Lexical resources - Modalities for the class project

1 Class project: General Description

The project consists in implementing NLP components for an existing online text editor. The online text editor is already implemented, hence the class project will focus on *adapting* the existing code: your code needs to be integrated with the existing code. This code canvas is available on github. The general idea is to have you interact with a more or less realistic situation.

The objective of the class project is to implement NLP components to complete the existing application. The code was set up so that whatever function written in the relevant python module (`/lexres-site/editor/nlp/services/edits.py` and `/lexres-site/editor/nlp/services/preds.py`) will be executed and processed. Your NLP components should be written as functions in either of these two files.

You are to implement *at least*:

1. a spellchecker using the `edits.py` module, and
2. a predictive text module using the `preds.py` module.

Note that *the final use case of the application is left open for you to decide, and this decision should guide your implementation choices.* For illustration purposes, the final intended usage of the application could vary from designing an editor tailored for a given language (be it Old English or Korean), or for a given literary style (be it poetry or scientific reporting)... Obviously, your take on what this editor will be used for impacts what you should edit and what you can predict.

You may also include supplementary code to improve on the minimal requirements (namely you can provide more than spellchecking and predictive text capabilities). These optional supplementary components can be **anything**: for instance, you could implement functions in `edits.py` to create a thesaurus, and propose potential synonyms for each word, etc. These supplementary components may help substantiate the usage you intend for this editor: eg., if you design it for poetry, it would make sense to add a rhyme suggestion function in the `preds.py` module. Any supplementary components and/or subsequent modification of the existing code has to be documented and fully functionnal. You must guarantee that the application behaves as expected.

2 Organisation

You are to work in pairs. Each of the two students must come from different curricula, preferably the one from the NLP masters, the other not. You are to hand in both

1. the adapted code for your components and
2. a report document (around 5pp.), detailing your implementation choices.

Your code and document should be submitted by mail, *both* to `tmickus@atilf.fr` and `ccerisara@loria.fr`, **before February 12, 2020**. Start by familiarizing yourself with the code. Make sure you can run the program and test it from your computer. Notify your teachers early on if you can't make the code canvas run.

3 Details pertaining to the code canvas

Some supplementary clarifications can be found on the github repository of the code canvas.

3.1 Installation

Create a virtual environment & download the necessary libraries

```
python3 -m venv .lexres
source .lexres/bin/activate
pip3 install -r requirements.txt
```

Go to the main directory of the website and start the app using the Django script:

```
cd lexres-site
python3 manage.py runserver
```

Once the server is started, go to the URL where the project is running, which should be http://127.0.0.1:8000/editor/. From this editor, you can test what you'll be coding.

3.2 Adapting the code canvas

The code canvas contains two (almost empty) python script files: `lexres/editor/nlp/services/edits.py` and `lexres-site/editor/nlp/services/preds.py`. All code can be added to these two files, and will be detected automatically by the web application.

Functions listed in `lexres-site/editor/nlp/services/preds.py` are intended to predict the next word, given what was already written by the user. Functions listed in `lexres-site/editor/nlp/services/edits.py` are meant to propose edits & changes to the text.

You are to implement *at least*:

1. a spellchecker using the `edits.py` module, and
2. a predictive text module using the `preds.py` module.

These files should contain functions which must have a specific signature. For now, these files contain one example each. Namely, all functions in `lexres-site/editor/nlp/services/preds.py` should have a keyword argument "text", and should yield a list of strings. Below is an example:

```python
def pred_foo_function(text=""):
    return ["a", "list", "of", "possible", "next", "words"]
```

Likewise, all functions in `lexres-site/editor/nlp/services/edits.py` should have a keyword argument `text`, and should yield a list of `SpanEdit` objects. Below is an example:

```python
def edit_bar_function(text=""):
    return [
        SpanEdit(
            beg_idx=10,
            end_idx=42,
            edit="You should replace `text[10:42]` with this string."),
    ]
```

The `SpanEdit` constructor used in the `lexres-site/editor/nlp/services/edits.py` module is defined in `lexres-site/editor/utils/nlp.py`. As a general guideline, this `utils/nlp.py` module would be a coherent to choice to declare supplementary classes and data structures.

Any supplementary non-standard library that you use should be added to the `requirements.txt` file. You can easily overwrite the `requirements.txt` with your environment details using the following command:

```
pip3 freeze > requirements.txt
```

4 Evaluation modalities: tips and highlights

The evaluation will be focused on what you understood from this class, how you adapted lexical resources for this task and how you solved each problem. To assess this aspect of your work, evaluation will rely heavily on the submitted 5-page report. This document should contain all the information tracing back and making explicit implementation choices and technical challenges encountered and solved. Although creativity will be highly appreciated, do keep in mind that the finality of the class project is to have you prove your ability to **concretize an idea** in a coherent and well-thought fashion.

We also expect the report document to include an **analysis** of what needs your solutions aim to fulfil, and what use cases it is designed for. We expect this analysis to be in line with your implementation and to highlight the relevant linguistic aspects that drove your design choices. A significant part of the grade will be dedicated to the thoroughness of this analysis.

Code **cleanliness** and code **readability** are important factors to take into consideration when submitting your code for external review. In the scope of this class project, thorough comments may help us understand the intent behind specific segments of code. Avoid hard-coding data if you can. Avoid global variables if you can. Avoid redundant code if you can. Comment your less obvious instructions and your functions.