```
from google.colab import drive
drive.mount('/content/drive')
```

```
Mounted at /content/drive
```

```
!pip install transformers
!pip install torch
```

```
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 24.6/24.6 MB 35.4 MB/s eta 0:00:00
Downloading nvidia_cuda_runtime_cu12-12.4.127-py3-none-manylinux2014_x86_64.whl (883 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 883.7/883.7 kB 40.0 MB/s eta 0:00:00
Downloading nvidia_cudnn_cu12-9.1.0.70-py3-none-manylinux2014_x86_64.whl (664.8 MB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 664.8/664.8 MB 2.2 MB/s eta 0:00:00
Downloading nvidia_cufft_cu12-11.2.1.3-py3-none-manylinux2014_x86_64.whl (211.5 MB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 211.5/211.5 MB 4.4 MB/s eta 0:00:00
Downloading nvidia_curand_cu12-10.3.5.147-py3-none-manylinux2014_x86_64.whl (56.3 MB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 56.3/56.3 MB 12.4 MB/s eta 0:00:00
Downloading nvidia_cusolver_cu12-11.6.1.9-py3-none-manylinux2014_x86_64.whl (127.9 MB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 127.9/127.9 MB 7.5 MB/s eta 0:00:00
Downloading nvidia_cusparse_cu12-12.3.1.170-py3-none-manylinux2014_x86_64.whl (207.5 MB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 207.5/207.5 MB 5.9 MB/s eta 0:00:00
Downloading nvidia_nvjitlink_cu12-12.4.127-py3-none-manylinux2014_x86_64.whl (21.1 MB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 21.1/21.1 MB 70.3 MB/s eta 0:00:00
Installing collected packages: nvidia-nvjitlink-cu12, nvidia-curand-cu12, nvidia-cufft-cu12, nvidia-cuda-runtime-cu12, nvidia-cuda-nv
  Attempting uninstall: nvidia-nvjitlink-cu12
    Found existing installation: nvidia-nvjitlink-cu12 12.5.82
    Uninstalling nvidia-nvjitlink-cu12-12.5.82:
      Successfully uninstalled nvidia-nvjitlink-cu12-12.5.82
  Attempting uninstall: nvidia-curand-cu12
    Found existing installation: nvidia-curand-cu12 10.3.6.82
    Uninstalling nvidia-curand-cu12-10.3.6.82:
      Successfully uninstalled nvidia-curand-cu12-10.3.6.82
  Attempting uninstall: nvidia-cufft-cu12
    Found existing installation: nvidia-cufft-cu12 11.2.3.61
    Uninstalling nvidia-cufft-cu12-11.2.3.61:
      Successfully uninstalled nvidia-cufft-cu12-11.2.3.61
  Attempting uninstall: nvidia-cuda-runtime-cu12
    Found existing installation: nvidia-cuda-runtime-cu12 12.5.82
    Uninstalling nvidia-cuda-runtime-cu12-12.5.82:
      Successfully uninstalled nvidia-cuda-runtime-cu12-12.5.82
  Attempting uninstall: nvidia-cuda-nvrtc-cu12
    Found existing installation: nvidia-cuda-nvrtc-cu12 12.5.82
    Uninstalling nvidia-cuda-nvrtc-cu12-12.5.82:
      Successfully uninstalled nvidia-cuda-nvrtc-cu12-12.5.82
  Attempting uninstall: nvidia-cuda-cupti-cu12
    Found existing installation: nvidia-cuda-cupti-cu12 12.5.82
    Uninstalling nvidia-cuda-cupti-cu12-12.5.82:
      Successfully uninstalled nvidia-cuda-cupti-cu12-12.5.82
  Attempting uninstall: nvidia-cublas-cu12
    Found existing installation: nvidia-cublas-cu12 12.5.3.2
    Uninstalling nvidia-cublas-cu12-12.5.3.2:
      Successfully uninstalled nvidia-cublas-cu12-12.5.3.2
  Attempting uninstall: nvidia-cusparse-cu12
    Found existing installation: nvidia-cusparse-cu12 12.5.1.3
    Uninstalling nvidia-cusparse-cu12-12.5.1.3:
      Successfully uninstalled nvidia-cusparse-cu12-12.5.1.3
  Attempting uninstall: nvidia-cudnn-cu12
    Found existing installation: nvidia-cudnn-cu12 9.3.0.75
    Uninstalling nvidia-cudnn-cu12-9.3.0.75:
      Successfully uninstalled nvidia-cudnn-cu12-9.3.0.75
  Attempting uninstall: nvidia-cusolver-cu12
    Found existing installation: nvidia-cusolver-cu12 11.6.3.83
    Uninstalling nvidia-cusolver-cu12-11.6.3.83:
      Successfully uninstalled nvidia-cusolver-cu12-11.6.3.83
Successfully installed nvidia-cublas-cu12-12.4.5.8 nvidia-cuda-cupti-cu12-12.4.127 nvidia-cuda-nvrtc-cu12-12.4.127 nvidia-cuda-runtim
```

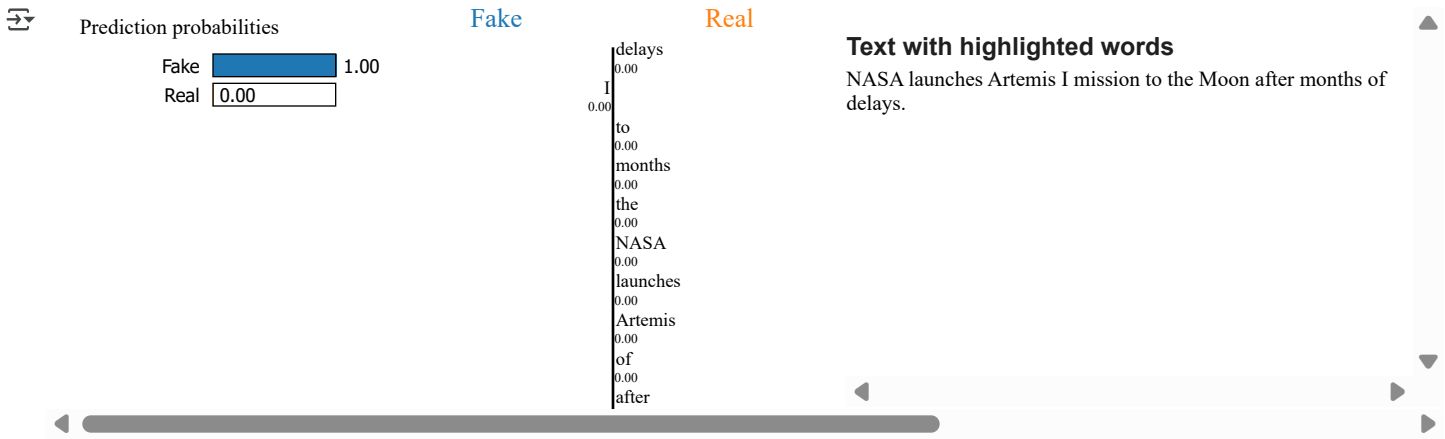
```
from transformers import BertForSequenceClassification, BertTokenizer

model_path = "/content/drive/MyDrive/fake_news_models"

tokenizer = BertTokenizer.from_pretrained(model_path)
model = BertForSequenceClassification.from_pretrained(model_path)
```

```
!pip install lime
```

```
Collecting lime
  Downloading lime-0.2.0.1.tar.gz (275 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 275.7/275.7 kB 6.0 MB/s eta 0:00:00
  Preparing metadata (setup.py) ... done
Requirement already satisfied: matplotlib in /usr/local/lib/python3.11/dist-packages (from lime) (3.10.0)
Requirement already satisfied: numpy in /usr/local/lib/python3.11/dist-packages (from lime) (2.0.2)
Requirement already satisfied: scipy in /usr/local/lib/python3.11/dist-packages (from lime) (1.14.1)
Requirement already satisfied: tqdm in /usr/local/lib/python3.11/dist-packages (from lime) (4.67.1)
Requirement already satisfied: scikit-learn>=0.18 in /usr/local/lib/python3.11/dist-packages (from lime) (1.6.1)
Requirement already satisfied: scikit-image>=0.12 in /usr/local/lib/python3.11/dist-packages (from lime) (0.25.2)
Requirement already satisfied: networkx>=3.0 in /usr/local/lib/python3.11/dist-packages (from scikit-image>=0.12->lime) (3.4.2)
Requirement already satisfied: pillow>=10.1 in /usr/local/lib/python3.11/dist-packages (from scikit-image>=0.12->lime) (11.1.0)
Requirement already satisfied: imageio!=2.35.0,>=2.33 in /usr/local/lib/python3.11/dist-packages (from scikit-image>=0.12->lime) (2.37.0
Requirement already satisfied: tifffile>=2022.8.12 in /usr/local/lib/python3.11/dist-packages (from scikit-image>=0.12->lime) (2025.3.30
Requirement already satisfied: packaging>=21 in /usr/local/lib/python3.11/dist-packages (from scikit-image>=0.12->lime) (24.2)
Requirement already satisfied: lazy-loader>=0.4 in /usr/local/lib/python3.11/dist-packages (from scikit-image>=0.12->lime) (0.4)
Requirement already satisfied: joblib>=1.2.0 in /usr/local/lib/python3.11/dist-packages (from scikit-learn>=0.18->lime) (1.4.2)
Requirement already satisfied: threadpoolctl>=3.1.0 in /usr/local/lib/python3.11/dist-packages (from scikit-learn>=0.18->lime) (3.6.0)
Requirement already satisfied: contourpy>=1.0.1 in /usr/local/lib/python3.11/dist-packages (from matplotlib->lime) (1.3.2)
Requirement already satisfied: cycler>=0.10 in /usr/local/lib/python3.11/dist-packages (from matplotlib->lime) (0.12.1)
Requirement already satisfied: fonttools>=4.22.0 in /usr/local/lib/python3.11/dist-packages (from matplotlib->lime) (4.57.0)
Requirement already satisfied: kiwisolver>=1.3.1 in /usr/local/lib/python3.11/dist-packages (from matplotlib->lime) (1.4.8)
Requirement already satisfied: pyparsing>=2.3.1 in /usr/local/lib/python3.11/dist-packages (from matplotlib->lime) (3.2.3)
Requirement already satisfied: python-dateutil>=2.7 in /usr/local/lib/python3.11/dist-packages (from matplotlib->lime) (2.8.2)
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.11/dist-packages (from python-dateutil>=2.7->matplotlib->lime) (1.17.0
Building wheels for collected packages: lime
  Building wheel for lime (setup.py) ... done
  Created wheel for lime: filename=lime-0.2.0.1-py3-none-any.whl size=283834 sha256=38b42eb18336b132a956356ee8bbb975024dd75e523542acda32
  Stored in directory: /root/.cache/pip/wheels/85/fa/a3/9c2d44c9f3cd77cf4e533b58900b2bf4487f2a17e8ec212a3d
Successfully built lime
Installing collected packages: lime
Successfully installed lime-0.2.0.1
```

```
from lime.lime_text import LimeTextExplainer
import torch
import numpy as np

# ✅ Correct class names (assuming: 0 = Fake, 1 = Real)
explainer = LimeTextExplainer(class_names=['Fake', 'Real'])

# ✅ Updated probability function to work with LIME
def predict_proba(texts):
    # Tokenize input texts for the BERT model
    inputs = tokenizer(texts, return_tensors="pt", padding=True, truncation=True, max_length=512).to(model.device)
    with torch.no_grad():
        outputs = model(**inputs)
        # Apply softmax to get probabilities
        probs = torch.nn.functional.softmax(outputs.logits, dim=1)
    return probs.cpu().numpy()  # Shape: [num_samples, 2] → [Fake, Real]
```

```
# ✅ Sample real news article for testing
text = "NASA launches Artemis I mission to the Moon after months of delays."

# ✅ Run LIME explanation
exp = explainer.explain_instance(text, predict_proba, num_features=10)

# ✅ Visualize explanation in notebook
exp.show_in_notebook()
```

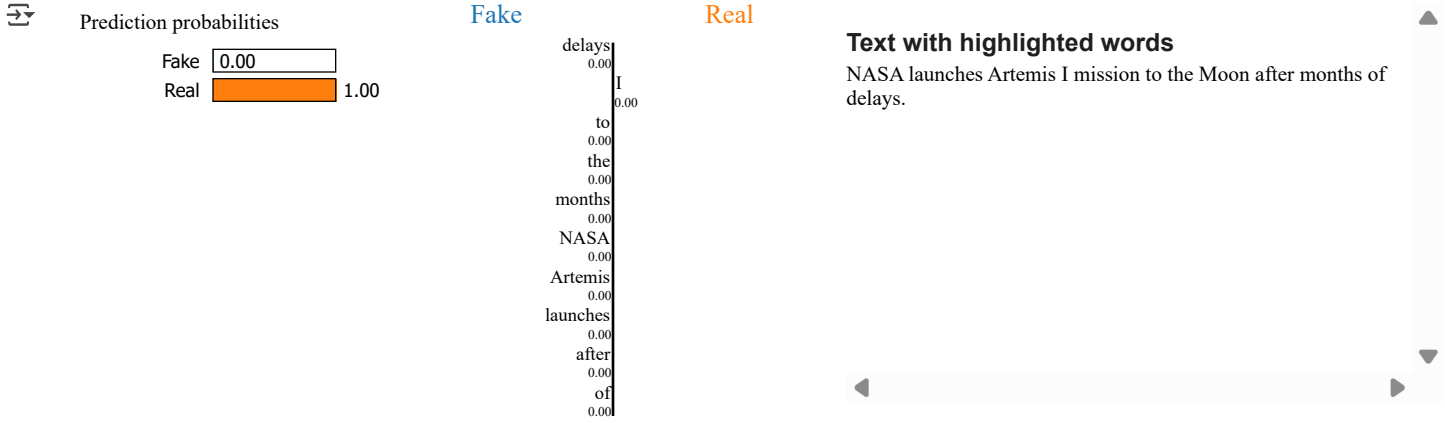



```
def predict_proba(texts):
    inputs = tokenizer(texts, return_tensors="pt", padding=True, truncation=True, max_length=512).to(model.device)
    with torch.no_grad():
        outputs = model(**inputs)
        probs = torch.nn.functional.softmax(outputs.logits, dim=1)
    return probs.cpu().numpy()[:, [1, 0]]  # Flip columns: [Real, Fake] → [Fake, Real]
```

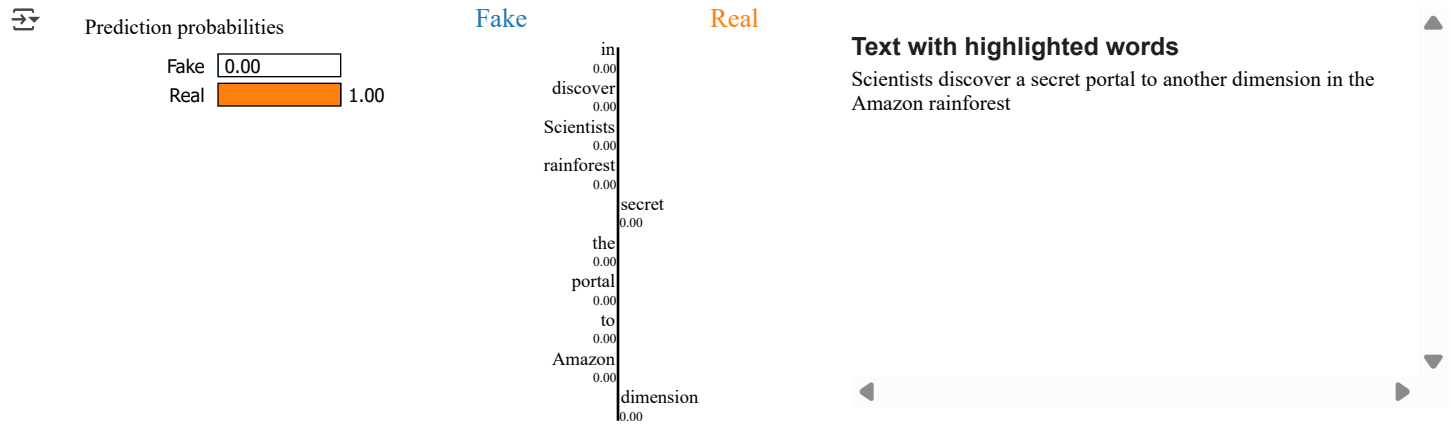
```
probs = predict_proba([text])
predicted_label = np.argmax(probs[0])
print(f"Predicted Label: {'Fake' if predicted_label == 0 else 'Real'}")
print(f"Confidence: {probs[0][predicted_label]:.4f}")
```

```
Predicted Label: Real
Confidence: 1.0000
```

```
# ✅ Sample real news article for testing
text = "NASA launches Artemis I mission to the Moon after months of delays."

# ✅ Run LIME explanation
exp = explainer.explain_instance(text, predict_proba, num_features=10)

# ✅ Visualize explanation in notebook
exp.show_in_notebook()
```

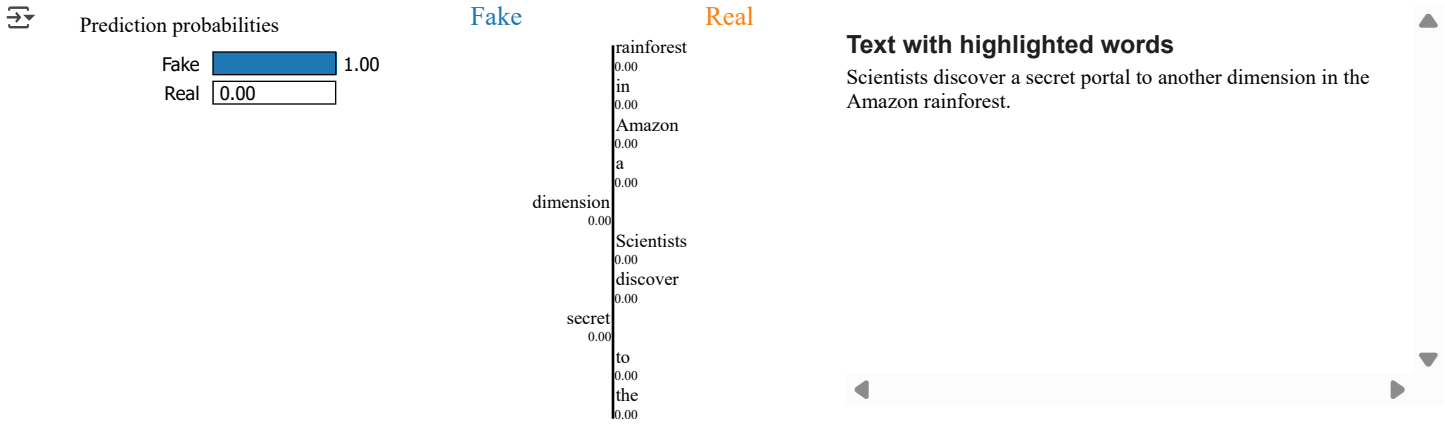



```
# ✅ Sample real news article for testing
text = "Scientists discover a secret portal to another dimension in the Amazon rainforest"

# ✅ Run LIME explanation
exp = explainer.explain_instance(text, predict_proba, num_features=10)

# ✅ Visualize explanation in notebook
exp.show_in_notebook()
```



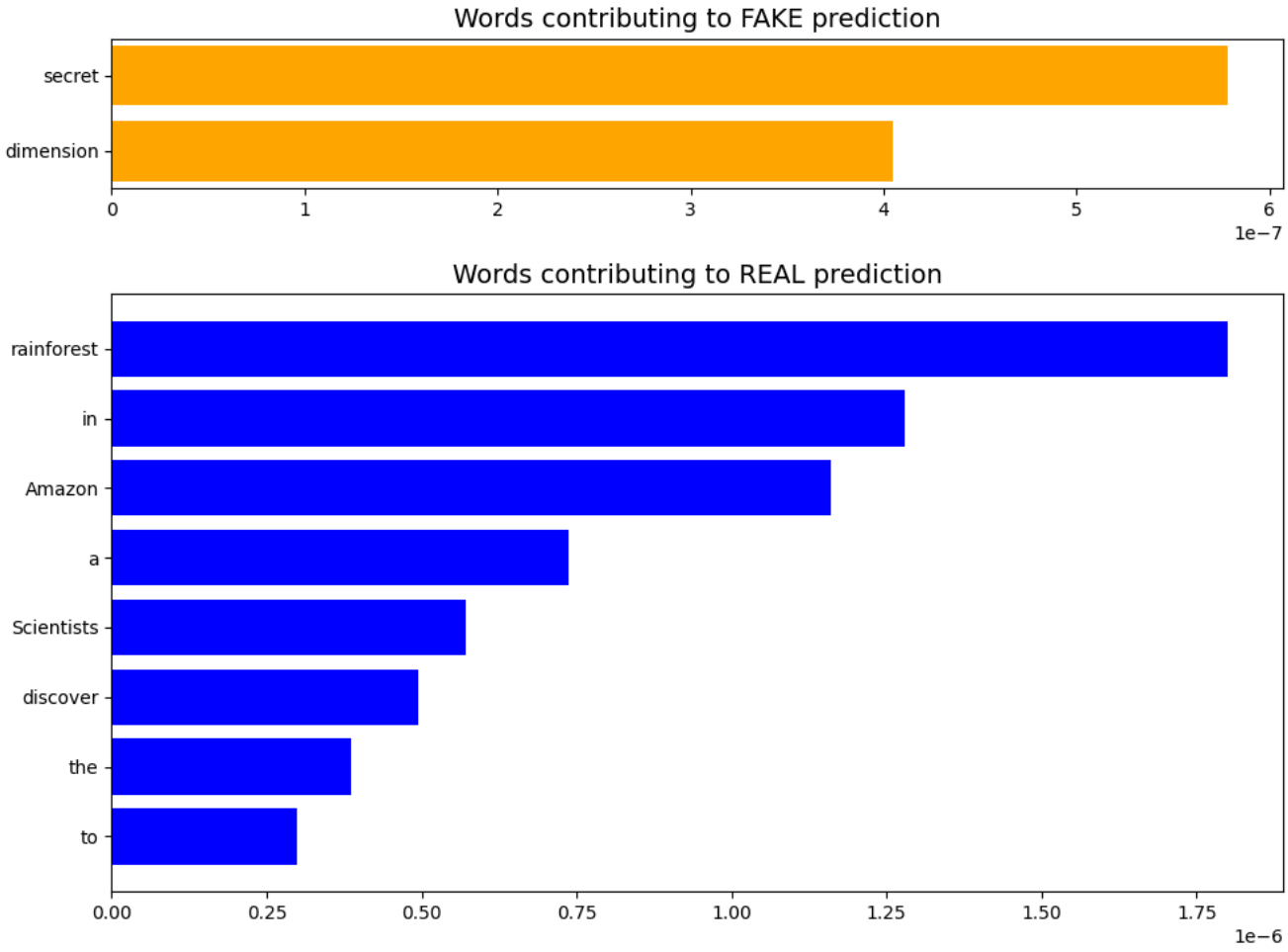

```
from lime import lime_text
from lime.lime_text import LimeTextExplainer
import matplotlib.pyplot as plt

# Define class names
class_names = ['Fake', 'Real']

# Create the explainer with correct class names
explainer = LimeTextExplainer(class_names=class_names)

# Define custom color for the prediction bars
predict_proba_color = {
    0: 'orange',   # Fake class
    1: 'blue'      # Real class
}

# While plotting, make sure the bars use these colors
# (You need to slightly modify the plotting function to apply this.)
```

```
from lime.lime_text import LimeTextExplainer
import numpy as np
import torch

# Correct class names order
class_names = ['Fake', 'Real']

explainer = LimeTextExplainer(class_names=class_names)

def predict_proba(texts):
    inputs = tokenizer(texts, return_tensors="pt", padding=True, truncation=True).to(model.device)
    with torch.no_grad():
        outputs = model(**inputs)
        probs = torch.nn.functional.softmax(outputs.logits, dim=1)
    probs = probs[:, [0, 1]]  # <-- No flip needed if class_names correct
    return probs.cpu().numpy()

# Example usage
text = "Scientists discover a secret portal to another dimension in the Amazon rainforest."
exp = explainer.explain_instance(text, predict_proba, num_features=10)
exp.show_in_notebook()
```




```
from transformers import AutoTokenizer, AutoModelForSequenceClassification
from lime.lime_text import LimeTextExplainer
import torch
import numpy as np
import matplotlib.pyplot as plt

# 1. Load model and tokenizer
from transformers import BertForSequenceClassification, BertTokenizer

model_path = "/content/drive/MyDrive/fake_news_models"

tokenizer = BertTokenizer.from_pretrained(model_path)
model = BertForSequenceClassification.from_pretrained(model_path)

model = model.cpu()

# 2. Define predict function
def predict_proba(texts):
    inputs = tokenizer(texts, return_tensors="pt", padding=True, truncation=True, max_length=512)
    with torch.no_grad():
        outputs = model(**inputs)
        probs = torch.nn.functional.softmax(outputs.logits, dim=1)
    return probs.cpu().numpy()

# 3. Class names
class_names = ['Fake', 'Real']

# 4. Create explainer
explainer = LimeTextExplainer(class_names=class_names)

# 5. Predict and explain
text = "Scientists discover a secret portal to another dimension in the Amazon rainforest."
exp = explainer.explain_instance(text, predict_proba, num_features=10)

# 6. New Custom Split Plot
def split_contribution_plot(exp, class_names):
    # Get the feature weights
    exp_list = exp.as_list()

    fake_words = []
    fake_scores = []
    real_words = []
    real_scores = []

    for word, weight in exp_list:
        if weight < 0:
            fake_words.append(word)
            fake_scores.append(-weight)  # Take positive value for visualization
        else:
            real_words.append(word)
            real_scores.append(weight)

    fig, (ax1, ax2) = plt.subplots(2, 1, figsize=(10, 8), gridspec_kw={'height_ratios': [len(fake_words), len(real_words)]})
    fig.suptitle('Word Contributions to Prediction\n', fontsize=16, fontweight='bold')

    # Plot Fake
    ax1.barh(fake_words, fake_scores, color='orange')
    ax1.set_title('Words contributing to FAKE prediction', fontsize=14)
```

```
    ax1.invert_yaxis()

    # Plot Real
    ax2.barh(real_words, real_scores, color='blue')
    ax2.set_title('Words contributing to REAL prediction', fontsize=14)
    ax2.invert_yaxis()

    plt.tight_layout()
    plt.show()

    # Show colored text in notebook too
    exp.show_in_notebook(text=True)

# 7. Plot Split Graph
split_contribution_plot(exp, class_names)
```



Fake Real

Prediction probabilities

Fake 1.00

Real 0.00

rainforest 0.00

in 0.00

Text with highlighted words

Scientists discover a secret portal to another dimension in the Amazon rainforest.