Generator parafraz

Cel projektu:

Stworzenie programu generującego parafrazy tekstów według zadanych kryteriów.

Opis projektu:

Program komunikuje się z użytkownikiem poprzez terminal. Pobiera zdanie od użytkownika, a następnie informację o tym, jaką modyfikację wprowadzonego zdania chce wykonać. Są to:

1. Zmiana losowego słowa w zdaniu na synonim,
2. Rozbudowa zdana o przymiotniki opisujące rzeczowniki,
3. Zmiana ostatniego słowa w zdaniu tak, aby nowopowstałe zdanie rymowało się z oryginalnym
4. Wprowadzenie dodatkowego zdania tak, aby program zwrócił rymowankę zmieniając ostatnie słowo w jednym z nich.

Po wybraniu modyfikacji, program zwraca zmienione zdanie. Program kończy się w momencie, gdy użytkownik zasygnalizuje chęć przerwania po wykonaniu modyfikacji.

Instrukcja użytkownika:

Program korzysta z bibliotek *requests, spycy, re* oraz *choice.*

Program działa poprzez uruchomienie pliku generator.py przez interpreter.

Budowa programu:

classes.py:

- klasa *Paraphraze* – służy reprezentacji słowa mającego podlec podmianie podczas tworzenia parafrazy tekstu użytkownika
    - atrybuty:
        - *name*: przetrzymuje reprezentację tekstową danego słowa;
    - metody:
        - *_return_json* – przyjmuje adres url, za pomocą biblioteki *requests* pobiera dane spod adresu, zwracając je w formacie json;
        - *_change_to_array* – przyjmuje obiekt formatu json. Z każdego słownika w nim pobiera wartość klucza ‘word’. Zwraca kolekcję tych wartości.
        - *pos* – przy pomocy biblioteki *spacy* sprawdza, jaką częścią mowy jest słowo i zwraca ją w formie zrozumiałej dla datamuse API;
        - *synonym*
        - *expand*
        - *rhyme*

        Korzystają z datamuse API, aby pobrać odpowiednie dane dla słowa określonego przez atrybut name (synonym- synonimy, expand- przymiotniki często używane do opisania słowa, rhyme- słowa rymujące si ę z nim). Zwracają kolekcję słów spełniających kryteria.

operations.py – zawiera funkcje wykonujące modyfikacje na tekstach użytkownika:

- *split_sentance* – przyjmuje obiekt typu string i korzystając z biblioteki *re* zwraca kolekcję, której elementami są pojedyncze słowa (słowa przedzielone apostrofem, np. what’s traktuje jako jedno) i znaki interpunkcyjne z przyjętego obiektu;
- *join_sentence* – przyjmuje kolekcję słów i symboli i zwraca obiekt typu string powstały z połączenia je w taki sposób, aby zachować zasady interpunkcji (brak spacji przed znakiem, pojedyncza spacja po znaku);
- *get_last_word* – zwraca pierwszy od końca element kolekcji obiektów typu string, który nie jest znakiem interpunkcyjnym;

- *lower_first_letter* – przyjmuje obiekt typu string i zmienia jego pierwszą literę na małą;
- *upper_first_letter* – przyjmuje obiekt typu string i zmienia jego pierwszą literę na wielką;
- *parts_of_speech* – zwraca słownik, w którym kluczami są słowa z przyjmowanego zdania, a wartościami odpowiadające im części mowy. Części mowy używane przez bibliotekę *spacy* zmieniane są na takie, które są używane w datamuse API.
- *create_synonym*
    1. Przyjmuje zdanie;
    2. Zmienia pierwszą literę w zdaniu na małą;
    3. Tworzy kolekcję pojedynczych słów i znaków interpunkcyjnych z przyjętego zdania;
    4. Losuje jedno słowo z utworzonej kolekcji i pobiera listę jego synonimów, po czym losowo wybiera z niej synonim do użycia (jeśli nie znajdzie synonimu dla wylosowanego słowa, losuje ponownie; jeśli wylosowany synonim nie jest tą samą częścią mowy (np. słowo look może być rzeczownikiem lub czasownikiem, zależy od kontekstu zdania), co wylosowane słowo, synonim losowany ponownie);
    5. Zwraca przyjęte zdanie z podmienionym przez synonim losowym słowem;
- *expand_with_adj*
    1. Przyjmuje zdanie;
    2. Znajduje w zdaniu rzeczowniki, tworzy obiekty klasy Paraphraze i tworzy słownik, z rzeczownikami, jako klucze, i przymiotnikami je opisującymi, jako wartości;
    3. Zwraca przyjęte zdanie z dodanym losowym przymiotnikiem do każdego rzeczownika.
- *change_to_rhyme*
    1. Przyjmuje zdanie;
    2. Tworzy kolekcję rymów do ostatniego słowa;
    3. Podmienia to słowo losowo wybranym rymem.
- *make_rhyme*
    1. Przyjmuje dwa zdania;
    2. Wybiera losowo zdanie do zmodyfikowania;
    3. W wybranym zdaniu podmienia ostatnie słowo, tak, aby rymowało się ono ze zdaniem niewybranym;
    4. Zwraca zmodyfikowane zdania w postaci dwuwersowego wierszyka.

generator.py

- *main* – funkcja komunikująca się z użytkownikiem
    1. Pobiera od użytkownika zdanie;
    2. Pobiera informację o tym, co użytkownik chce zrobić ze swoim zdaniem;
    3. Wykonuje odpowiednie modyfikacje na zdaniu;
    4. Zwraca zmodyfikowane zdanie.

    Program działa w pętli, pobierając nowe zdanie od użytkownika po wykonaniu zadanej modyfikacji.

Refleksja:

Co można było ulepszyć:

Program można było rozbudować o więcej metod modyfikacji tekstu, np. pozwalać użytkownikowi wybrać słowo ze zdania, które ma zostać zmodyfikowane albo rozbudowanie zdania o przysłówki.

Co się udało:

Wszystkie początkowe założenia zostały zrealizowane. Ponadto, w programie udało się wykorzystać algorytmy NLP przy zamianie słów na synonimy i wyszukiwaniu przymiotników opisujących rzeczowniki w zdaniu. Dzięki temu generowane zdania mają większy sens semantyczny. Ponadto generowane zdania zachowują interpunkcję oryginalnych zdań, dzięki wykorzystaniu wyrażeń regularnych.