BERT Explanability

Some simple exploration on explainability of BERT models

```
import torch

# Check if CUDA is available
device = torch.device("cuda" if torch.cuda.is_available() else "cpu")
print(f"Using device: {device}")
```

```
Using device: cuda
```

```
from transformers import AutoTokenizer,
        AutoModelForSequenceClassification

# Load model and tokenizer
tokenizer = AutoTokenizer.from_pretrained("bert-base-uncased")
model = AutoModelForSequenceClassification.from_pretrained("keanteng/bert-
        base-raw-climate-sentiment-wqf7007").to(device)
model.eval()
```

```
/usr/local/lib/python3.11/dist-
packages/huggingface_hub/utils/_auth.py:94: UserWarning:
The secret `HF_TOKEN` does not exist in your Colab secrets.
To authenticate with the Hugging Face Hub, create a token in your
settings tab (https://huggingface.co/settings/tokens), set it as
secret in your Google Colab and restart your session.
You will be able to reuse this secret in all of your notebooks.
Please note that authentication is recommended but still optional to
access public models or datasets.
  warnings.warn(
```

{"model_id":"5f3b935532d14c6d98c231287929ea7c","version_major":2,"version_minor":0}

{"model_id":"532bb6f1759542e1a74e4f184cd5ab64","version_major":2,"version_minor":0}

{"model_id":"eb7cb3f02a76497398ed27d42b2050ee","version_major":2,"version_minor":0}

{"model_id":"53a43613822943ccbd40179c5b380d79","version_major":2,"version_minor":0}

{"model_id":"c3f6ce7785984a0eac42b38ff3f5d802","version_major":2,"version_minor":0}

{"model_id":"cb3398eb1f6a491882f4cc2878648616","version_major":2,"version_minor":0}

```
BertForSequenceClassification(
  (bert): BertModel(
    (embeddings): BertEmbeddings(
      (word_embeddings): Embedding(30522, 768, padding_idx=0)
      (position_embeddings): Embedding(512, 768)
      (token_type_embeddings): Embedding(2, 768)
      (LayerNorm): LayerNorm((768,), eps=1e-12,
elementwise_affine=True)
      (dropout): Dropout(p=0.1, inplace=False)
    )
    (encoder): BertEncoder(
      (layer): ModuleList(
        (0-11): 12 x BertLayer(
          (attention): BertAttention(
            (self): BertSdpaSelfAttention(
              (query): Linear(in_features=768, out_features=768,
bias=True)
              (key): Linear(in_features=768, out_features=768,
bias=True)
              (value): Linear(in_features=768, out_features=768,
bias=True)
              (dropout): Dropout(p=0.1, inplace=False)
            )
            (output): BertSelfOutput(
              (dense): Linear(in_features=768, out_features=768,
bias=True)
              (LayerNorm): LayerNorm((768,), eps=1e-12,
elementwise_affine=True)
              (dropout): Dropout(p=0.1, inplace=False)
            )
          )
          (intermediate): BertIntermediate(
            (dense): Linear(in_features=768, out_features=3072,
bias=True)
            (intermediate_act_fn): GELUActivation()
          )
          (output): BertOutput(
            (dense): Linear(in_features=3072, out_features=768,
bias=True)
            (LayerNorm): LayerNorm((768,), eps=1e-12,
elementwise_affine=True)
            (dropout): Dropout(p=0.1, inplace=False)
          )
```

```
          )
        )
      )
      (pooler): BertPooler(
        (dense): Linear(in_features=768, out_features=768, bias=True)
        (activation): Tanh()
      )
    )
    (dropout): Dropout(p=0.1, inplace=False)
    (classifier): Linear(in_features=768, out_features=4, bias=True)
)
```

Using Lime

```
!pip install numpy --quiet
!pip install lime --quiet
!pip install captum --quiet
```

```python
from captum.attr import IntegratedGradients, LayerIntegratedGradients,
        Lime
from captum.attr import visualization
import numpy as np


class BERTExplainer:
    def __init__(self, model, tokenizer, device):
        self.model = model
        self.tokenizer = tokenizer
        self.device = device
        self.class_names = ['anti', 'neutral', 'pro', 'news']
        self.embedding_layer = model.bert.embeddings if hasattr(model,
        'bert') else model.embeddings


    def _tokenize_and_move(self, text):
        """Tokenize text and move to device"""
        inputs = self.tokenizer(text, return_tensors="pt",
        padding=True, truncation=True)
        return {key: val.to(self.device) for key, val in
        inputs.items()}


    def _get_prediction(self, inputs):
        """Get model prediction"""
        with torch.no_grad():
            outputs = self.model(**inputs)
            probs = torch.nn.functional.softmax(outputs.logits,
        dim=-1)
```

```
            return probs

    def _to_numpy(self, tensor):
        """Convert tensor to numpy, handling device transfer"""
        return tensor.detach().cpu().numpy()

    def predict_proba_for_lime(self, texts):
        """Prediction function for LIME"""
        inputs = self._tokenize_and_move(texts)
        probs = self._get_prediction(inputs)
        return self._to_numpy(probs)

    def get_lime_explanation(self, text, num_features=10):
        """Get LIME explanation"""
        from lime.lime_text import LimeTextExplainer

        explainer = LimeTextExplainer(class_names=self.class_names)
        exp = explainer.explain_instance(text,
        self.predict_proba_for_lime, num_features=num_features)
        return exp

    def forward_for_captum(self, input_ids):
        """Forward function for Captum LIME"""
        input_ids = input_ids.long()
        attention_mask = torch.ones_like(input_ids)

        with torch.no_grad():
            outputs = self.model(input_ids=input_ids,
        attention_mask=attention_mask)
            return torch.nn.functional.softmax(outputs.logits, dim=-1)

    def get_integrated_gradients(self, text, target_class=None,
        n_steps=50):
        """Get Integrated Gradients attributions"""
        inputs = self._tokenize_and_move(text)
        input_ids = inputs['input_ids']
        attention_mask = inputs['attention_mask']

        # Get embeddings and prediction
        with torch.no_grad():
            inputs_embeds = self.embedding_layer(input_ids)
            pred = self._get_prediction(inputs)
            pred_class = torch.argmax(pred, dim=1).item()
```

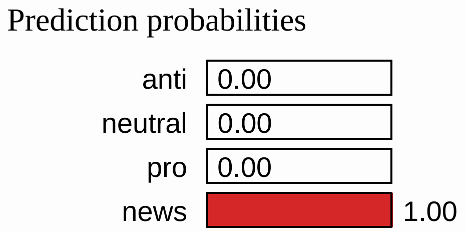
```
        target_class = target_class or pred_class

        # Prediction function for IG
        def predict_fn(inputs_embeds, attention_mask):
            outputs = self.model(inputs_embeds=inputs_embeds,
        attention_mask=attention_mask)
            return torch.nn.functional.softmax(outputs.logits, dim=-1)

        # Initialize and compute attributions
        lig = LayerIntegratedGradients(predict_fn,
        self.embedding_layer)
        baseline_embeds = torch.zeros_like(inputs_embeds)

        attributions = lig.attribute(
            inputs=inputs_embeds,
            baselines=baseline_embeds,
            target=target_class,
            additional_forward_args=(attention_mask,),
            n_steps=n_steps
        )

        return {
            'attributions':
        self._to_numpy(attributions.sum(dim=-1).squeeze(0)),
            'tokens':
        self.tokenizer.convert_ids_to_tokens(input_ids[0]),
            'pred_class': pred_class,
            'pred_probs': self._to_numpy(pred[0]),
            'input_ids': input_ids
        }

    def get_captum_lime(self, text, target_class=None, n_samples=100):
        """Get Captum LIME attributions"""
        inputs = self._tokenize_and_move(text)
        input_ids = inputs['input_ids']

        # Get prediction
        pred = self.forward_for_captum(input_ids)
        pred_class = torch.argmax(pred, dim=1).item()
        target_class = target_class or pred_class

        # Initialize LIME
        lime = Lime(self.forward_for_captum)

        attributions = lime.attribute(
```

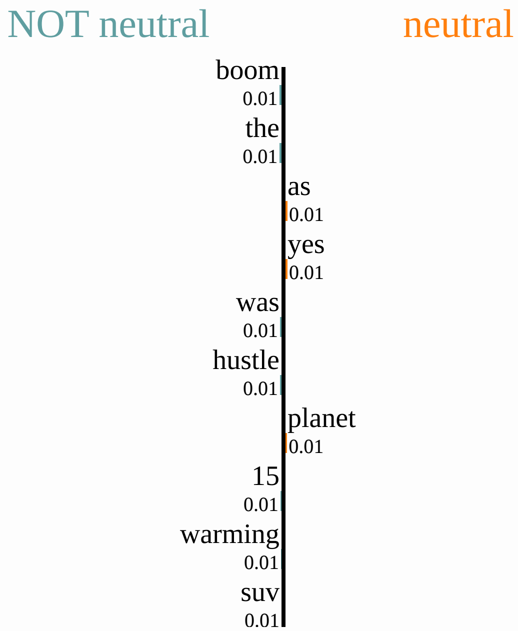
```
            input_ids,
            target=target_class,
            n_samples=n_samples,
            perturbations_per_eval=10
        )

        return {
            'attributions': self._to_numpy(attributions.squeeze(0)),
            'tokens':
        self.tokenizer.convert_ids_to_tokens(input_ids[0]),
            'pred_class': pred_class,
            'pred_probs': self._to_numpy(pred[0])
        }

    # Add this method to the BERTExplainer class, after the existing
        methods
    def group_subword_attributions(self, tokens, attributions):
        """Group subword tokens back into words"""
        grouped_tokens = []
        grouped_attrs = []
        current_word = ""
        current_attr = 0

        for token, attr in zip(tokens, attributions):
            if token.startswith('##'):
                current_word += token[2:]  # Remove ##
                current_attr += attr
            else:
                if current_word:  # Save previous word
                    grouped_tokens.append(current_word)
                    grouped_attrs.append(current_attr)
                current_word = token
                current_attr = attr

        # Don't forget the last word
        if current_word:
            grouped_tokens.append(current_word)
            grouped_attrs.append(current_attr)

        return grouped_tokens, grouped_attrs

    def visualize_attributions(self, result_dict, method_name="",
        group_subwords=True):
        """Visualize attribution results"""
```

```
tokens = result_dict['tokens']
attr_scores = result_dict['attributions']
pred_class = result_dict['pred_class']
pred_probs = result_dict['pred_probs']

print(f"\n{method_name} Results:")
print(f"Predicted class: {self.class_names[pred_class]}
(confidence: {pred_probs[pred_class]:.3f})")
print(f"All probabilities: {[f'{self.class_names[i]}:
{pred_probs[i]:.3f}' for i in range(len(self.class_names))]}")

if group_subwords:
    # Group subword tokens
    grouped_tokens, grouped_attrs =
self.group_subword_attributions(tokens, attr_scores)

    print("\nWord-level attributions (grouped subwords):")
    for token, score in zip(grouped_tokens, grouped_attrs):
        if token not in ['[CLS]', '[SEP]', '[PAD]']:
            print(f"{token:20} {score:8.4f}")

    print("\nOriginal token-level attributions:")
    for token, score in zip(tokens, attr_scores):
        if token not in ['[CLS]', '[SEP]', '[PAD]']:
            print(f"{token:15} {score:8.4f}")

    # Create visualization data for grouped tokens (word-
level)
    vis_data_grouped = visualization.VisualizationDataRecord(
        np.array(grouped_attrs),
        pred_probs[pred_class],
        pred_class,
        self.class_names[pred_class],
        self.class_names[pred_class],
        np.array(grouped_attrs).sum(),
        grouped_tokens,
        1
    )

    # Create visualization data for original tokens (token-
level)
    vis_data_original = visualization.VisualizationDataRecord(
        attr_scores,
        pred_probs[pred_class],
        pred_class,
        self.class_names[pred_class],
```

```
                self.class_names[pred_class],
                attr_scores.sum(),
                tokens,
                1
            )

            return vis_data_grouped, vis_data_original, tokens,
        attr_scores  # Return both versions
        else:
            print("\nToken attributions:")
            for token, score in zip(tokens, attr_scores):
                if token not in ['[CLS]', '[SEP]', '[PAD]']:
                    print(f"{token:15} {score:8.4f}")

            # Create visualization data for original tokens
            vis_data_original = visualization.VisualizationDataRecord(
                attr_scores,
                pred_probs[pred_class],
                pred_class,
                self.class_names[pred_class],
                self.class_names[pred_class],
                attr_scores.sum(),
                tokens,
                1
            )

            return vis_data_original, None, tokens, attr_scores

# Initialize explainer
explainer = BERTExplainer(model, tokenizer, device)


# Example usage
text = "@tiniebeany climate change is an interesting hustle as it was
        global warming but the planet stopped warming for 15 yes while
        the suv boom"


print("="*60)
print("LIME Explanation (Original):")
lime_exp = explainer.get_lime_explanation(text)
lime_exp.show_in_notebook(text=True)
```

```
============================================================
LIME Explanation (Original):
```





Text with highlighted words

@tiniebeany climate change is an interesting hustle as it was global warming but the planet stopped warming for 15 yes while the suv boom

```
print("="*60)
print("Integrated Gradients:")
ig_result = explainer.get_integrated_gradients(text)
vis_data_ig_grouped, vis_data_ig_original, original_tokens,
        original_attrs = explainer.visualize_attributions(ig_result,
        "Integrated Gradients")
```

```
============================================================
Integrated Gradients:


Integrated Gradients Results:
Predicted class: news (confidence: 0.997)
All probabilities: ['anti: 0.001', 'neutral: 0.002', 'pro: 0.001',
'news: 0.997']


Word-level attributions (grouped subwords):
@                       0.0324
tiniebeany             -0.0725
climate                 0.1276
change                  0.1178
is                      0.0962
```

```
an                     0.0652
interesting           -0.0436
hustle                 0.0824
as                     0.0384
it                     0.0121
was                    0.0339
global                 0.1114
warming                0.0282
but                    0.0176
the                    0.0460
planet                 0.0578
stopped                0.0983
warming                0.0490
for                    0.0219
15                    -0.0294
yes                   -0.0209
while                 -0.0199
the                    0.0373
suv                    0.0574
boom                   0.1621
```

```
Original token-level attributions:
@                 0.0324
tin              -0.0626
##ie             -0.0408
##be              0.0017
##any             0.0292
climate           0.1276
change            0.1178
is                0.0962
an                0.0652
interesting      -0.0436
hu                0.0435
##stle            0.0388
as                0.0384
it                0.0121
was               0.0339
global            0.1114
warming           0.0282
but               0.0176
the               0.0460
planet            0.0578
stopped           0.0983
```

```
warming             0.0490
for                 0.0219
15                 -0.0294
yes                -0.0209
while              -0.0199
the                 0.0373
suv                 0.0574
boom                0.1621
```

```
print("="*60)
print("Captum LIME:")
try:
    captum_lime_result = explainer.get_captum_lime(text)
    vis_data_lime_grouped, vis_data_lime_original, _, _ =
        explainer.visualize_attributions(captum_lime_result, "Captum
        LIME")
except Exception as e:
    print(f"Captum LIME failed: {e}")
    vis_data_lime_grouped = None
    vis_data_lime_original = None
```

```
============================================================
Captum LIME:

Captum LIME Results:
Predicted class: news (confidence: 0.997)
All probabilities: ['anti: 0.001', 'neutral: 0.002', 'pro: 0.001',
'news: 0.997']

Word-level attributions (grouped subwords):
@                       0.0334
tiniebeany              0.0772
climate                 0.0242
change                  0.0890
is                      0.1093
an                      0.0000
interesting             0.0000
hustle                  0.2331
as                      0.0000
it                      0.0000
was                     0.0000
global                  0.0706
warming                 0.1355
but                    -0.0787
```

```
the                    0.0000
planet                 0.1999
stopped                0.1743
warming                0.1800
for                    0.0000
15                     0.1582
yes                    0.0000
while                  0.0000
the                    0.0093
suv                    0.0847
boom                   0.0931

Original token-level attributions:
@                 0.0334
tin               0.0000
##ie              0.0000
##be              0.0342
##any             0.0429
climate           0.0242
change            0.0890
is                0.1093
an                0.0000
interesting       0.0000
hu                0.1079
##stle            0.1252
as                0.0000
it                0.0000
was               0.0000
global            0.0706
warming           0.1355
but              -0.0787
the               0.0000
planet            0.1999
stopped           0.1743
warming           0.1800
for               0.0000
15                0.1582
yes               0.0000
while             0.0000
the               0.0093
suv               0.0847
boom              0.0931
```

```python
# HTML visualization with both grouped words and original tokens
try:
    from IPython.display import HTML, display

    print("\n" + "="*60)
    print("HTML Visualization (Word-level - Grouped Subwords):")
    html_grouped = visualization.visualize_text([vis_data_ig_grouped])
    #display(HTML(html_grouped.data))

    print("\nHTML Visualization (Token-level - Original Tokens):")
    html_original =
        visualization.visualize_text([vis_data_ig_original])
    #display(HTML(html_original.data))

    if vis_data_lime_grouped is not None:
        print("\nCaptum LIME HTML Visualization (Word-level - Grouped
        Subwords):")
        html_lime_grouped =
        visualization.visualize_text([vis_data_lime_grouped])
        #display(HTML(html_lime_grouped.data))

        print("\nCaptum LIME HTML Visualization (Token-level -
        Original Tokens):")
        html_lime_original =
        visualization.visualize_text([vis_data_lime_original])
        #display(HTML(html_lime_original.data))

except ImportError:
    print("IPython not available for HTML visualization")
```

```
============================================================
HTML Visualization (Word-level - Grouped Subwords):
```

Legend: Negative Neutral Positive

True Label	Predicted Label	Attribution Label	Attribution Score	Word Importance
news	**3 (1.00)**	**news**	**-0.14**	[CLS] @ tiniebeany climate change is an interesting hustle as it was global warming but the planet stopped warming for 15 yes while the suv boom [SEP]

```
HTML Visualization (Token-level - Original Tokens):
```

Legend: Negative Neutral Positive

True Label	Predicted Label	Attribution Label	Attribution Score	Word Importance
news	**3 (1.00)**	**news**	**-0.14**	[CLS] @ tin ##ie ##be ##any climate change is an interesting hu ##stle as it was global warming but the planet stopped warming for 15 yes while the suv boom [SEP]

```
Captum LIME HTML Visualization (Word-level - Grouped Subwords):
```

Legend: Negative Neutral Positive

True Label	Predicted Label	Attribution Label	Attribution Score	Word Importance
news	**3 (1.00)**	**news**	**1.59**	[CLS] @ tiniebeany climate change is an interesting hustle as it was global warming but the planet stopped warming for 15 yes while the suv boom [SEP]

```
Captum LIME HTML Visualization (Token-level - Original Tokens):
```

Legend: Negative Neutral Positive

True Label	Predicted Label	Attribution Label	Attribution Score	Word Importance
news	**3 (1.00)**	**news**	**1.59**	[CLS] @ tin ##ie ##be ##any climate change is an interesting hu ##stle as it was global warming but the

planet stopped warming for
15 yes while the suv boom
[SEP]

Why the output is different?

The differences between the HTML visualization (Integrated Gradients) and Captum LIME HTML visualization occur due to fundamental differences in how these two explainability methods work:

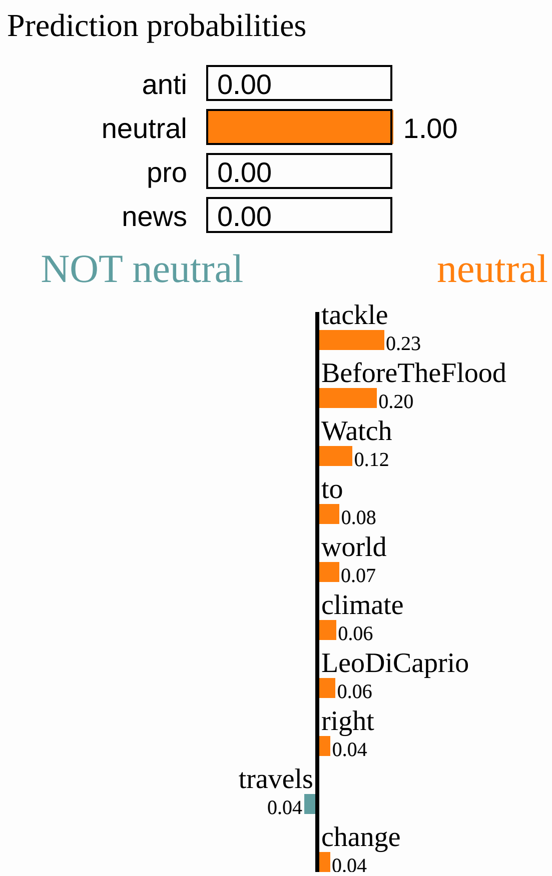
Key Differences:

1. Methodology
   - Integrated Gradients: Computes gradients by integrating along a straight path from a baseline (zeros) to the actual input embeddings. It captures how much each token contributes to the prediction based on the model's internal gradients.
   - Captum LIME: Uses a perturbation-based approach, creating many variations of the input by masking/removing tokens and observing how predictions change.
2. Attribution Calculation
   - Integrated Gradients:
     - Works at the embedding level
     - Provides smooth, continuous attributions
     - Captures the model's sensitivity to each token
   - Captum LIME:
     - Works by perturbing the input tokens
     - Fits a local linear model around the prediction
     - May have more discrete/binary-like attributions
3. Baseline Differences
   - Integrated Gradients: Uses zero embeddings as baseline
   - Captum LIME: Uses token removal/masking as perturbation method

Another Example

```
# Example usage
text = "#BeforeTheFlood Watch #BeforeTheFlood right here, as
        @LeoDiCaprio travels the world to tackle climate change...
        https://t.co/HCIZrPUhLF"


print("="*60)
print("LIME Explanation (Original):")
lime_exp = explainer.get_lime_explanation(text)
lime_exp.show_in_notebook(text=True)
```

```
============================================================
LIME Explanation (Original):
```



Text with highlighted words

#BeforeTheFlood Watch #BeforeTheFlood right here, as @LeoDiCaprio travels the world to tackle climate change... https://t.co/HCIZrPUhLF

```
print("="*60)
print("Integrated Gradients:")
ig_result = explainer.get_integrated_gradients(text)
vis_data_ig_grouped, vis_data_ig_original, original_tokens,
        original_attrs = explainer.visualize_attributions(ig_result,
        "Integrated Gradients")
```

```
============================================================
Integrated Gradients:


Integrated Gradients Results:
Predicted class: neutral (confidence: 0.999)
All probabilities: ['anti: 0.000', 'neutral: 0.999', 'pro: 0.001',
'news: 0.000']


Word-level attributions (grouped subwords):
#                      -0.0008
beforetheflood          0.0153
```

watch	0.0317
#	0.0004
beforetheflood	0.0110
right	0.0511
here	0.0380
,	-0.0267
as	-0.0215
@	0.0511
leodicaprio	0.0686
travels	-0.0086
the	0.0792
world	0.0143
to	0.1569
tackle	0.3170
climate	0.1846
change	0.2285
.	0.0018
.	-0.0297
.	0.0031
https	0.0174
:	-0.0001
/	-0.0033
/	-0.0043
t	0.0058
.	-0.0143
co	0.0045
/	-0.0142
hcizrpuhlf	-0.0148

Original token-level attributions:

#	-0.0008
before	-0.0245
##the	0.0130
##fl	0.0198
##ood	0.0070
watch	0.0317
#	0.0004
before	-0.0519
##the	0.0259
##fl	0.0249
##ood	0.0120
right	0.0511
here	0.0380

```
,                -0.0267
as               -0.0215
@                 0.0511
leo               0.0065
##dic             0.0081
##ap              0.0042
##rio             0.0497
travels          -0.0086
the               0.0792
world             0.0143
to                0.1569
tackle            0.3170
climate           0.1846
change            0.2285
.                 0.0018
.                -0.0297
.                 0.0031
https             0.0174
:                -0.0001
/                -0.0033
/                -0.0043
t                 0.0058
.                -0.0143
co                0.0045
/                -0.0142
hc                0.0300
##iz             -0.0198
##rp             -0.0129
##uh             -0.0168
##lf              0.0047
```

```python
print("="*60)
print("Captum LIME:")
try:
    captum_lime_result = explainer.get_captum_lime(text)
    vis_data_lime_grouped, vis_data_lime_original, _, _ =
        explainer.visualize_attributions(captum_lime_result, "Captum
        LIME")
except Exception as e:
    print(f"Captum LIME failed: {e}")
    vis_data_lime_grouped = None
    vis_data_lime_original = None
```

```
============================================================
Captum LIME:

Captum LIME Results:
Predicted class: neutral (confidence: 0.999)
All probabilities: ['anti: 0.000', 'neutral: 0.999', 'pro: 0.001',
'news: 0.000']

Word-level attributions (grouped subwords):
#                       0.0636
beforetheflood          0.0210
watch                   0.1490
#                       0.0000
beforetheflood          0.0395
right                   0.0000
here                    0.0657
,                       0.0000
as                      0.0000
@                       0.0000
leodicaprio             0.0488
travels                 0.0000
the                     0.0000
world                   0.0000
to                      0.0415
tackle                  0.1711
climate                 0.0088
change                  0.0456
.                       0.0000
.                       0.0000
.                       0.0000
https                   0.0000
:                       0.0017
/                       0.0415
/                       0.0000
t                       0.0623
.                       0.0000
co                      0.0000
/                      -0.0028
hcizrpuhlf              0.0500

Original token-level attributions:
#                  0.0636
before            -0.0095
```

##the	0.0000
##fl	0.0000
##ood	0.0305
watch	0.1490
#	0.0000
before	0.0060
##the	0.0336
##fl	0.0000
##ood	0.0000
right	0.0000
here	0.0657
,	0.0000
as	0.0000
@	0.0000
leo	-0.0209
##dic	0.0000
##ap	0.0000
##rio	0.0698
travels	0.0000
the	0.0000
world	0.0000
to	0.0415
tackle	0.1711
climate	0.0088
change	0.0456
.	0.0000
.	0.0000
.	0.0000
https	0.0000
:	0.0017
/	0.0415
/	0.0000
t	0.0623
.	0.0000
co	0.0000
/	-0.0028
hc	0.0000
##iz	0.0500
##rp	0.0000
##uh	0.0000
##lf	0.0000

```python
# HTML visualization with both grouped words and original tokens
try:
    from IPython.display import HTML, display

    print("\n" + "="*60)
    print("HTML Visualization (Word-level - Grouped Subwords):")
    html_grouped = visualization.visualize_text([vis_data_ig_grouped])
    #display(HTML(html_grouped.data))

    print("\nHTML Visualization (Token-level - Original Tokens):")
    html_original =
        visualization.visualize_text([vis_data_ig_original])
    #display(HTML(html_original.data))

    if vis_data_lime_grouped is not None:
        print("\nCaptum LIME HTML Visualization (Word-level - Grouped
        Subwords):")
        html_lime_grouped =
        visualization.visualize_text([vis_data_lime_grouped])
        #display(HTML(html_lime_grouped.data))

        print("\nCaptum LIME HTML Visualization (Token-level -
        Original Tokens):")
        html_lime_original =
        visualization.visualize_text([vis_data_lime_original])
        #display(HTML(html_lime_original.data))

except ImportError:
    print("IPython not available for HTML visualization")
```

```
============================================================
HTML Visualization (Word-level - Grouped Subwords):
```

Legend: Negative Neutral Positive

True Label	Predicted Label	Attribution Label	Attribution Score	Word Importance
neutral	**1 (1.00)**	**neutral**	**1.09**	[CLS] # beforetheflood watch # beforetheflood right here , as @ leodicaprio travels the world to tackle climate change . . . https : / / t . co / hcizrpuhlf [SEP]

HTML Visualization (Token-level - Original Tokens):

Legend: Negative Neutral Positive

True Label	Predicted Label	Attribution Label	Attribution Score	Word Importance
neutral	**1 (1.00)**	**neutral**	**1.09**	[CLS] # before ##the ##fl ##ood watch # before ##the ##fl ##ood right here , as @ leo ##dic ##ap ##rio travels the world to tackle climate change . . . https : / / t . co / hc ##iz ##rp ##uh ##lf [SEP]

Captum LIME HTML Visualization (Word-level - Grouped Subwords):

Legend: Negative Neutral Positive

True Label	Predicted Label	Attribution Label	Attribution Score	Word Importance
neutral	**1 (1.00)**	**neutral**	**0.60**	[CLS] # beforetheflood watch # beforetheflood right here , as @ leodicaprio travels the world to tackle climate change . . . https : / / t . co / hcizrpuhlf [SEP]

Captum LIME HTML Visualization (Token-level - Original Tokens):

Legend: Negative Neutral Positive

True Label	Predicted Label	Attribution Label	Attribution Score	Word Importance
neutral	**1 (1.00)**	**neutral**	**0.60**	[CLS] # before ##the ##fl ##ood watch # before

	##the ##fl ##ood right here , as @ leo ##dic ##ap ##rio travels the world to tackle climate change . . . https : / / t . co / hc ##iz ##rp ##uh ##lf [SEP]