

Проект по программированию: «Бот-чистописец»

Хребтовская Лиза, Северина Ксения, БКЛ-194



Идея проекта

Создать программу, которая могла бы анализировать текст и улучшать его на стилистическом уровне:

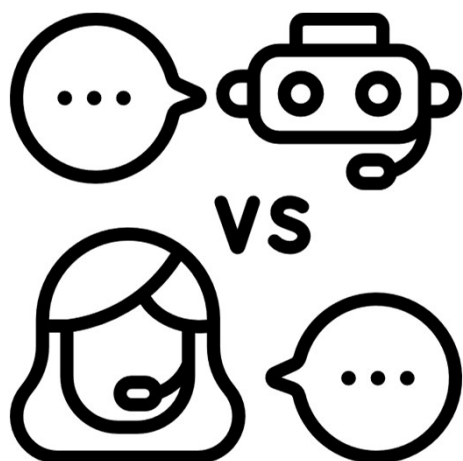
1) Находить слова-паразиты и:

- a. удалять их, если они не несут смысловой нагрузки
- b. предлагать пользователю список синонимов к

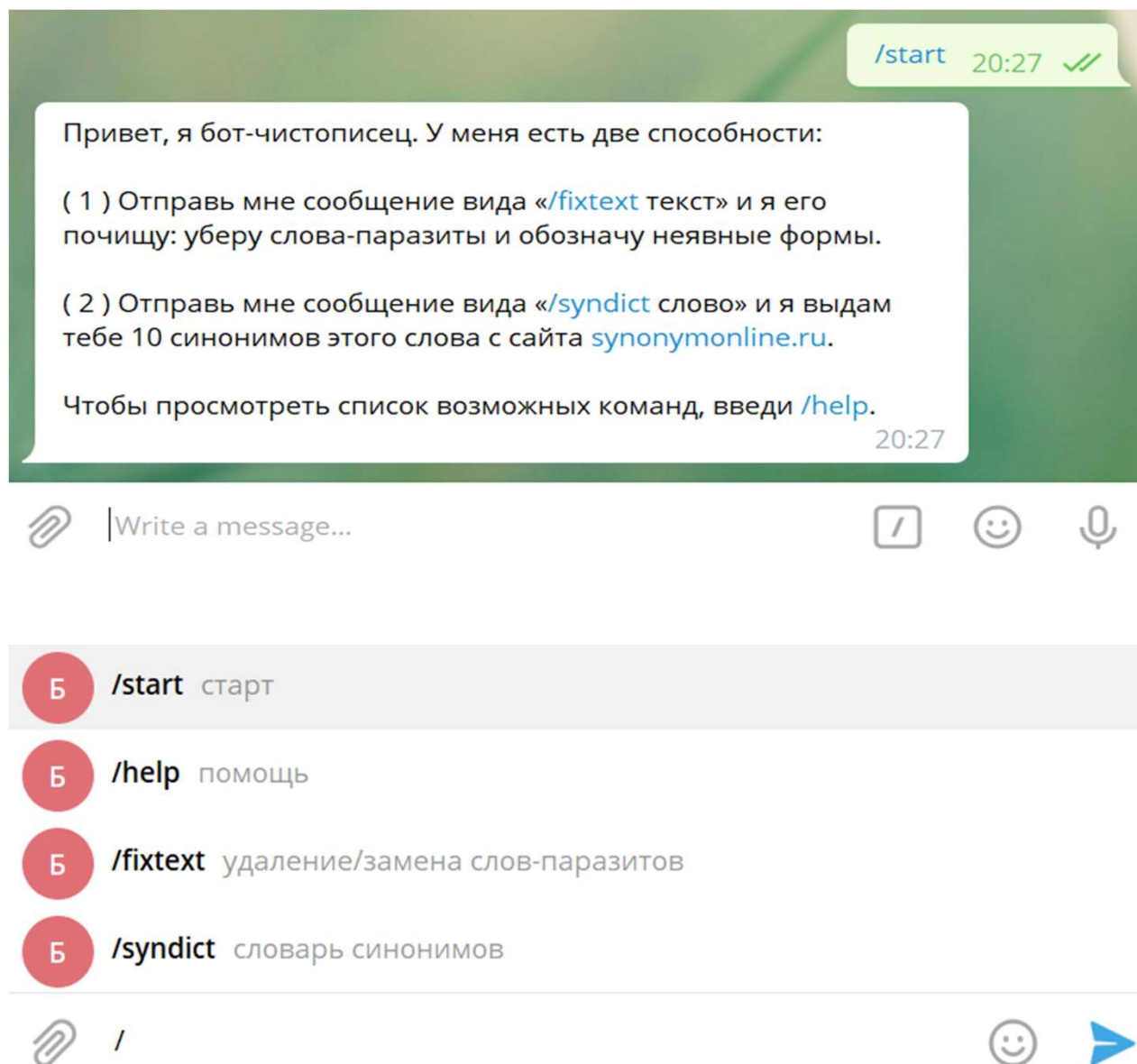
словам-паразитам

2) Предлагать синонимы к любому слову, введённому пользователем

Как работает бот?



Команды



/fixtext Слушай кароч прикинь я сегодня вообще конкретно опоздал. Сначала я как бы проснулся по будильнику, но решил ещё так сказать полежать 3 минуточки в кровати. Прост я подумал что пофигу и по любому успею. А когда блин посмотрел на время внатуре не понял почему уже 10:00. Вообще хз походу я теперь не перепису экзамен 20:42 ✓✓

Пример текста

1. Бот возвращает текст, удаляя слова-паразиты, к которым нет синонимов в словаре + выделяет капсом слова, к которым есть синонимы

Новый текст выглядит так:

„КАРОЧ я сегодня КОНКРЕТНО опоздал. сначала я проснулся по будильнику, но решил ещё полежать 3 минуточки в кровати. я подумал что ПОФИГУ и ПО ЛЮБОМУ успею. а когда посмотрел на время ВНАТУРЕ не понял почему уже 10:00. ВООБЩЕМ ХЗ ПОХОДУ я теперь не перепису экзамен“

2. Программа указывает какие слова были удалены в полученном тексте

Из твоего текста я удалил следующие паразиты:

- как бы
- так сказать
- слушай
- прикинь
- вообще
- прост
- блин

3. Программа предлагает синонимы к выделенным капсом словам-паразитам

В твоём тексте нужно провести следующие замены:

- по любому => в любом случае
- кароч => в двух словах, если кратко, то
- конкретно => сильно
- пофигу => неважно
- внатуре => в самом деле, правда, так и есть
- вообще => в итоге, в общей сложности
- хз => не знаю
- походу => похоже на то, что

/syndict милый 22:26 ✓✓

У слова слишком много синонимов. Ниже представлены 10 возможных вариантов, остальные доступны по ссылке:
<https://synonymonline.ru/%D0%9C/%D0%BC%D0%B8%D0%BB%D1%8B%D0%B9>

приветливый
приятный
ненаглядный
болезный
ласковый
тактичный
возлюбленный
дорогой
драгоценный
желанный

synonymonline.ru

Синонимы к слову «милый» и близкие по смыслу выражения

приветливый, приятный, ненаглядный, болезный, ласковый, тактичный, а также друг...



22:26



Как работает словарь синонимов, если их:

БОЛЬШЕ 10

МЕНЬШЕ 10

/syndict котик 22:27 ✓✓

Повезло! У слова не так много синонимов, и все поместились в одно сообщение.

тюлень
секач
кот
котенок
хорь
котенок
котенок
котишка
коток
мурлыка

22:27

База с синонимами слов

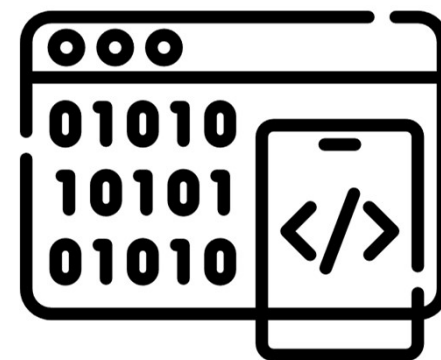
Интересные места в коде

1. Удаление/замена слов-паразитов

```
for parasite in DICT_PAR:
# если паразит есть в тексте...
if parasite in text:
    parasList.append(parasite)
    # и если паразит ни на что не заменяется...
    if DICT_PAR[parasite] == "":
        # то мы его удаляем из текста
        text = text.replace(parasite, "")
        deleteList.append(parasite)
    # но если заменяется...
else:
    # то выделяем в тексте капсом...
    text = text.replace(parasite, parasite.upper())
    # и записываем паразит в отдельный список
    fixList.append(parasite)
```

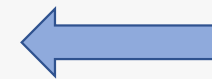
← I. изначальный код:

Удалял “ну” в “уснуть”((



```
new_wordList = wordList

# проверяем каждое слово с новым списке слов текста
for word in range(len(new_wordList)):
    # все буквы маленькие
    new_wordList[word] = new_wordList[word].lower()
    for parasite in DICT_PAR:
        # если слово является паразитом...
        if new_wordList[word] == parasite:
            # записываем его в список всех паразитов
            parasList.append(parasite)
            # если не заменяется, то удаляем и записываем в список удаленных
            if DICT_PAR[parasite] == "":
                wordList[word] = ""
                deleteList.append(parasite)
            # если заменяется, то выделяем капсом и записываем в список для фикса
        else:
            wordList[word] = wordList[word].upper()
            fixList.append(parasite)
```



II. поправленный код:
не реагировал на
паразиты из двух слов(

```
text = text.lower()
```

```
# если паразит = 2 слова
```

```
for parasite in DICT_PAR:
```

```
    if " " in parasite:
```

```
        # если паразит есть в тексте...
```

```
        if parasite in text:
```

```
            parasList.append(parasite)
```

```
            # и если паразит ни на что не заменяется...
```

```
            if DICT_PAR[parasite] == "":
```

```
                # то мы его удаляем из текста
```

```
                text = text.replace(parasite, "")
```

```
                deleteList.append(parasite)
```

```
            # но если заменяется...
```

```
        else:
```

```
            # то выделяем в тексте капсом...
```

```
            caps = parasite.upper()
```

```
            text = text.replace(parasite, caps)
```

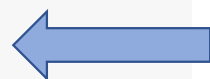
```
            # и записываем паразит в отдельный список
```

```
            fixList.append(parasite)
```

```
# делаем список слов с измененными двойными паразитами
```

```
wordList = list_making(text)
```

```
new_wordList = wordList
```



III. Правильно работающий код



```
# если паразит = 1 слово
```

```
for word in range(len(new_wordList)):
```

```
    for parasite in DICT_PAR:
```

```
        # если слово является паразитом...
```

```
        if new_wordList[word] == parasite:
```

```
            # записываем его в список всех паразитов
```

```
            parasList.append(parasite)
```

```
            # если не заменяется, то удаляем и записываем в список удаленных
```

```
            if DICT_PAR[parasite] == "":
```

```
                wordList[word] = ""
```

```
                deleteList.append(parasite)
```

```
            # если заменяется, то выделяем капсом и записываем в список для фикса
```

```
        else:
```

```
            wordList[word] = wordList[word].upper()
```

```
            fixList.append(parasite)
```



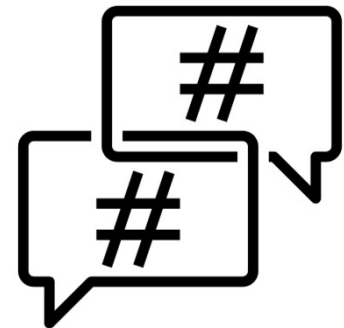
Интересные места в коде

```
"""
СОЗДАЕМ СПИСОК СЛОВ ТЕКСТА
ЧТОБЫ ЗНАКИ ПРЕПИНАНИЯ СЧИТАЛИСЬ ЗА СЛОВА
"""

# тире нет тк оно и так отделено пробелами с обеих сторон
def list_making(text):
    text = text.replace(".", " .")
    text = text.replace(",", " ,")
    text = text.replace("!", " !")
    text = text.replace("?", " ?")
    text = text.replace(":", " :")
    text = text.replace(";", " ;")
    text = text.replace("(", "( ")
    text = text.replace(")", " )")
    text = text.replace("«", "« ")
    text = text.replace("»", " »")
    text = re.sub(r'\\"(.+?)\\"', r'« \1 »', text)
    wordList = text.split()

    return wordList
```

← 2. Функция, отделяющая пробелами знаки препинания



Интересные места в коде

```
"""
УБРАТЬ ПРОБЕЛЫ КОГДА ЗАМЕНЯЕМ ПАРАЗИТЫ
"""

def space_del(text):
    # убираем кучу пробелов, оставляем один красивенький
    text = re.sub(r"\s+", " ", text)

    # убираем пробелы в начале и конце строки
    if text.startswith(" "):
        text = "" + text[1:]
    if text.endswith(" "):
        text = text[:-1] + ""

    # убираем одинарные пробелы перед знаками препинания + после
    text = text.replace(" .", ".")
    text = text.replace(" ,", ",")
    text = text.replace(" !", "!")
    text = text.replace(" ?", "?")
    text = text.replace(" :", ":")
    text = text.replace(" ;", ";")
    text = text.replace(" )", ")")
    text = text.replace(" ( ", "(")
    text = text.replace(" »", "»")
    text = text.replace("« ", "«")

    return text
```



3. Функция, удаляющая лишние пробелы, которые появлялись в процессе работы бота

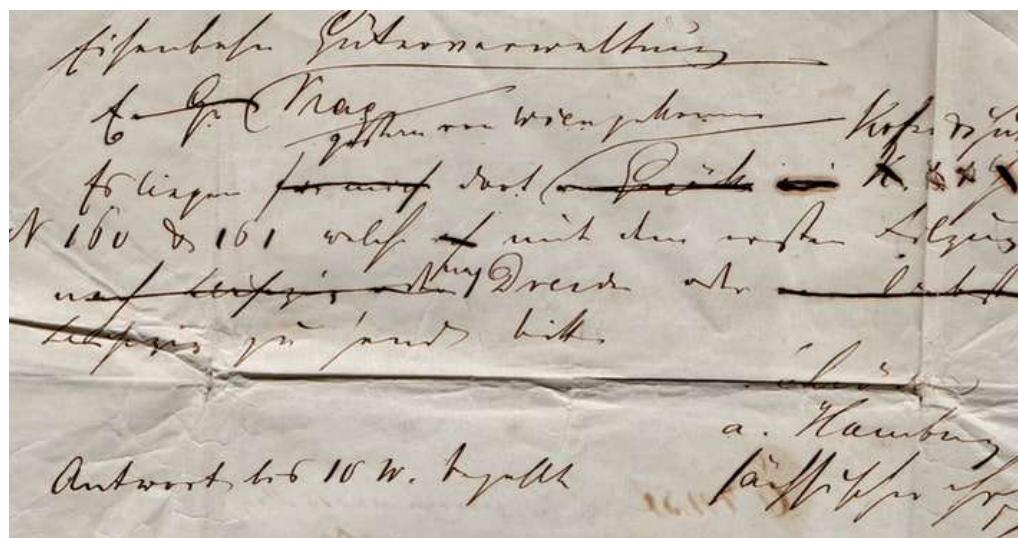
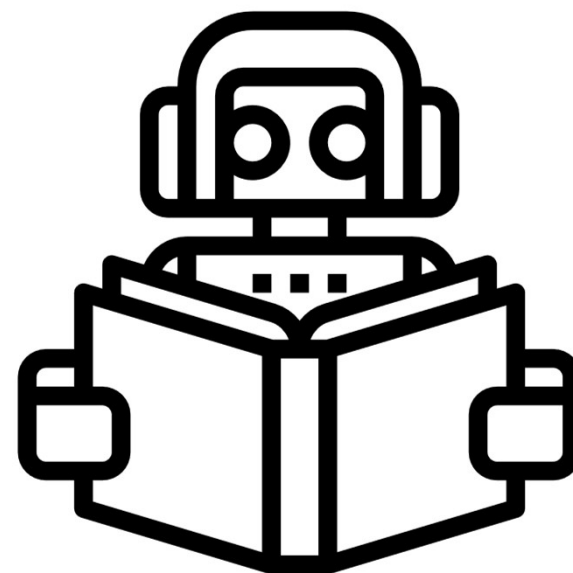


Что получилось?

Бот, который находит в отправленном тексте слова-паразиты, удаляет/заменяет их в зависимости от конкретного случая

+

Бот обладает базой синонимов для любых слов, предлагает 10 самых частотных синонимов



Модули, использованные в коде:

- re
- telebot
- requests
- time