Муниципальное автономное образовательное учреждение “Лицей №2”

эдик, иди к парсеру

**Программирование интерактивного чат-бота**

Выполнили:

ученики 10Б класса

Кузьмин А.А

Шаяхметов А.И

Юриков М.Д

ученик 10 А класса

Почуев Э.К

Руководители:

Солдатова С.Е

Свиридченко С.В

Альметьевск 2021

**Оглавление**

1. Введение
   1. Актуальность
   2. Цели и задачи
2. Основная часть
   1. Модули
      1. Парсер
      2. Морфологический анализатор
      3. Алгоритм поиска
      4. Графическая часть
   2. Принцип взаимодействия модулей
3. Заключение
   1. Финальный вид и оценка
   2. Вывод
4. Источники
   1. Библиотеки для языка Python
   2. Приложения**( не уверен что здесь нужно писать о них)**
      1. Иллюстрация принципа работы парсера
      2. Иллюстрация принципа работы морфологического анализатора
      3. Иллюстрация принципа работы алгоритма поиска
      4. Иллюстрация Графической части
      5. Блок-схема принципа работы бота

**Введение**

Актуальность

Нашей команде было интересно попробовать свои силы в промышленном программировании. В этом году мы приняли участие в хакатоне Цифрового прорыва. Из всех представленных на нём кейсов**(учителя вряд ли поймут что такое кейс**) нас привлек вариант от Ростелекома (чат-бот) по следующим причинам:

* В наше время создание чат-ботов является одним из самых прибыльных сфер в ИТ
* Боты позволяют минимизировать затраты на обучение специалистов

Цели

* Создание интерактивного бота
* Поднятие квалификации
* Полезные знакомства
* Пополнение портфолио

Задачи

1. Разбить задачу на подзадачи (модули)
2. Осуществить сборку модулей в единую систему
3. Провести тестирование, выявить баги
4. Доработать прототип
5. Провести оценку итогового продукта, сравнить с аналогами
6. Запустить на хостинге

**Основная часть**

Для удобства выполнения поставленной задачи было принято решение разделить работу на модули:

* Парсер
* Морфологический анализатор
* Поисковой алгоритм
* Графическая оболочка

Парсер

Прежде чем говорить о работе модуля, нужно понять что такое парсер. **Парсер** - это ПО, выделяющее определенные части информации из массива данных. Алгоритм работы парсера может различаться в разных реализациях, но основной принцип остается неизменным.

* Программа сканирует данные, поступающие на вход, будь то текст, веб-страница или другой набор информации, и вычленяет из них некоторые элементы.
* Что именно будет выделять парсер из массива данных – зависит от конкретной задачи. Обычно программы можно настраивать таким образом, чтобы получать нужные результаты.
* Правила поиска чаще всего задаются регулярными выражениями – строками, составленными по определенным правилам и дающими программе пояснение, что и как искать.
* На основе собранной информации формируется отчет или таблица, в которой отражены все полученные результаты.

Парсер для этого бота работает с помощью трех библиотек для Python:

* requests
* BeautifulSoup4[[1]](#footnote-0)
* CSV

Библиотека requests используется для получения html[[2]](#footnote-1) кода страницы путем запрашивания данных страницы с помощью функции get[[3]](#footnote-2) и его сохранения с помощью встроенных функций Python open(), read(), write() и close(). Однако данная библиотека не позволяет выделить нужную информацию страницы сайта и сократить время на сбор информации. Для этого нужна библиотека bs4. Она позволяет определять теги[[4]](#footnote-3) html кода, находить нужную информацию(с помощью функции find\_all(тег, класс(по умолчанию None))) и, на основе полученного, с помощью библиотеки CSV, сохранять и использовать данные в последующем.

Изучив сайт и его html-разметку, нужно получить необходимые теги для последующей работы, а именно: выделить полученные данные(ссылки и текст) и внести их в массивы[[5]](#footnote-4)(links,names).(Однако можно столкнуться с рядом трудностей: код страницы может неправильно отображаться ввиду некорректной кодировки; часть кода разметки не сохраняется так как работает за счет скриптов). Но для корректного нахождения запроса в последующем требуется совместить элементы массива links(ссылки) с элементами массива names(текст гипертекста). Для этого можно воспользоваться присваиванием одного из объектов к другому с помощью использования **словарей** - встроенной коллекции произвольных объектов с доступом по ключу(их иногда еще называют ассоциативными массивами или хеш-таблицами). Ключом в данном случае будет являться текст в массиве names, а присваиваемый объектом- ссылка в массиве links.

Данный модуль является основополагающим для всего проекта. В последующем данный модуль будет возвращать словарь(показано в приложении <NUMBER>), необходимый для других модулей.

Алгоритм поиска

там все ищут и все найдут. Еще что-то про релевантность

Морфологический анализатор

Морфологический анализ - это определение характеристик слова на основе того, как это слово пишется. При морфологическом анализе **не** используется информация о соседних словах.

Для правильного и эффективного осуществления алгоритма поиска и сопоставления информации из базы данных и запроса пользователя было принято создать отдельный модуль, который отвечал за разделение входящего запроса на отдельные слова и их «подгонку» к одному единому стандарту по заданным характеристикам.

Таким образом, цепочка преобразований исходного текста в итоговый вариант выглядит так:

1. Получение запроса
2. Разделение запроса на слова, добавление их в массив
3. Приведение каждого элемента массива к форме именительного падежа, единственного числа. Все заглавные слова заменяются строчными
4. Сравнение полученного массива с массивом “паразитных” слов, исключение совпадений
5. Вывод результата

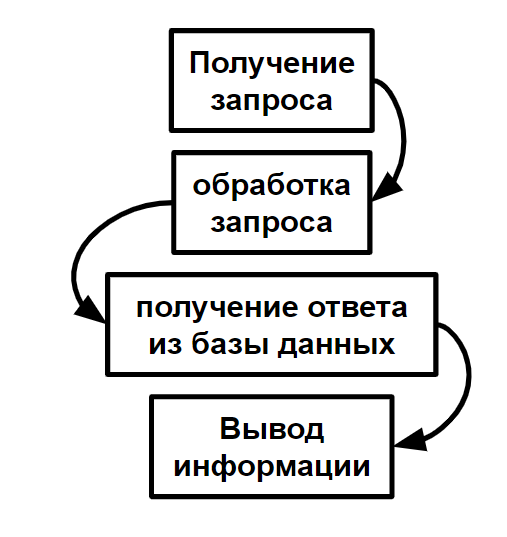
(Пример работы алгоритма на примере см. в приложении №\*) (Картинку кинул в “картинки для реферата”)

Для осуществления работы этого алгоритма было решено использовать библиотеку pymorphy2 - морфологический анализатор для русского языка, написанный на языке Python и использующий словари из [OpenCorpora](http://opencorpora.org/). Он работает под Python 2.6, 2.7, 3.2 и 3.3 и распространяется по лицензии MIT.В коде она отвечает за приведение слова к «нормальной форме», прочее делается встроенными командами python.

Графическая оболочка

Принцип работы

**приложение к принципу работы**



**Заключение**

Финальный вид и оценка

Исходя из всего вышесказанного, бот можно оценить как весьма удовлетворительный продукт. Он выполняет свою задачу в полном объеме от изначально заданного.

Вывод

Мы показали вам нашу реализацию чат-бота, рассказали о его плюсах и минусах, выделили особенности и сравнили с аналогами.

Мы улучшили наши навыки программирования и работы в команде.

В процессе разработки мы получили важные советы по презентации нашего проекта от нашего питчера, а в конце хакатона он оставил нам свои контакты. Через пару дней к нам обратился представитель Ростелекома с предложением сотрудничества.

**Источники**

Документация библиотек на языке Python

1. requests
2. beautifulsoup4
3. pymorphy2
4. pyTelegramBotAPI
5. os
6. CSV

Ссылки

<https://www.activetraffic.ru/wiki/parser/>

1. далее bs4 [↑](#footnote-ref-0)
2. элемент языка разметки гипертекста. [↑](#footnote-ref-1)
3. также может использоваться метод post в случае автоматической отправки формы [↑](#footnote-ref-2)
4. элемент языка разметки гипертекста. [↑](#footnote-ref-3)
5. массив в Python - список (list) [↑](#footnote-ref-4)