```
from transformers import BertTokenizer, BertForMaskedLM
from transformers import pipeline
from nltk.corpus import stopwords
from nltk.tokenize import word_tokenize
from nltk.stem import WordNetLemmatizer
from gensim.models import Word2Vec
from nltk.sentiment.vader import SentimentIntensityAnalyzer
import nltk
nltk.download('punkt')
nltk.download('stopwords')
nltk.download('wordnet')
nltk.download('vader_lexicon')
# Load pre-trained BERT model and tokenizer
model_name = "bert-base-uncased"
tokenizer = BertTokenizer.from_pretrained(model_name)
model = BertForMaskedLM.from_pretrained(model_name)

# Define the summarization pipeline
summarization_pipeline = pipeline("summarization")
```

```
[nltk_data] Downloading package punkt to /root/nltk_data...
[nltk_data]   Unzipping tokenizers/punkt.zip.
[nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data]   Unzipping corpora/stopwords.zip.
[nltk_data] Downloading package wordnet to /root/nltk_data...
[nltk_data] Downloading package vader_lexicon to /root/nltk_data...
/usr/local/lib/python3.10/dist-packages/huggingface_hub/utils/_token.py:89: 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/sett
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(
```

```
tokenizer_config.json:  100%                                     48.0/48.0  [00:00<00:00,  2.43kB/s]

vocab.txt:  100%                                     232k/232k  [00:00<00:00,  2.99MB/s]

tokenizer.json:  100%                                     466k/466k  [00:00<00:00,  11.5MB/s]
```

```
/usr/local/lib/python3.10/dist-packages/huggingface_hub/file_download.py:1132: FutureWarning: `resume_downl
  warnings.warn(
config.json:  100%                                    570/570  [00:00<00:00,  39.6kB/s]
model.safetensors:  100%                                  440M/440M  [00:05<00:00,  68.3MB/s]
Some weights of the model checkpoint at bert-base-uncased were not used when initializing BertForMaskedLM:
- This IS expected if you are initializing BertForMaskedLM from the checkpoint of a model trained on anothe
- This IS NOT expected if you are initializing BertForMaskedLM from the checkpoint of a model that you expe
No model was supplied, defaulted to sshleifer/distilbart-cnn-12-6 and revision a4f8f3e (https://huggingface
Using a pipeline without specifying a model name and revision in production is not recommended.
config.json:  100%                                    1.80k/1.80k  [00:00<00:00,  49.6kB/s]
pytorch_model.bin:  100%                                  1.22G/1.22G  [00:14<00:00,  87.6MB/s]
tokenizer_config.json:  100%                                26.0/26.0  [00:00<00:00,  1.28kB/s]
vocab.json:  100%                                     899k/899k  [00:00<00:00,  19.9MB/s]
merges.txt:  100%                                     456k/456k  [00:00<00:00,  16.1MB/s]
```

Cricket Commentary Analysis

This project provides a set of Python functions to analyze cricket commentary and identify interesting overs and generate summaries. It utilizes natural language processing techniques, including sentiment analysis and word embeddings, to score and highlight significant moments in the commentary.

Features

- **Sentiment Analysis:** Calculates sentiment scores for each over to identify emotionally charged moments.
- **Word Scoring:** Identifies cricket-related terms and their frequency within overs to score their relevance.
- **Highlight Extraction:** Based on the calculated scores, the project extracts and summarizes the most interesting overs.

- **Summarization:** Uses a pre-trained BART model for generating concise summaries of the highlighted overs.

Prerequisites

To run this code, you need to have the following libraries installed:

- `transformers`
- `nltk`
- `gensim`
- `pandas` (although not explicitly used in the provided code, it's a common library for data handling)

You can install them using pip:

```
import nltk
nltk.download('punkt')
nltk.download('stopwords')
nltk.download('wordnet')
nltk.download('vader_lexicon')
```

```
    text_lines = [
        ["commentary line 1", "commentary line 2", ...],
        ["commentary line 1", "commentary line 2", ...],
        ...
    ]
```

```
    from transformers import BertTokenizer, BertForMaskedLM
    from transformers import pipeline

    model_name = "bert-base-uncased"
    tokenizer = BertTokenizer.from_pretrained(model_name)
    model = BertForMaskedLM.from_pretrained(model_name)

    summarization_pipeline = pipeline("summarization")
```

```
n = 5 # Number of highlights to display
v_and_s(text_lines, n)
```

```
def summarize_over(over):
    over_text = " ".join(over)
    summary = summarization_pipeline(over_text, max_length=100, min_length=10, do_sample=False)[0]['summar
    return summary

def highlights(text_lines, n):
    interesting_scores = calculate_interesting_scores(text_lines)
    sorted_scores = sorted(interesting_scores.items(), key=lambda x: x[1], reverse=True)
    top_n_overs = dict(sorted_scores[:n])
    highlights = {}
    for over_number in top_n_overs.keys():
        over_text = text_lines[over_number - 1]
        summary = summarize_over(over_text)
        highlights[over_number] = summary
    return highlights

def calculate_interesting_scores(text_lines):
    stop_words = set(stopwords.words('english'))
    lemmatizer = WordNetLemmatizer()

    def preprocess_text(text_list):
        preprocessed_text = []
        for text in text_list:
            tokens = word_tokenize(text.lower())
            tokens = [lemmatizer.lemmatize(token) for token in tokens if token.isalnum() and token not in
            preprocessed_text.append(tokens)
        return preprocessed_text

    def wordscore(text_lines):
        scores = {}
        for i, sublist in enumerate(text_lines):
            preprocessed_commentary = preprocess_text(sublist)
            preprocessed_commentary = preprocess_text(sublist)
            word2vec_model = Word2Vec(preprocessed_commentary, vector_size=100, window=5, min_count=1, wor
            cricket_related_terms = ["six", "four", "boundary", "wicket", "catch", "stump"]
            related_words = {}
            for term in cricket_related_terms:
                if term in word2vec_model.wv.key_to_index:
                    similar_words = word2vec_model.wv.most_similar(term, topn=5)
                    related_words[term] = [word for word, _ in similar_words]
            count_related_words = sum(len(words) for words in related_words.values())
            scores[i + 1] = count_related_words
        return scores

    def sentimentscore(text_lines):
        sia = SentimentIntensityAnalyzer()
        over_sentiment_scores = {}
        for over_number, over in enumerate(text_lines, start=1):
            over_text = ' '.join(over)
            sentiment_score = sia.polarity_scores(over_text)["compound"]
            over_sentiment_scores[over_number] = sentiment_score
        return over_sentiment_scores

    sentiment_scores = sentimentscore(text_lines)
    word_scores = wordscore(text_lines)

    min_sentiment = min(sentiment_scores.values())
    max_sentiment = max(sentiment_scores.values())
    sentiment_range = max_sentiment - min_sentiment
    normalized_sentiments = {over: (score - min_sentiment) / sentiment_range
                             for over, score in sentiment_scores.items()}

    min_word_score = min(word_scores.values())
    max_word_score = max(word_scores.values())
    word_score_range = max_word_score - min_word_score
    normalized_word_scores = {over: (score - min_word_score) / word_score_range
                              for over, score in word_scores.items()}

    interesting_scores = {over: normalized_sentiments[over] + normalized_word_scores[over]
                          for over in sentiment_scores.keys()}
```

```
                        for over in sentiment_scores.keys()}

    return interesting_scores

def v_and_s(text_lines, n):
    interesting_scores = calculate_interesting_scores(text_lines)
    highlighted_overs = highlights(text_lines, n)
    sorted_highlights = sorted(highlighted_overs.items(), key=lambda x: interesting_scores[x[0]], reverse=
    print("Highlighted Overs:")
    for over, summary in sorted_highlights:
        print(f"Over {over}: {summary}\n")
```

```
text_lines = [
    ["on the pads to start from Amir, no swing, worked down to fine leg to get off the mark.",
    "drifts down leg this time, no swing whatsoever. Warner misses his flick",
    "off the outside half and that races away. Better channel, outside off on a good length. Warner prods
    "leg side-ish again, 140 kph. Amir looking for swing but there is none there. Off the pad to fine leg"
    "much better. 139 kph, coming back in on off, pushed to mid-on",
    "138 kph, nice channel. Around off, full. Renshaw watches the ball onto his bat, dabbed towards point"
    ],
    [
     "nicely bowled. Full around off at 130 kph, Warner blocks to mid-off",
    "short and wide, poor ball. Warner goes back and slaps the ball through cover for four. Easy as ... No
    "superb batting. Good length outside off, sits up for Warner to stay back and punch through cover with
    "no bother says Warner. Gets a good length delivery outside off, Warner square-punches through extra c
    "nicely bowled. 132 kph, full and straight. Off the outside half to gully",
    "nice end, 132 kph outside off, no stroke offered",
    ],
    [
        "beauty. Some swing at last. Goes wide of the crease, bowls it full just outside off. Warner is lu
    "141 kph, tight around off. Defended into the off side",
    "oh no. This has been an awful start from Pakistan. Full outside off from Amir, stay there. Warner dri
    "around off on a full length, defended to mid-on",
    "more runs. On Warner's pads this time and Warner says thanks with a flick through midwicket. Not ever
    "tight around off at 143 kph, blocked",
    ],
    [
    [
        "angled away outside off, left alone",
    "pushed wider outside off, waaaaay too wide to draw a stroke, left alone",
    "136 kph, full outside off, driven to mid-off",
    "134 kph, yorker outside off, jammed out",
    "angled away again, left alone",
    "full and wide, another leave, Some control, a maiden",
    ],
    [
        "140 kph, on the pads, worked to fine leg for a couple. Warner hurries back for two, makes it with
    "excellent change-up. The most impressive factor was the direction, right at Warner's neck. He ducks c
    "like this length, short again at off. Warner sways out this time",
    "prodded into the off side, gets one towards cover",
    "139 kph, back of a length outside off. Renshaw bails his bat out. A good leave",
    "139 kph, good line around off, defended back",
    ],
    ["lovely batting. Full and wide outside off, Warner reaches out for it and lifts the ball over extra c
    "good length on off, 137 kph, defended into the off side",
    "134 kph, short and wide. Quite a bad ball to Warner. He chops one back towards and over the stumps. A
    "nicely bowled. No swing but at least the line is good. Full, straight, blocked",
    "finds the gap for four more. Warner is in some touch. Short ball on leg, Warner takes it on, gets on
    "racing away. He's 38 now. Six overs bowled. Good length outside off, a chopped drive-punch through ex
    ],
    [
     "full and straight, flicked through square leg for four more. Good from Renshaw, keeping his own amic
    "141 kph, excellent single. Full on off, dabbed towards extra cover for one. Warner will want the stri
    "143 kph, Amir goes wide of the crease, good length on off, pushed towards cover",
    "slower ball, 126 kph, Warner drives to wide mid-off. Amir looking to get Warner splicing a drive in t
    "bouncer on off, Renshaw avoids with a duck",
    "140 kph, full outside off, played down towards gully",
    ],
    [
        "137 kph, good length on off, no swing. Dead straight, defended",
    "134 kph, blocked to deep point for one",
    "back of a length on off, Renshaw gets on top of the bounce to defend",
    "good length outside off, left alone",
    "pushed fuller and slightly closer to off. Sixth stump. Renshaw shoulders arms",
    "another leave outside off",
    ],
    [
        "four more. Back of a length on off, sits up for Warner to swat in front of square. There is a mar
    "back of a length outside off, Warner lets that go with a sway. Whaaat?",
    "back of a length on off, dabbed towards gully, who has to dive to save some runs",
    "crashed behind point. Not wide enough to cut. Warner stays inside the line, opens the face and guides
    "full and wide this time. Warner's feet are not close to the ball, the ball flies past the edge. The f
    "brings this one back in on off, blocked to mid-off",
    ],
    [
        "142 kph, Renshaw lets this one go",
    "141 kph, tight around off, defended into the off side",
    "full and wide outside off, no stroke offered",
    "good length, wide outside off. Left alone",
    "another one that is left alone outside off",
    "left alone, a maiden"
]
]
```

```
n = 5
v_and_s(text_lines, n)
```

```
Highlighted Overs:
Over 6:  Warner takes it on, gets on top of the bounce and plays the ball in the gap at backward square leg

Over 7:  Renshaw gives it 143 kph, Amir goes wide of the crease, good length on off, pushed towards cover s

Over 2:  Yasir Shah gets across and saves one nicely bowled . Warner square-punches through extra cover wit

Over 3:  Warner drives straight to Misbah at mid-off with a flick through mid-on . Amir bowls full outside

Over 5:  The most impressive factor was the direction right at Warner's neck . 140 kph, on the pads, worked
```