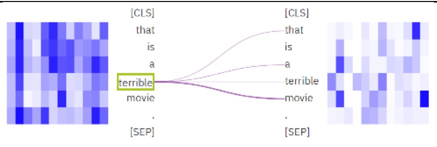
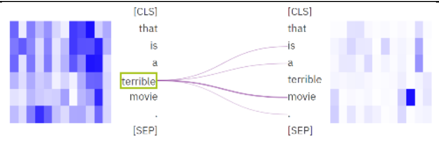
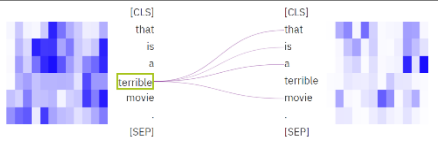
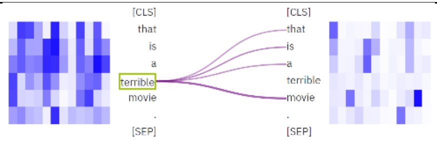
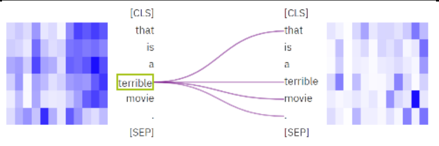
Selected heads: 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12

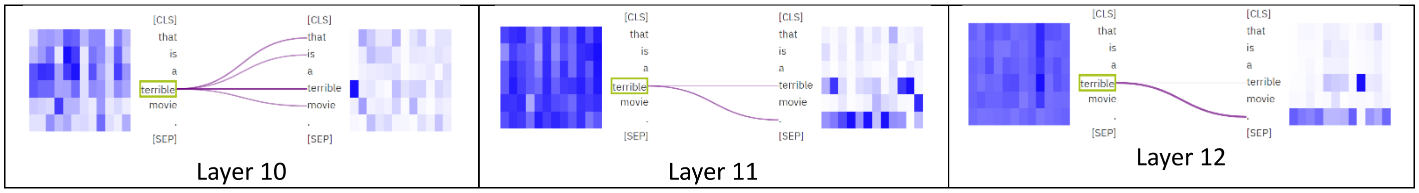
Select all heads Unselect all heads

You focus on one token by **click**. You can mask any token by **double click**.

You can select and de-select a head by a **click** on the heatmap columns

2. Bert v.s DistilBert

 Layer 1	 Layer 2	 Layer 3
 Layer 4	 Layer 5	 Layer 6
 Layer 7	 Layer 8	 Layer 9



▲Bert

With the same input sentence "That is a terrible movie.", we can observe that, because Bert is a more complicated model than DistilBert, not only in the 2nd layer, many layers like layer 3, 4, 5, 6, 7, 8, 11, contains heads with very high attention between the word "terrible" and "movie". This may imply that Bert is capable of capturing more relevance between keywords and may be able to classify movie reviews better than DistilBert, given the tradeoff of more computations.

3. GPT
   GPT is primarily designed for generation tasks, so it works differently from BERT-based models, and may not be as related to the sentiment analysis task.
   With the same input sentence "That is a terrible movie.", the GPT model would generate idea, thing, way, thought, situation when focusing on "terrible". These are all nouns, so we may say that GPT can understand the adjective(terrible) + noun structure in English.

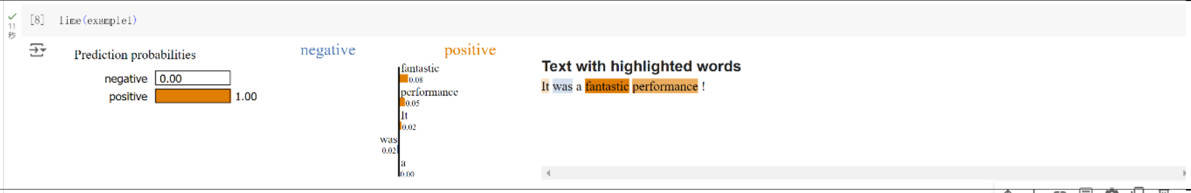
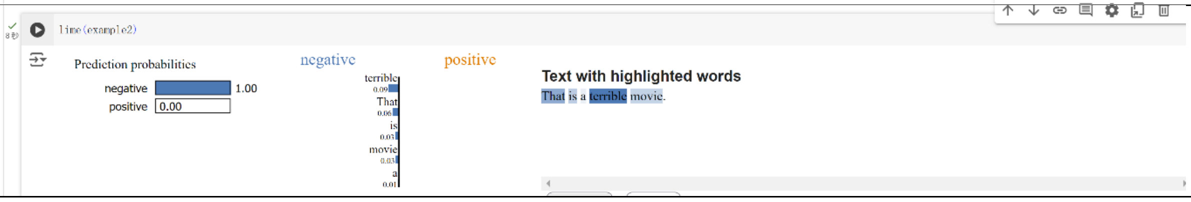
Part 2: LIME

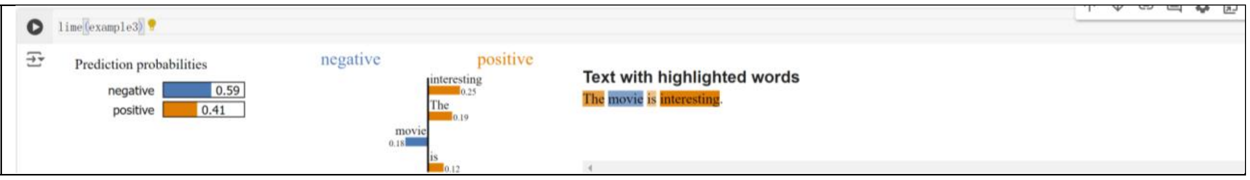
In Lime, the coefficients are used to show the contribution of each feature (word) to the prediction. Features that have a larger impact on the prediction will have larger coefficients.

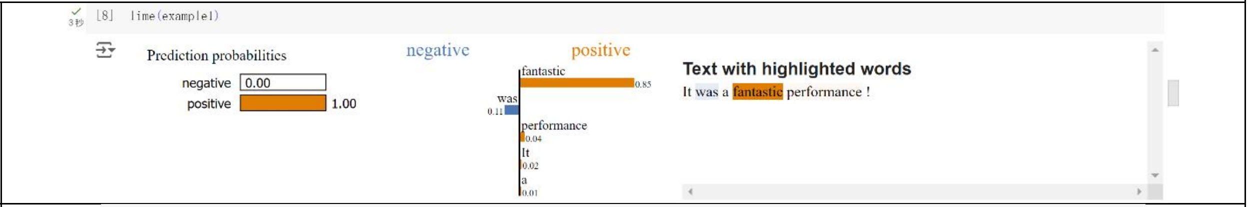
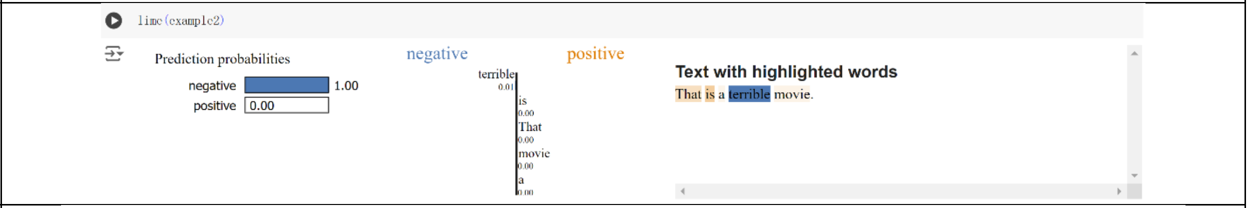
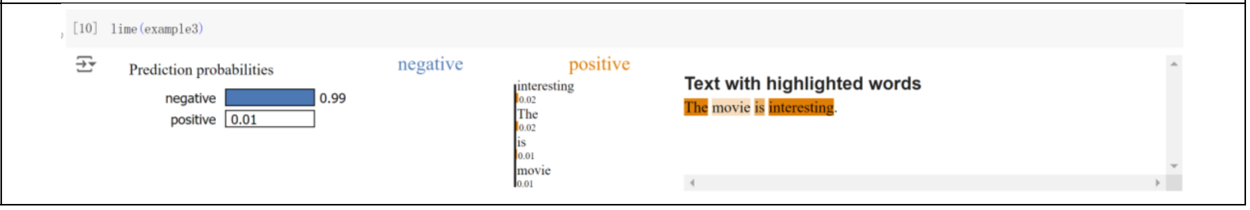
example1 = 'It was a fantastic performance!'

example2 = 'That is a terrible movie.'

example3 = 'The movie is interesting.'

TA_model_1	
Example1	 [8] lime(example1) Prediction probabilities negative 0.00 positive 1.00 negative positive fantastic 0.08 performance 0.05 It 0.02 was 0.02 a 0.00 Text with highlighted words It was a fantastic performance !
Example2	 lime(example2) Prediction probabilities negative 1.00 positive 0.00 negative positive terrible 0.09 That 0.06 is 0.03 movie 0.03 a 0.01 Text with highlighted words That is a terrible movie.

Example3	 `lime(example3)` Prediction probabilities: negative 0.59, positive 0.41; negative / positive: interesting 0.25, The 0.19, movie 0.18, is 0.12; Text with highlighted words: The movie is interesting.

TA_model_2	
Example1	 `[8] lime(example1)` Prediction probabilities: negative 0.00, positive 1.00; negative / positive: fantastic 0.85, was 0.11, performance 0.04, It 0.02, a 0.01; Text with highlighted words: It was a fantastic performance !
Example2	 `lime(example2)` Prediction probabilities: negative 1.00, positive 0.00; negative / positive: terrible 0.01, is 0.00, That 0.00, movie 0.00, a 0.00; Text with highlighted words: That is a terrible movie.
Example3	 `[10] lime(example3)` Prediction probabilities: negative 0.99, positive 0.01; negative / positive: interesting 0.02, The 0.02, is 0.01, movie 0.01; Text with highlighted words: The movie is interesting.

We can see the both TA_model_1 and TA_model_2's predictions were pretty accurate, and the sentiment-relevant keyword such as "fantastic", and "terrible" have larger coefficients, which indicates the model is paying attention to important features. However, to understand the performance of the model more, like in part 1, I tried different sentence structures.

For example, example3 = 'The movie is interesting.'

This is an intriguing case. We see that TA_model_1 and TA_model_2 both made the wrong prediction. Also, TA_model_1 somewhat performed better than TA_model_2, as the prediction probabilities of negative and positive were close, in contrast with the fact that TA_model_2 predicted example 3 as negative confidently (incorrect).

Furthermore, with this example, we can see a discrepancy between the LIME coefficients and the prediction. This discrepancy may be due to how models are approximated in Lime. Lime creates a simple model(decision tree or linear model) to approximate the predictions of the complex model locally around the input text. The main disadvantages when it comes to using LIME for NLP purposes is that it has been found to be unstable, meaning that different sampling around the same local data can lead to very different explanation results (Molnar, 2019), and that LIME only provides local interpretability. Furthermore, since LIME creates a linear local model around the instance of interest, it assumes that the local instant can be interpreted linearly, which is not always the case.

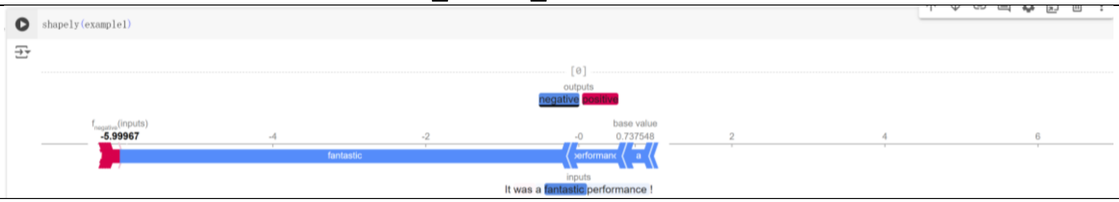
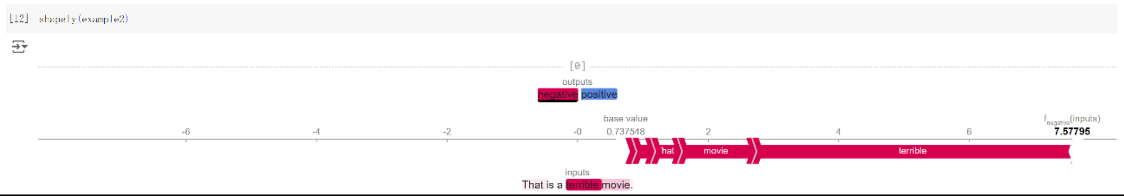
However, as we can see above, at least in terms of local interpretability, most of the feature weights make sense in terms of our own understanding of disaster-related words, and the LIME explainer itself has a short runtime meaning it is easy to produce many explanations for different local predictions in a short space of time, if needed.

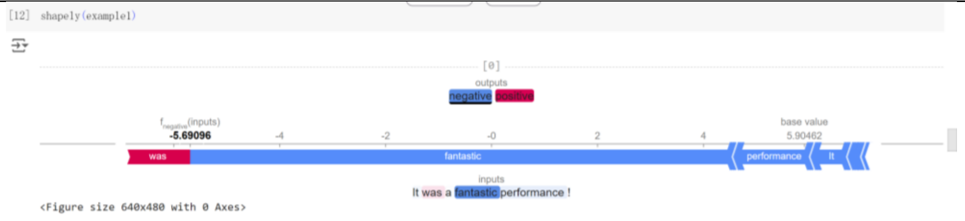
Part 3: SHAP

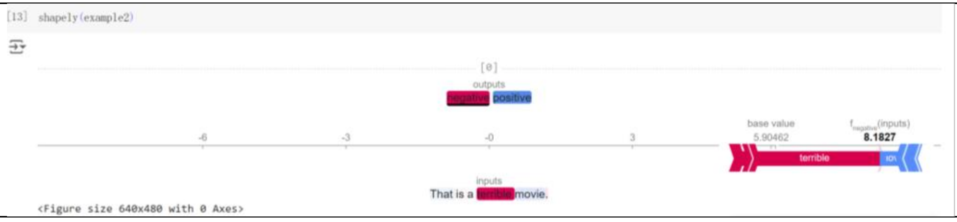
The SHAP explanation model is additive, meaning it decomposes a prediction into the sum of effects from each feature being introduced into a conditional expectation. The final prediction for a specific instance can be obtained by summing the SHAP values of all features along with the base value: final_prediction = base_value + sum(SHAP values of all features).

example1 = 'It was a fantastic performance!'

example2 = 'That is a terrible movie.'

TA_model_1	
Example1	 `shapely(example1)`
Exampe2	 `[12] shapely(example2)`

TA_model_2	
Example1	 `[12] shapely(example1)` `<Figure size 640x480 with 0 Axes>`

Example2	 `shapely(example2)` `<Figure size 640x480 with 0 Axes>` [0] outputs negative positive -6 -3 -0 3 base value 5.90462 f negative(inputs) 8.1827 terrible inputs That is a terrible movie.

In both models, we can see that in example1, "fantastic" has a significant negative contribution to the output, pushing the baseline value toward the specific prediction value of the input: $f_{negative}$. This is expected, because as a positive word, its contribution is pushing the prediction away from the possibility of predicting the example as a negative class ($f_{negative}$ is decreased).

In contrast, in example2, "terrible" pushed $f_{negative}$ value towards positive. This means that the model is likely to predict negative (because $f_{negative}$ is increased) mostly because the word "terrible". On the other hand, we can observe that the effect of "terrible" was more significant in in TA_model_1 than TA_model_2.

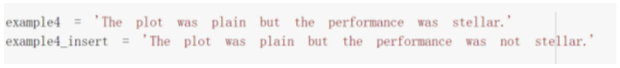
Part 4: Attack

In this part, I worked on TA_model_1

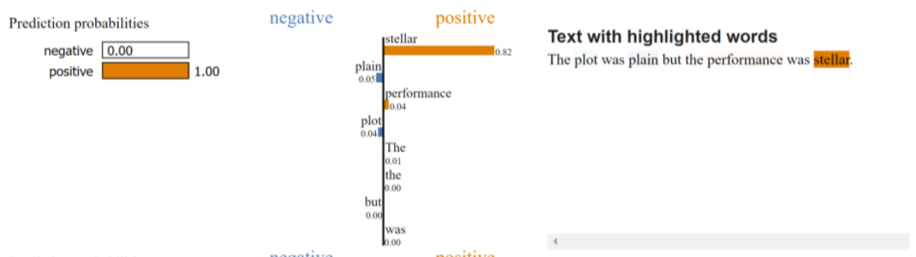
1. Word insertion

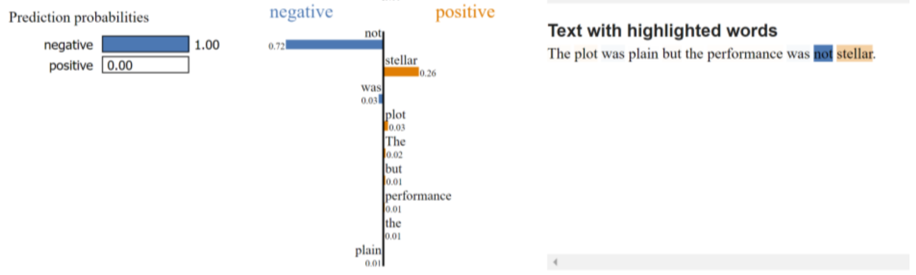
By inserting only a single word "not", we can sometimes invert the meaning of the sentence.

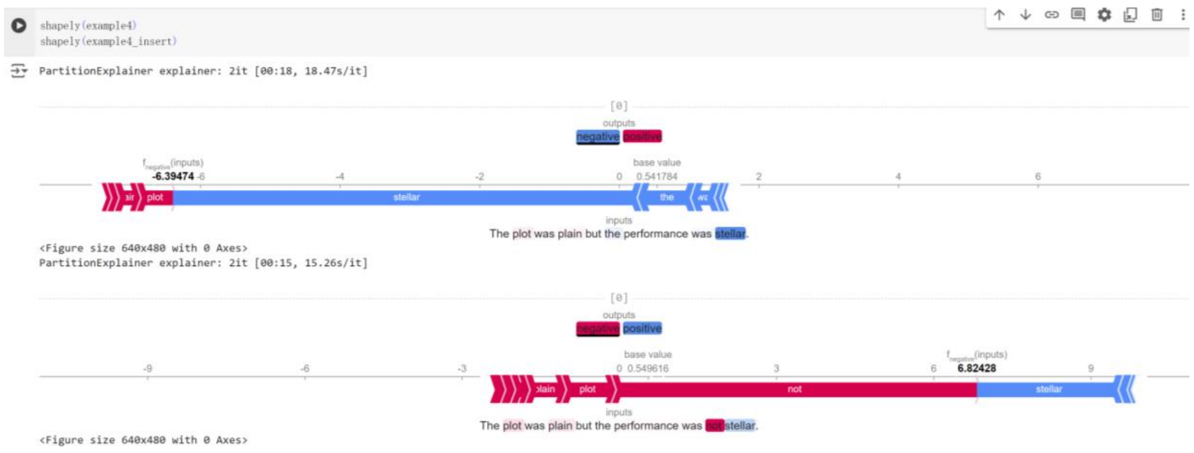
In example 4, by inserting the negating word "not" before, the word "not" dominates the negative prediction, and the result of LIME and SHAP is totally inverted.



```
example4 = 'The plot was plain but the performance was stellar.'
example4_insert = 'The plot was plain but the performance was not stellar.'
```







```
shapely(example4)
shapely(example4_insert)
```

```
PartitionExplainer explainer: 2it [00:18, 18.47s/it]
```

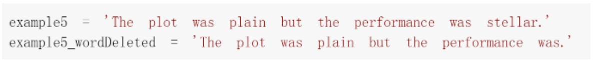
```
<Figure size 640x480 with 0 Axes>
PartitionExplainer explainer: 2it [00:15, 15.26s/it]
```

```
<Figure size 640x480 with 0 Axes>
```

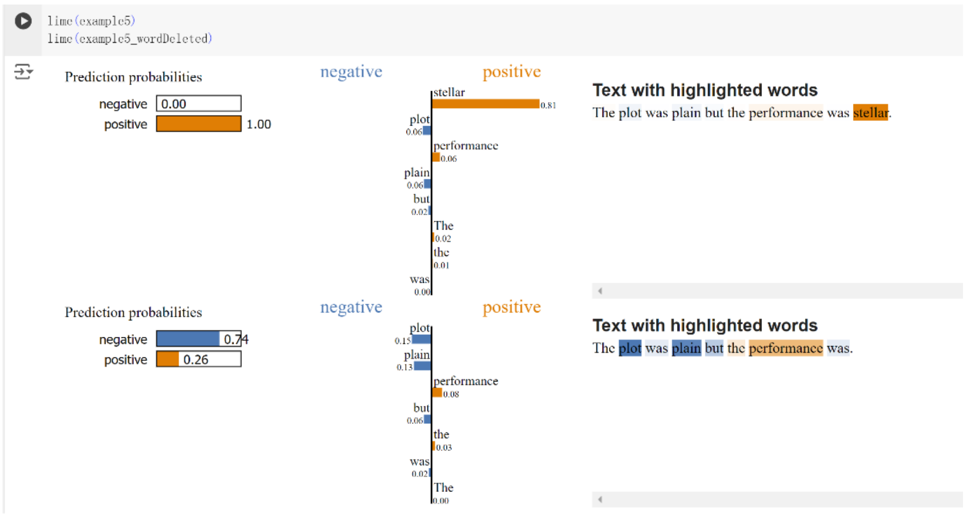
2. Word deletion

By removing a key adjective, we can observe how the model handles incomplete information.

In example5, by deleting the positive keyword “stellar”, but keeping the word “but”, which indicates an opposite meaning of the fronter and back part of the sentence, Lime was attacked and made a different prediction. In the attacked sentence, “plain” was captured as the prominent negative feature, despite that the model captured some sense of the logic of “but” and assigned some positive value after the words after “but”.



```
example5 = 'The plot was plain but the performance was stellar.'
example5_wordDeleted = 'The plot was plain but the performance was.'
```



```
lime(example5)
lime(example5_wordDeleted)
```

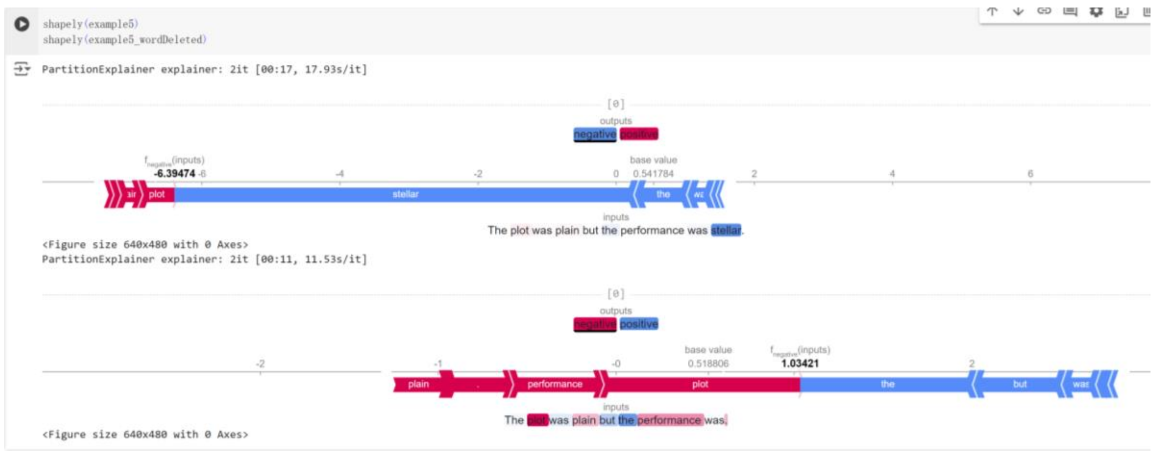
Text with highlighted words

The plot was plain but the performance was stellar.

Text with highlighted words

The plot was plain but the performance was.

In SHAP, deleting a word also greatly influenced the result, indicating that deleting a word may be an efficient way to attack.



```
shapely(example5)
shapely(example5_wordDeleted)
```

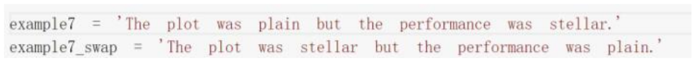
```
PartitionExplainer explainer: 2it [00:17, 17.93s/it]
```

```
<Figure size 640x480 with 0 Axes>
PartitionExplainer explainer: 2it [00:11, 11.53s/it]
```

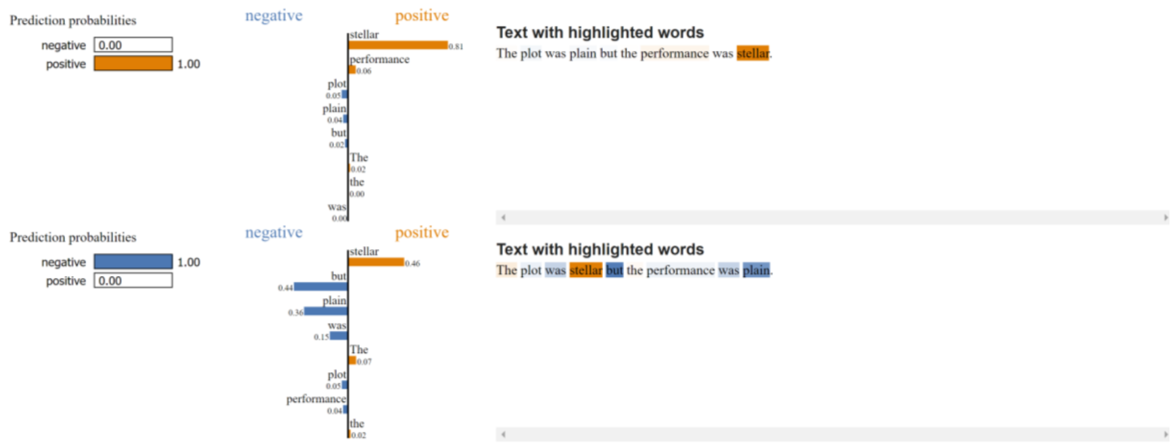
```
<Figure size 640x480 with 0 Axes>
```

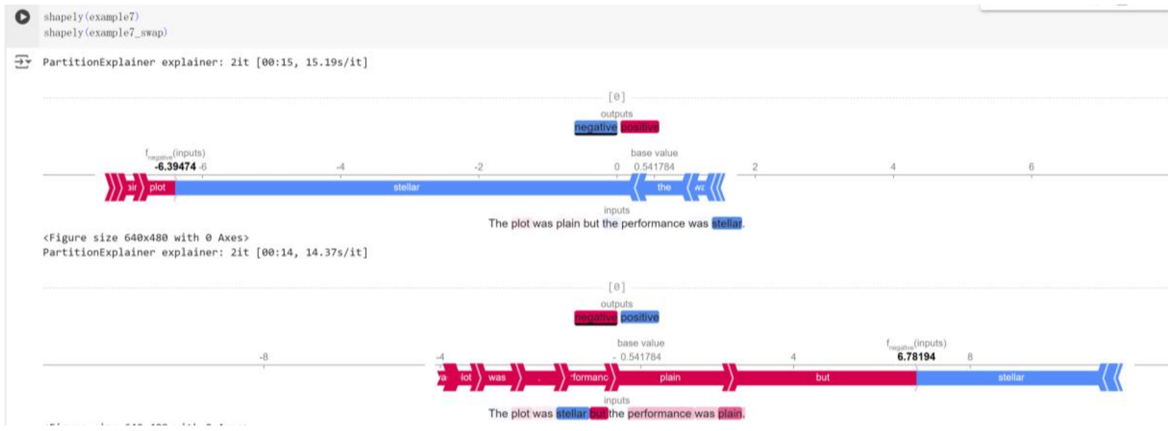
3. Swap words:

By swapping the order of the keywords "plain"(negative) and "stellar"(positive), the prediction is totally inverted as expected, because the order of words before and after the conjunction "but" is crucial to the meaning of English sentences.



```
example7 = 'The plot was plain but the performance was stellar.'
example7_swap = 'The plot was stellar but the performance was plain.'
```





```
shapely(example7)
shapely(example7_swap)
```

```
PartitionExplainer explainer: 2it [00:15, 15.19s/it]
```

```
<Figure size 640x480 with 0 Axes>
PartitionExplainer explainer: 2it [00:14, 14.37s/it]
```

[0]
outputs
negative positive
base value
0.541784
f negative(inputs)
6.78194
-8 -4 4 8
a lot was . formanc plain but stellar
inputs
The plot was stellar but the performance was plain.

4. Model attack prevention:
   (1) Adversarial Training:
   Incorporate adversarial examples into the training set. By training the model on both clean and perturbed data, it becomes more robust to slight modifications intended to deceive the model.
   (2) Input Sanitization:
   Before processing inputs, use techniques like spell-checking and grammar validation to correct or reject inputs that seem crafted to exploit the model's weaknesses.
   (3) Regularization Techniques:
   Implement techniques like dropout or L2 regularization to reduce model complexity and overfitting, which can make the model less sensitive to slight input modifications.
   (4) Model Monitoring:
   Continuously monitor the model's performance on new data. Set up alerts for unusual patterns in predictions that might indicate an attack, allowing to respond quickly.
   (5) Explainability and Transparency:
   Use explainability tools, like LIME or SHAP, to understand how the model is making decisions. Insights from these tools can help identify vulnerabilities or biases in the model that attackers could exploit.

Part 5: Problems I met and how I solved them.

1. Understanding SHAP Interpretations:
   Initially, interpreting the results provided by SHAP was challenging. The outputs, which focus on the impact of individual features on the model's decision-making process, were not intuitive to me. This confusion arose particularly in understanding how positive and negative SHAP values affect the prediction outcome, especially since SHAP values indicate the direction and magnitude of a feature's impact on the prediction relative to a baseline. To overcome this, I engaged in extensive research, including reading articles and watching tutorial videos on platforms like Medium and YouTube. These resources helped clarify how SHAP values work and their implications for model predictions, which improved my ability to accurately interpret and explain the results.
2. Adversarial Attack Implementation:
   Implementing adversarial attacks posed a significant challenge. Crafting effective adversarial examples that could alter the model's predictions without deviating significantly from the original input required a delicate balance. I experimented with various techniques such as typo, word insertion, deletion, and swapping, which required

careful consideration of the linguistic structure and semantic content of sentences because not all the methods or examples could generate effective attack effects.

3. Technical Issues with loading HW2 models:
   During the assignment, I initially wanted to try my own HW2 model. I spent a lot of time and finally could download the model into .pt files, however, I failed loading them due to unexpected issues. Unfortunately, I couldn't resolve the problem at the end.