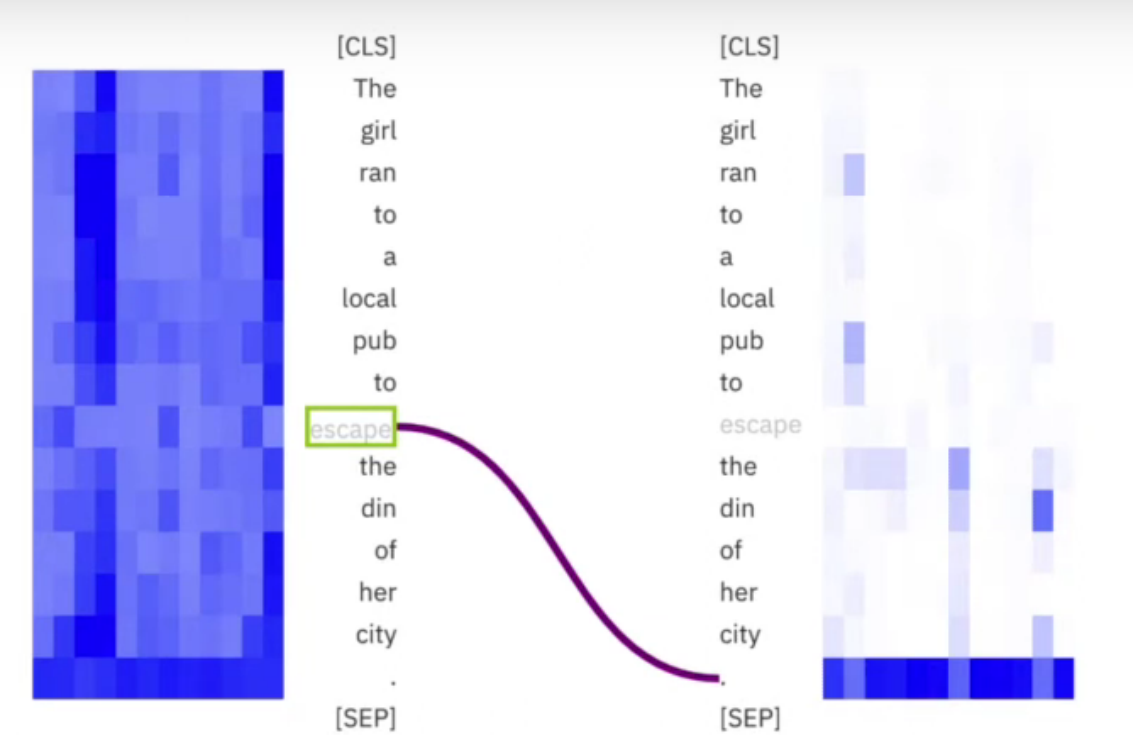
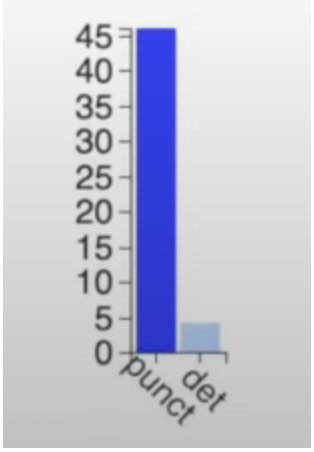
hw4_109700046

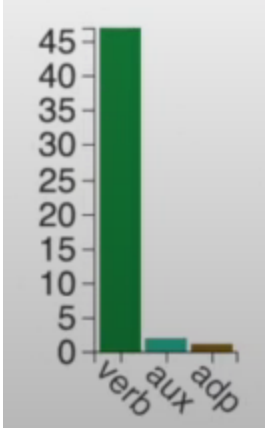
The report should at least include the following items.

1. **Describe your understanding and findings about the attention mechanism by exBERT. (20%)**
    - → exBERT is a visualization tool designed to dive deeper into the attention mechanisms behind BERT, analyzing and having better understand of the model.
    - In BERT, the attention mechanism calculates the importance of each token relative to all other tokens in the input sequence. This is done using self-attention, where the model looks at different positions within the sequence to collect relevant information. The attention weights derived from this process indicate how much attention each token should pay to every other token.
        - exBERT dives into the attention patterns learned by BERT and discovers how they contribute to model prediction, uncovering the reason and decision-making process behind BERT by analyzing the attention weights assigned to different tokens.
    - For example, when we do word masking test, we'll notice that the model become more predictable as the layers go deeper, being able to precisely determine the part of speech or other properties of the masked word.





Predict POS of ‘escape’ at layer 1

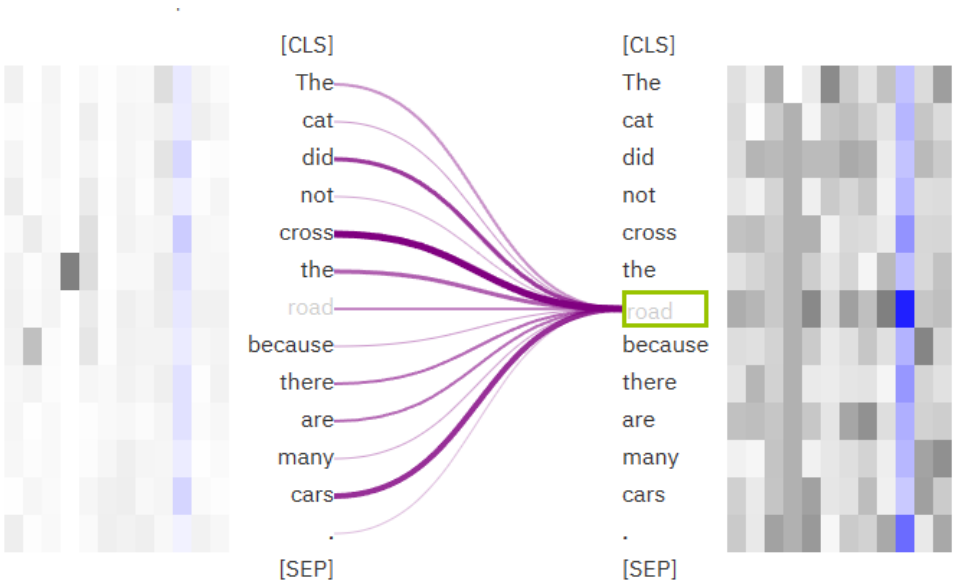


Predict POS of ‘escape’ at layer 12

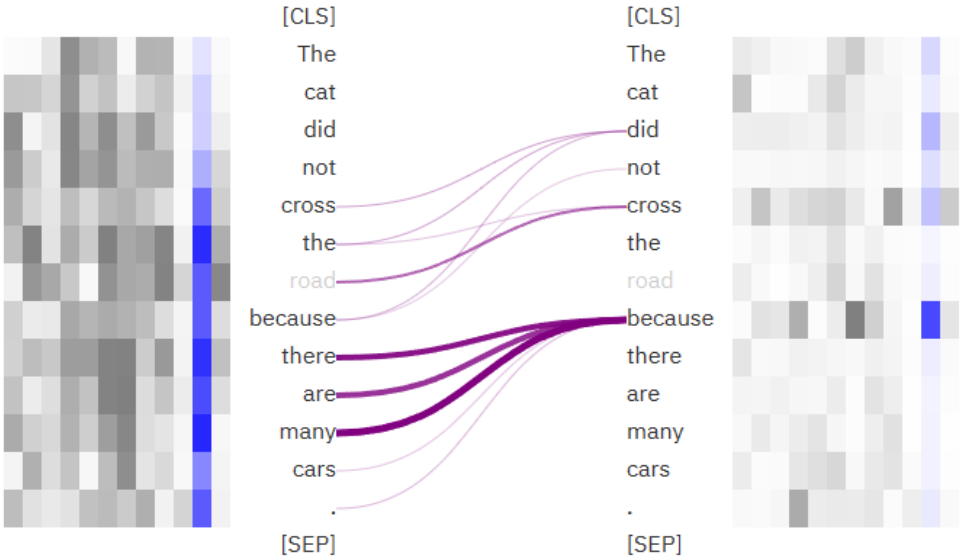
- Theoretically:
  - Early Layers → Capture basic linguistic features such as Parts of Speech (POS) and simple syntactic dependencies. These layers often focus on local context, understanding immediate neighboring tokens, but usually not as ‘accurate’ as expected.
  - Middle Layers → Focus on more complex dependencies and relationships, capturing longer-range interactions between tokens. They start to abstract higher-level features.
  - Later Layers → Refine specific patterns and relationships, often specializing in very particular aspects of the input data. These layers contribute to the model's final understanding used for tasks like classification or generation.

- Next, I primarily adjust the number of layers and selected heads to observe differences in the visualization patterns. In a multi-head attention mechanism, the input is processed through multiple parallel attention heads. Each head learns different patterns and captures various types of dependencies or relationships. This allows the model to focus on different aspects of the context simultaneously.
  - Testing
  - input sequence → ‘The cat did not cross the road because there are many cars.’
    - Choose layer = 3

- selected head = 10

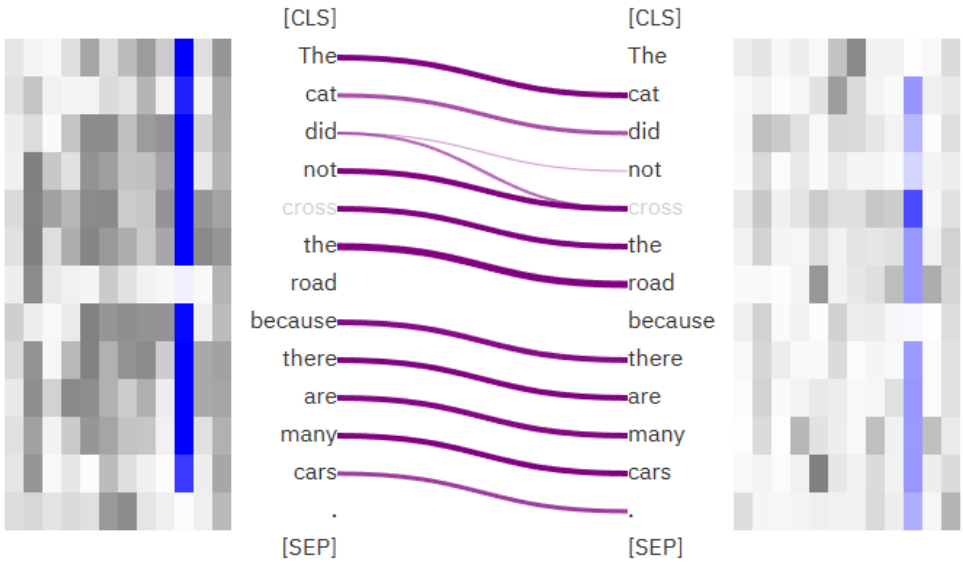


  - We focus on the word ‘road’, and we’ll notice that the word ‘cross’ contributes the most, followed by the word ‘cars’, in predicting the word ‘road’ based on the distributed weights.
- Choose layer = 8
- selected head = 11



- We notice that the phrase 'there are many' strongly contributes to the word 'because' since they have a cause-and-effect relationship, and the attention mechanism captures that.

- Choose layer = 7
  - selected head = 10



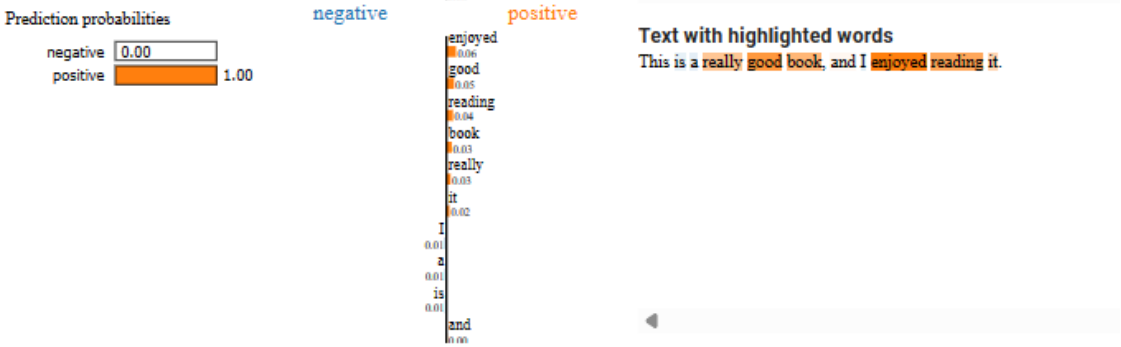
  - We can see that the masked word to be predicted is in highly relevant to its preceding word, and so do the other words that are not masked.

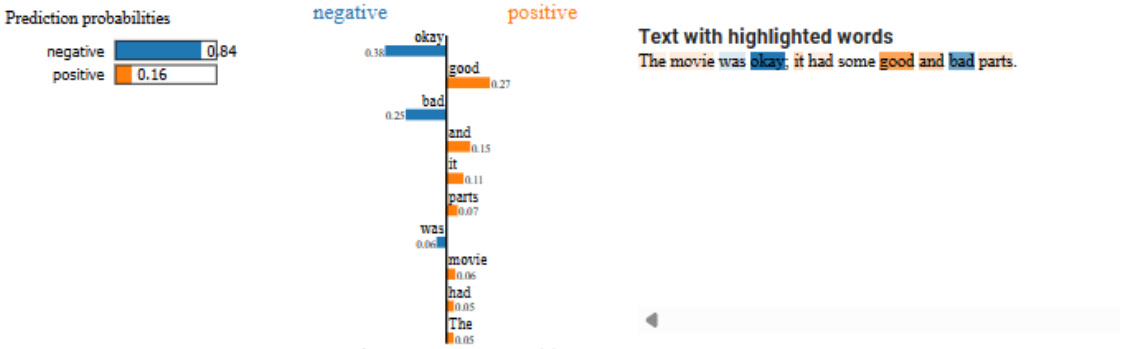
2. **Compare at least 2 sentiment classification models (e.g., TA_model_1.pt, your model in HW2). (20%)**

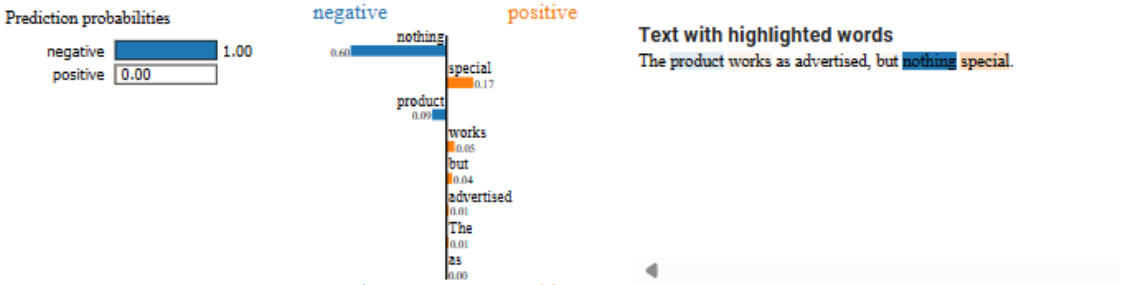
   - Test set
     - highlyPositive = 'It was a fantastic performance !'
     - Positive = 'This is a really good book, and I enjoyed reading it.'
     - Neutral1 = 'The movie was okay; it had some good and bad parts.'
     - Neutral2 = 'The product works as advertised, but nothing special.'
     - highlyNegative = 'The experience was terrible, and I will never go back there again.'
       Negative = 'I was disappointed with the quality of the item.'

    - Negative = 'I was disappointed with the quality of the item.'
- Lime

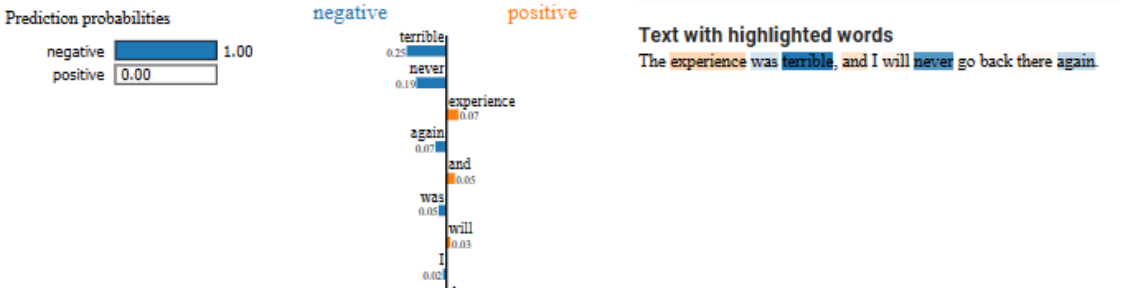

Prediction probabilities
negative 0.00
positive 1.00
negative
positive
fantastic 0.08
performance 0.05
It 0.03
was 0.03
a 0.00
Text with highlighted words
It was a fantastic performance !


Prediction probabilities
negative 0.00
positive 1.00
negative
positive
enjoyed 0.06
good 0.05
reading 0.04
book 0.03
really 0.03
it 0.02
I 0.01
a 0.01
is 0.01
and 0.00
Text with highlighted words
This is a really good book, and I enjoyed reading it.

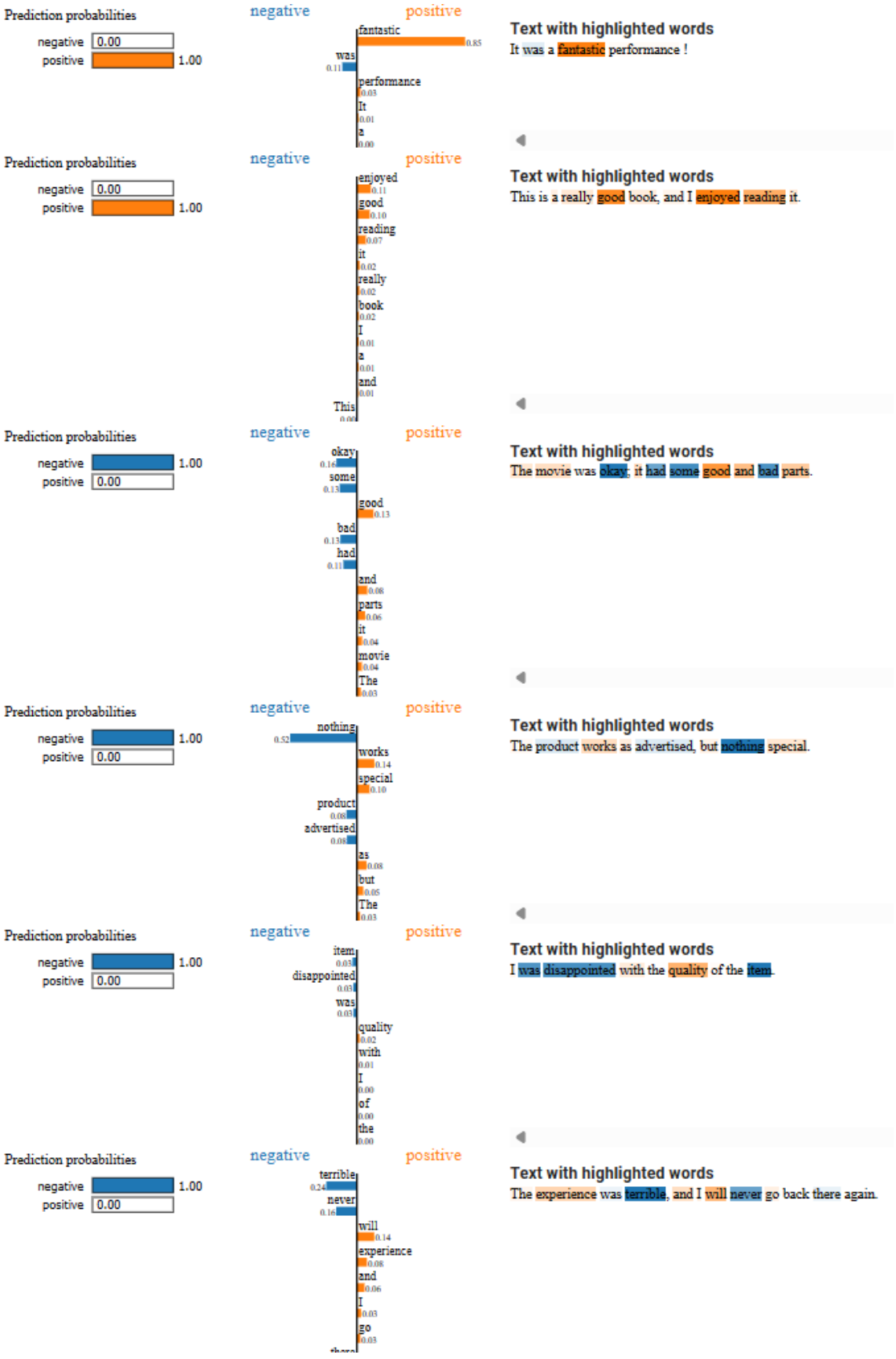

Prediction probabilities
negative 0.84
positive 0.16
negative
positive
okay 0.38
good 0.27
bad 0.25
and 0.15
it 0.11
parts 0.07
was 0.06
movie 0.06
had 0.05
The 0.05
Text with highlighted words
The movie was okay, it had some good and bad parts.


Prediction probabilities
negative 1.00
positive 0.00
negative
positive
nothing 0.60
special 0.17
product 0.09
works 0.05
but 0.04
advertised 0.01
The 0.01
as 0.00
Text with highlighted words
The product works as advertised, but nothing special.


Prediction probabilities
negative 1.00
positive 0.00
negative
positive
disappointed 0.65
quality 0.12
item 0.08
the 0.07
of 0.03
was 0.02
I 0.02
with 0.01
Text with highlighted words
I was disappointed with the quality of the item.


Prediction probabilities
negative 1.00
positive 0.00
negative
positive
terrible 0.25
never 0.19
experience 0.07
again 0.07
and 0.05
was 0.05
will 0.03
I 0.02
Text with highlighted words
The experience was terrible, and I will never go back there again.

lime, TA_model_1.pt

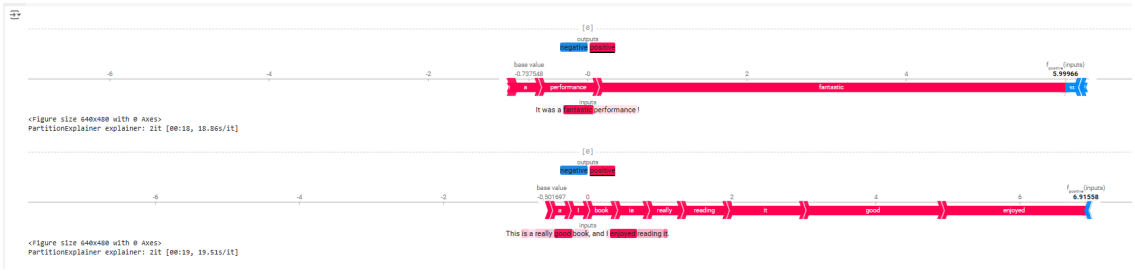

Prediction probabilities
negative 0.00
positive 1.00
negative
positive
fantastic 0.85
was 0.11
performance 0.03
It 0.01
a 0.00
Text with highlighted words
It was a fantastic performance !
Prediction probabilities
negative 0.00
positive 1.00
negative
positive
enjoyed 0.11
good 0.10
reading 0.07
it 0.02
really 0.02
book 0.02
I 0.01
a 0.01
and 0.01
This 0.00
Text with highlighted words
This is a really good book, and I enjoyed reading it.
Prediction probabilities
negative 1.00
positive 0.00
negative
positive
okay 0.14
some 0.13
good 0.13
bad 0.13
had 0.11
and 0.08
parts 0.06
it 0.04
movie 0.04
The 0.03
Text with highlighted words
The movie was okay, it had some good and bad parts.
Prediction probabilities
negative 1.00
positive 0.00
negative
positive
nothing 0.52
works 0.14
special 0.10
product 0.08
advertised 0.08
as 0.08
but 0.05
The 0.03
Text with highlighted words
The product works as advertised, but nothing special.
Prediction probabilities
negative 1.00
positive 0.00
negative
positive
item 0.03
disappointed 0.03
was 0.03
quality 0.02
with 0.01
I 0.00
of 0.00
the 0.00
Text with highlighted words
I was disappointed with the quality of the item.
Prediction probabilities
negative 1.00
positive 0.00
negative
positive
terrible 0.24
never 0.16
will 0.14
experience 0.08
and 0.06
I 0.03
go 0.03
Text with highlighted words
The experience was terrible, and I will never go back there again.

lime, TA_model_2.pt

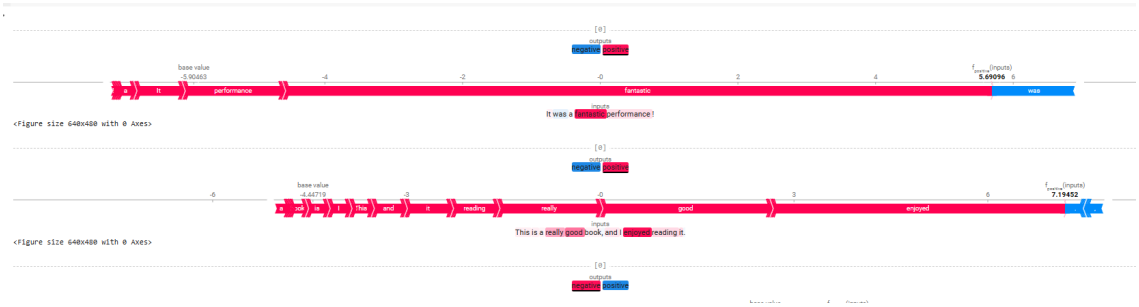
First, both of the models aren't able to classify neutral sentence properly (0.5, 0.5), so neutral sentence should not be suitable for these two class sentiment models.

In my opinion, from the above test cases, TA_model_1.pt tends to give the words it thinks pretty negative a high negative score (4th ('nothing':0.6) and 5th ('disappointed':0.65) examples), on the other hand, TA_model_2.pt is encouraged to give higher score to those words that are considered to have high positive feedback (1th example ('fantastic':0.85)). For TA_model_1.pt, it may have a higher probability that it will lead to False Negative than TA_model_2.pt, on the other hand, TA_model_2.pt may have a higher probability that it will lead to False Positive than TA_model_1.pt.

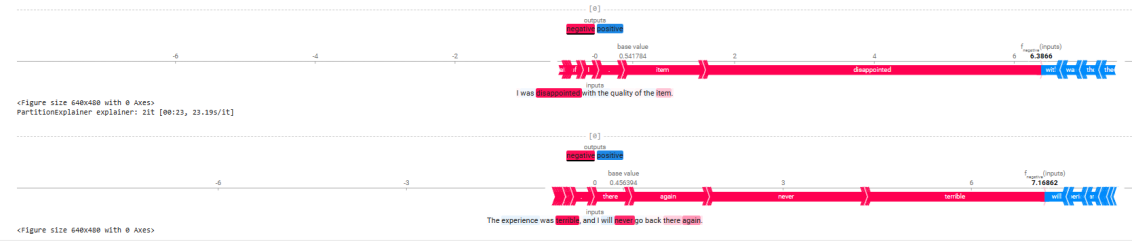
- shapely



TA_model_1.pt, positive results



TA_model_2.pt, positive results



TA_model_1.pt, negative results

TA_model_2.pt, negative results

As we have assumed in Lime, we can notice that TA_model_1.pt have a higher base value for negative results and a lower base value for negative results, vice versa for TA_model_2.pt, which can somehow prove our assumption that TA_model_1.pt tens to False Negative more and TA_model_2.pt tends to False Positive more.

3. **Compare the explanation of LIME and SHAP. (20%)**

   LIME focus on creating local approximations to understand a single prediction, it approximates the behavior of the complex model with a simpler and explainable model, and this model explains how the complex model behaves around the specific instance being analyzed.

   - LIME generates a dataset of perturbed samples around the instance being explained. training an interpretable model like a linear model on these perturbed instances to approximate the behavior of the model locally.

   SHAP connect game theory with local explanations to provide a unified measure of feature importance. It leverages the concept of Shapley values, which come from cooperative game theory for the sake of fairly distributing the prediction output among the features. SHAP values determine the importance of a feature by averaging its impact on the prediction over all possible ways that feature could be included or excluded in the model's input,

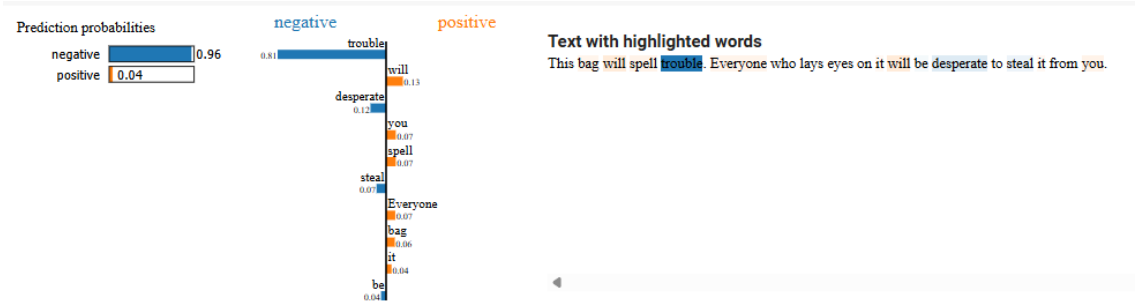
ensuring a fair and consistent attribution of the prediction to the individual features.

LIME and SHAP differ in how they explain ML model prediction, LIME provides explanations by generating feature importance weights for input variables, indicating how much each feature influences a specific prediction. For the instance in our homework, LIME shows that the word ‘fantastic’ contributes positively to the classification, while the word ‘nothing’ has negative classification. On the other hand, SHAP offers a more comprehensive approach by providing features attribution based on Shapley values from game theory, quantifying the importance of each feature, not only for a specific prediction but also in the context of the overall model. and we’ll see the representation of the contribution of each feature to the final prediction. We can already see their difference from the result in problem 2., except from providing the contribution of each word in the context, SHAP also has a base value so that the predictions won’t be only based on the weighted sum of positive and negative feedback, proving a more robust and comprehensive assessment.

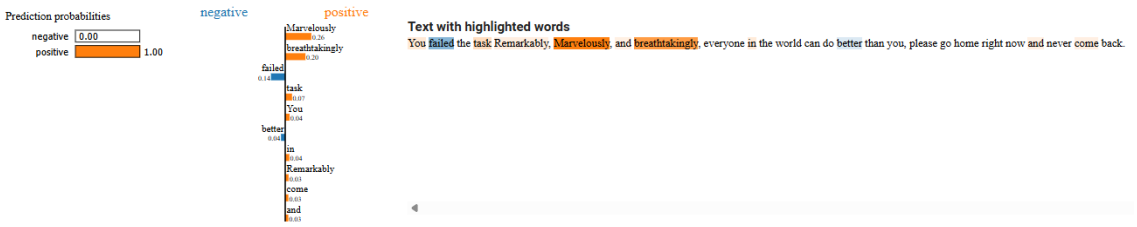
Summing up, LIME generates local explanations for a prediction by perturbation-based methods, while SHAP generate local and global explanations by assigning Shapley values based on some game theory concepts.

4. **Try 3 different input sentences for attacks. Also, describe your findings and how to prevent the attack if you retrain the model in the future. (20%)**

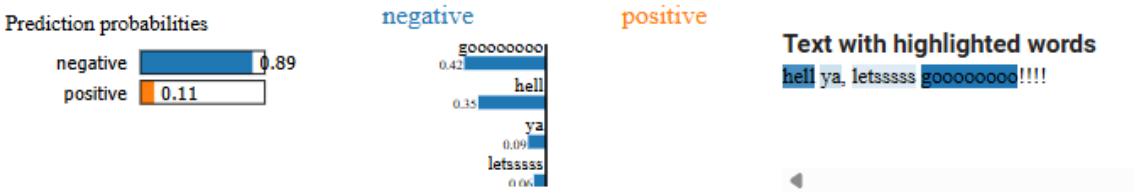
    - Attack 1 (Positive)
        - ‘This bag will spell trouble. Everyone who lays eyes on it will be desperate to steal it from you.’

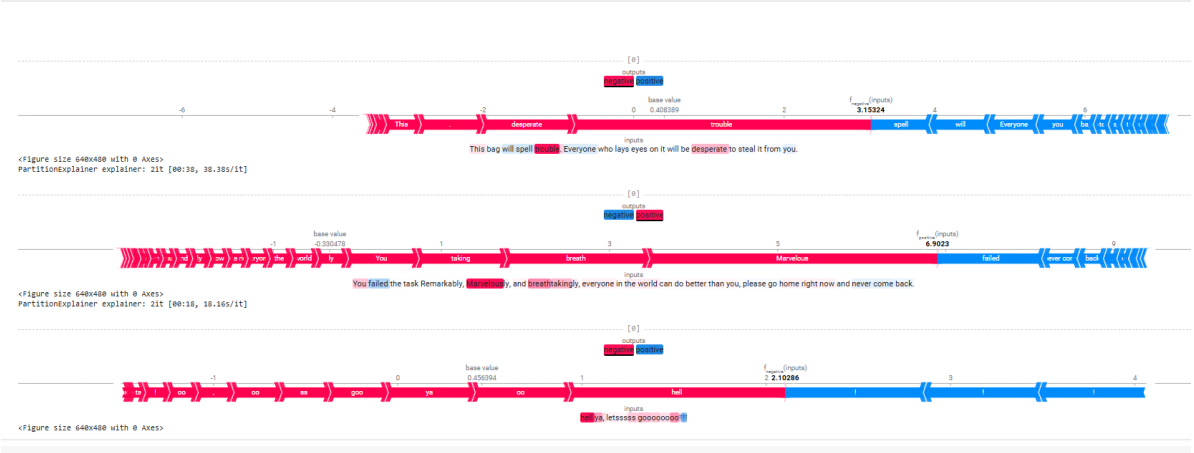


- Attack 2 (Negative)
    - ‘You failed the task Remarkably, Marvelously, and breathtakingly, everyone in the world can do better than you, please go home right now and never come back.’



- Attack 3 (Positive)
    - ‘hell ya, letsssss goooooooo!!!!’





SHAP has the similar results.

When I use indirect praise to imply the desirability of the bag in attack 1, the model doesn’t fully understand the meaning behind the sentence, so it expectedly failed the task. Only LLM can prevent such attacks since they are trained on vast amounts of diverse text data, enabling them to better grasp

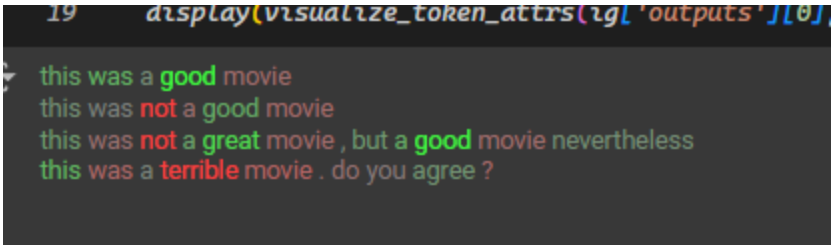
nuances, context, and implied meanings in language. following the next attack, we use a lot of highly positive adv. to describe a negative adj., which the model didn't notice the correct meaning. To address the problem, I think the model should pay more attention and be more sensitive on adjacent words, because some words can make different meanings when they are together. The last attack is a sentence doesn't show up in formal cases, but more likely in a chat box or social media, which should be very discrepant with the original training dataset. We'll need to finetune the model with more unformal sentences in order to handle such test cases if we want. Besides, we can remove the unseen words to get a fair judgment since the model should admit that it has no capability to judge the word, receiving a fair final judgement.

5. **Describe problems you meet and how you solve them. (20%)**

   At first, I didn't really understand the usage of exBert until watching the demo video for a few times and spend a lot of time to get some feasible results to fit in the report, and figuring out the concepts they are representing also requires some efforts. I met some problem running the colab code for the first week due the module version problem, not until someone pointed out the right version of the module 'transformers' on Microsoft Teams did I successfully run LIME and SHAP; and those following steps ran quite smoothly. The last obstacle I finally conquered is forming the appropriate sentences for the attacks. I tried some sentences that aren't confusing enough for the model such as 'Today is not a good day, because I didn't see any beautiful girls', while I finally find some sentences dramatic enough after trial-and-error.

6. **Implement other explanation techniques (Bonus)**

   I tried to rerun other people's GitHub repo to test the explanation of integrated gradients and found out that the results were quite enlightening, but not as exquisite as LIME and SHAP. The integrated gradients method effectively highlighted the importance of the words in the context of sentiment analysis, as seen in the example sentences provided. Words like "good," "great," and "terrible" were appropriately emphasized, which aligns with the expected sentiment conveyed by the sentences, which is quite interesting.


19 display(visualize_token_attrs(ig['outputs'][0]
this was a good movie
this was not a good movie
this was not a great movie , but a good movie nevertheless
this was a terrible movie . do you agree ?