```
import feedparser
import torch
from transformers import AutoModelForSequenceClassification, AutoTokenizer, pipeline

# Set the device to GPU if available, otherwise CPU
device = torch.device("cuda" if torch.cuda.is_available() else "cpu")

ticker = 'META'
keyword = 'meta'

# Load the pre-trained financial sentiment analysis model and tokenizer
model_name = "ProsusAI/finbert"
model = AutoModelForSequenceClassification.from_pretrained(model_name).to(device)
tokenizer = AutoTokenizer.from_pretrained(model_name)

# Create a pipeline for convenient sentiment analysis using the loaded model and tokenizer
pipe = pipeline("text-classification", model=model, tokenizer=tokenizer, device=device)

# Define the RSS feed URL based on the ticker symbol
rss_url = f'https://finance.yahoo.com/rss/headline?s={ticker}'
feed = feedparser.parse(rss_url)

# Initialize variables to track sentiment scores
total_score = 0
num_articles = 0

# Loop through each news entry in the feed
for i, entry in enumerate(feed.entries):
    # Skip entries that don't contain the keyword in the summary (case-insensitive)
    if keyword.lower() not in entry.summary.lower():
        continue
    print(f'Title: {entry.title}')
    print(f'Link: {entry.link}')
    print(f'Published: {entry.published}')
    print(f'Summary: {entry.summary}')
    # Use the pipeline to analyze sentiment of the entry's summary
    sentiment = pipe(entry.summary)[0]
    print(f'Sentiment: {sentiment["label"]}, Score: {sentiment["score"]}')
    print('-' * 40)
    # Update sentiment score based on the predicted sentiment label (positive/negative)
    if sentiment['label'] == 'positive':
        total_score += sentiment['score']
    elif sentiment['label'] == 'negative':
        total_score -= sentiment['score']
    num_articles += 1

# Calculate the overall sentiment score if there are relevant articles
if num_articles > 0:
    final_score = total_score / num_articles
    print(
        f'Overall Sentiment: {"Positive" if final_score >= 0.15 else "Negative" if
final_score <= -0.15 else "Neutral"}, Score: {final_score}')
else:
    print('No relevant articles found for the specified keyword.')
```