Extending COMMENTATOR for Code-Mixed Language Pairs: Integration of Gujarati-English

The modifications focus on the backend components (Flask/Python) and frontend (React.js).

Frontend Configuration Changes: For Token-level language identification

File: Home.js

Steps:

1. **Update Language Toggle Function:** Modify the language toggle function to accommodate Gujarati tags instead of Hindi tags.

   Old Implementation:

   ```
   const toggle = letter => {
       if (letter === 'h') {
           return 'e';
       }
       else if (letter === 'e') {
           return 'u';
       } else if (letter === 'u') {
           return 'h';
       }
   };
   ```

   New Implementation:

   ```
   const toggle = letter => {
       if (letter === 'g') {  // 'g' for Gujarati instead of 'h' for Hindi
           return 'e';
       }
       else if (letter === 'e') {
           return 'u';
       } else if (letter === 'u') {
           return 'g';  // 'g' for Gujarati instead of 'h' for Hindi
       }
   };
   ```

2. **Customise the colors used to represent different language tags, including Gujarati.**

   Old Implementation:

```
const StyledWord = styled.div`
   border-radius: 8px;
   padding: 8px 8px;
   text-align: center;
   background-color: ${props => ((props.individualTag) === 'e') ? '#bbdfc8' : '#f3f2c9'};
   background-color: ${props => ((props.individualTag) === 'u') && '#D4DCE9'};
   cursor: pointer;
   display:flex;
   flex: 0 1 10%;
   justify-content: center;
`;
```

   New Implementation:

```
const StyledWord = styled.div`
         border-radius: 8px;
         padding: 8px 8px; text-align: center;
         background-color: ${props => ((props.individualTag) === 'e') ? '#bbdfc8' :
         '#f3f2c9'};
         background-color: ${props => ((props.individualTag) === 'u') && '#D4DCE9'};
         background-color: ${props => ((props.individualTag) === 'g') && '#f3c9c9'};
         // Add new color for Gujarati cursor: pointer;
         display:flex; flex: 0 1 10%;
         justify-content: center; `;
```

Frontend Configuration Changes: For Token-level POS tagging

File: POS.js
Steps:

3. **POS Tags Array Update:** The provided POS tags in the posTags array and their associated colors are based on default settings. You should modify these tags and colors according to the specific POS tagging API/Tool you use in your implementation.

```
const posTags = [
  "NOUN",
  "PROPN",
  "VERB",
  "ADJ",
  "ADV",
  "ADP",
  "PRON",
  "DET",
  "CONJ",
  "PART",
  "PRON_WH",
  "PART_NEG",
  "NUM",
  "X"
];
```

Updating the POS Tag colors:

```
const colorData = {
          "NOUN": "#fad",
          "PROPN": "#87CEEB",
          "VERB": "#BA55D3",
          "ADJ": "red",
          "ADV": "#ACE95B",
          "ADP": "#D74222",
          "PRON": "#E256D5",
          "DET": "#FFA07A",
          "CONJ": "#92B050",
          "PART": "#19E4AE",
          "PRON_WH": "#8A12D3",
          "PART_NEG": "#2AA9BB",
          "NUM": "#C6DA2D",
          "X": "#6A4BD3",
};
```

Backend Configuration Changes: For Token-level language identification

Language Identification Models:

- **Model Integration**: Integrate or update the Microsoft Language identification model with an existing pre-trained model of your choice, like FastText, to support Gujarati-English language identification.
- **Model Training and Updates**: Regularly update and retrain models with new data to improve accuracy.

File: app.py

Steps:

1. **API Enhancements:** Remove the LID Tool Import:

   ```
   from LID_tool.getLanguage import langIdentify
   ```

   With imports for the specific pre-trained model(s) you intend to use:

   ```
   from indictrans import Transliterator
   from langdetect import detect, detect_langs
   import fasttext
   ```

2. **In the admin_file_upload() function, replace the classifier that handles Hindi-English identification with the one for Gujarati-English.**

   Old Implementation:

   ```
   lang = langIdentify(sentence, 'classifiers/HiEn.classifier')
   tags = []

   for elem in lang:
       inter = [elem[0]]
       for i in range(1, len(elem)):
           if elem[i] == '1':
               inter.append(elem[i-1][0])
       if len(inter) == 1:
           inter.append('h')
       tags.append(inter)
   ```

   New Implementation:

   ```
   PRETRAINED_MODEL_PATH = 'path/to/your/pretrained/model'
   model = fasttext.load_model(PRETRAINED_MODEL_PATH)
   ```

```
lang = model.predict(sentence)
tags = []

for elem in lang:
    inter = [elem[0]]
    for i in range(1, len(elem)):
        if elem[i] == '1':
            inter.append(elem[i-1][0])
    if len(inter) == 1:
        inter.append('g')  # 'g' for Gujarati instead of 'h' for Hindi
    tags.append(inter)
```

3. **In the csv_download() function, Update Counters for Gujarati:**

   Old Implementation:

```
en_count = 0
hi_count = 0
token_count = 0
lang_ind_count = 0

for i in range(len(tag)):
    if tag[i]['value'] == 'e':
        en_count += 1
    elif tag[i]['value'] == 'h':
        hi_count += 1
    elif tag[i]['value'] == 'u':
        lang_ind_count += 1
    token_count += 1
```

   New Implementation:

```
en_count = 0
gu_count = 0  # 'gu' for Gujarati instead of 'hi' for Hindi
token_count = 0
lang_ind_count = 0

for i in range(len(tag)):
    if tag[i]['value'] == 'e':
        en_count += 1
    elif tag[i]['value'] == 'g':  # 'g' for Gujarati instead of 'h' for Hindi
        gu_count += 1
    elif tag[i]['value'] == 'u':
        lang_ind_count += 1
    token_count += 1
```

Backend Configuration Changes: For Token-level POS tagging

File: app.py
POS Tagging NLP Libraries/Models:

- **Model Integration**: Integrate or update the codeswitch NLP Library with an existing NLP Libraries/model of your choice, which can support other code-mixed language pairs.
- **Model Training and Updates**: Regularly update and retrain models with new data to improve accuracy.

Steps:

4. **API Enhancements:** Remove the LID Tool Import:

   ```
   from codeswitch.codeswitch import POS
   ```

   With imports for the specific pre-trained model(s) you intend to use:

   ```
   import spacy
   import nltk from nltk import pos_tag
   ```

5. **In the admin_file_upload() function, replace the classifier that handles Hindi-English identification with the one for Gujarati-English.**

   Old Implementation:

   ```
   pos = POS('hin-eng')
   pos_tags = pos.tag(text)
   print(pos_tags)
   ```

   New Implementation:

   ```
   nlp = spacy.load('en_core_web_sm')
   doc = nlp(sentence)
   for pos_tags in doc:
           print(pos_tags.text, pos_tags.pos_)
   ```

   Our findings can be generalised to code-mixed language pairs such as Marathi-English, Bangla-English, etc.