МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РФ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ

ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

«ТЮМЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»

ИНСТИТУТ МАТЕМАТИКИ И КОМПЬЮТЕРНЫХ НАУК

КАФЕДРА ИНФОРМАЦИОННЫХ СИСТЕМ

ОТЧЁТ

**по учебной практике**

Место прохождения практики

Кафедра Информационных Систем ТюмГУ

|  |
| --- |
| **Выполнил:**  Студент группы 25ИСИТ169  Зухритдинов А.Н.  (Подпись)  **Научный руководитель:**  к.ф.н., доцент кафедры ИС  Бидуля Ю.В.  (Подпись) |

Тюмень – 2018

Оглавление

[Введение 3](#_Toc2222112)

[1 Описание предметной области 6](#_Toc2222113)

[2 Постановка задачи 7](#_Toc2222114)

[2.1 Название проекта 7](#_Toc2222115)

[2.2 Назначение проекта 7](#_Toc2222116)

[2.3 Цель разработки 7](#_Toc2222117)

[2.4 Задачи 7](#_Toc2222118)

[2.5 Ожидаемый результат 7](#_Toc2222119)

[2.6 Функциональные требования 7](#_Toc2222120)

[Средства разработки 8](#_Toc2222121)

[3 Проектирование программного обеспечения 9](#_Toc2222122)

[3.1 Средства разработки 9](#_Toc2222123)

[3.2 Описание приложения 9](#_Toc2222124)

[3.3 Программный код 10](#_Toc2222125)

Введение

Анализ данных – область математики, занимающаяся построением и исследованием наиболее общих математических методов и вычислительных алгоритмов извлечения знаний из экспериментальных (в широком смысле) данных; процесс исследования, фильтрации, преобразования и моделирования данных с целью извлечения полезной информации и принятия решений.

Информационная система (ИС) – это система, включающая совокупность компонентов, отвечающих за выполнение ряда функций: извлечение, обработка, хранение, а также организационных ресурсов: человеческие, финансовые, технические, программные и т.д.

Курсовая работа включает в себя разработку приложения для обработки отзывов покупателей интернет-магазинов, пользователей интернет ресурсов, что предоставляют услуги и сервисы, переработки текста, и ведение анализа данных на имеющихся ресурсах с помощью применения готовых модулей для парсинга, сбора данных из ресурсов на языке Python и использование алгоритмов обработки текста и принятия решений, такие как «токенизация» и «стемминг». Данная задача считается релевантной из задач обработки данных и принятия решений на основании данных. Стемминг – это процесс нахождения основы слова. Это, в свою очередь, помогает избежать принятие неадекватных, нерелевантных решений. Токенизация – выделение слов из приложений, это считается одним из фундаментальных апсектов при обработке текста. выделения сущностей тут не обойтись, т.е. здесь имеет смысл обозначения слов по их категориям, частям речи, чтобы легче было структурировать данные. Большая часть интернета плохо структурирована, так же, как сам естественный язык пользователей. Разбором алгоритмов и методов обработки естественного языка занимается особое направление искусственного интеллекта: NLP (Natural Language Processing). Пайплайн – это разработка программных продуктов или разбор алгоритма по этапам. Пайплайн «NLP», подробная схема указана на Рис.1:

* Разбор предложений по специальным знакам. В основном, это точки, восклицательные знаки и т.п. Это помогает эффективнее разбирать слова по частям речи и составлять дерево сущностей;
* Токенизация включает в себя выделение слов по атрибутам, т.е. это своего выделение слов и сохранение их в словари;
* Определение частей речи. После данного этапа у каждого токена(слова) появляется значение, наименование его части речи;
* Лемматизация – приведение к начальной форме слов: у глаголов к инфинитиву, у существительных к именительному падежу ед. числа и т.п;
* Выделение и удаление «стоп-слов». Данный шаг включает в себя удаление разного рода частей речи (союзы, предлоги и т.п.), чтобы и смысл не менялся, и для увеличения эффектности алгоритма, т.е. для отсечения «воды»;
* Парсинг зависимостей. Создание дерева зависимостей между токенами, с обозначением их связей. Целью алгоритма является дерево токенов, у которых есть единственный родитель и связи между ними;
* Поиск групп существительных. Каждое сочетание токенов принимается за существительное и группируются, если это возможно;
* Распознавание именованных сущностей. Каждый токен, в соответствии с NER-моделями (Named Entity Relations), ставится в соответствие по его признакам. Данный этап фильтрует слова, дав возможность отличить;
* Разрешение кореференции. Это избавляет утечку информации за счет замены местоимений «его, ее, их и т.п.» на имена существительные, на то, о чем идет в данном тексте речь.

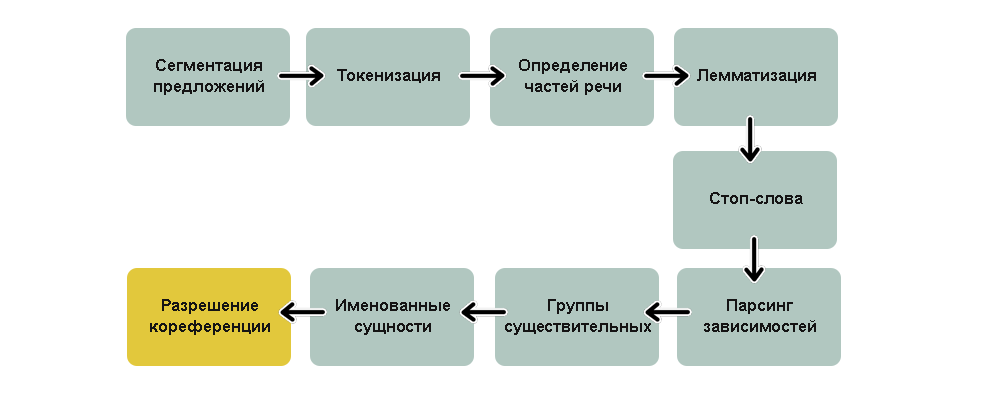


Рисунок 1 – схема NLP

Для реализации задачи курсовой работы будут использованы не все перечисленные алгоритмы и шаги. Стоит отметить то, что для данной работы должна быть представлена выборка из немалого количества отзывов, товаров и услуг, т.е. количество интернет ресурсов должно быть большим, чтобы удовлетворять условия задачи.

1 Описание предметной области

Существуют ряд интернет-сайтов: магазины, сайты услуг и прочие. Каждый пользователь вправе оставлять отзывы: положительные, негативные или нейтральные. Для выявления характера отзывов многие ресурсы используют сентиментный анализ, он же тональный. Если говорить проще, то этот анализ делает вывод на основе мнений людей, негативный или положительный. Но его чаще всего используют, кроме выводов о тоне мнений, для выделения аспектов(субъектов), относительно которых люди выражают свои взгляды. Например, аспектом можно считать камеру, если человек хвалит или, наоборот, выражает негативные эмоции по поводу «объектива» или «плохой контрастности» фотографий. Допустим, имеется выборка по интернет-магазинам, и заказчик требует создать список товаров для продажи. Для этой задачи можно реализовать сентиментный модуль для построения списка товаров на основе рецензий покупателей. Но этот подход не всегда эффективен, так как отзывы могут быть подкупными или «фейковыми». Здесь и появляется необходимость воспользоваться машинным обучением. Но и это иногда является не столь эффективным. В данной задаче курсовой работы требуется реализовать систему для сбора данных из сайтов и анализа их тона с последующим выводом отчетов. Так же, для достижения этой цели, надобно использовать предобработку полученных или существующих данных.

2 Постановка задачи

## 2.1 Название проекта

Система для сбора и анализа отзывов о качестве товаров и услуг.

## 2.2 Назначение проекта

Система предназначена для вывода товаров, полученных с помощью парсинга или с использованием ключей открытого доступа(API), соответствующих требованиям пользователя. Также система должна выявлять характеристику по аспектам и выводить отчеты.

## 2.3 Цель разработки

* Разработать метод сбора и анализа данных Интернет-ресурса
* Реализовать в виде приложения для анализа отзывов и услуг

## 2.4 Задачи

* Изучить средства программирования приложений для анализа данных.
* Изучить существующие методы предобработки и анализа текстов.
* Изучить методы анализа текста отзывов.
* Разработать модуль парсинга Интернет-источников.
* Разработать модуль анализа текстов с выводом отчетов.

## 2.5 Ожидаемый результат

Программное приложение на языке Python.

## 2.6 Функциональные требования

* Парсинг данных из Интернет-ресурсов
* Сохранение данных в формате CSV в формате:
  + Категория (Наименование, Ссылка)
  + Подкатегория (Наименование, Ссылка)
  + Продукт (Наименование, Цена, Категория, Ссылка)
  + Отзыв (Комментарий, Оценка, ФИО, Ссылка к продукту)
* Сравнение цен продуктов и услуг источников
* Вывод количественной статистики
* Возможность выгрузки на другие базы данных
* Формирование отчетов

3 Проектирование программного обеспечения

## 3.1 Изучение средства программирования

На практику был выбран язык Python, как наиболее релевантный язык для анализа данных. В итоге курса были изучены основные фундаментальные принципы анализа данных и язык программирования Python. На Рис.2 представлен сертификат.



## 3.2 Средства разработки

* Среда разработки:
  + Sublime Text 3
  + Visual Studio
* Python

Использование “Sublime Text 3” сильно повысило компиляцию программного кода из-за отсутствия излишних модулей, что есть в Visual Studio. Для реализации модуля парсера был выбран Python: есть большинство модулей, для упрощенного выполнения поставленных задач, прост в изучении.

## 3.3 Описание приложения

Разработан модуль по сбору, обработке и сохранению данных о товарах в CSV-таблицы, для дальнейшего манипулирования.

Были использованы модули для связи с операционной системой(os), записыванием и считыванием информации о товарах(‘csv’), обработки json-объектов(‘json’), парсинга страниц(‘BeautifulSoup’), и получения get-запросов(‘requests’).

Для общей информации, приведу ниже описание модулей:

* BeautifulSoup – модуль, отвечающий за парсинг данных из сайта. Основными его функциями являются ‘find(), find\_all()’,которые принимают в аргументы HTML-теги и набор атрибутов, такие как классы, уникальные номера и т.д.
* Os – модуль операционной системы, обеспечивающий доступ к папкам, файлам, событиям и т.д.
* CSV – считывает, записывает в CSV-файлы данные, полученные при обработке информации. Основные функции: ‘writer(), reader()’.
* Requests – модуль, который отпраляет Post, Get запросы к серверсу и получает либо идентифицирующий номер ответа, либо объект от сервера.

Далее будет представлен код модуля парсера «МВидео».

## 3.4 Программный код

Модуль «parser.py». Тут создается объект парсера «МВидео» после вхождения в главный метод. Далее идут функции, которые возвращают категории и подкатегории.

from MVideoParser import \*

from datetime import datetime

# from multiprocessing import Pool

def main():

start = datetime.now()

# Создает объект парсера MVideo

html\_ = MVideo()

# Возвращает список категорий

print(html\_.GetCategories())

# Возвращает список подкатегорий

print(html\_.GetSubCategories())

# Сохраняет товары в CSV-файл и возвращает список товаров.

# Но так как товаров может быть очень много, функции сохранения и возвращения списка выключены.

print(html\_.GetProducts())

end = datetime.now()

totalTime = end-start

print(str(totalTime.total\_seconds())) # Потраченное время на парсинг и сохранение файлов

if \_\_name\_\_ == '\_\_main\_\_':

main()

Модуль парсера «МВидео» - «MVideoParser». Данный модуль является главным, который выполняет роль парсера. Он получает данные и сохраняет в CSV-файлы. В модуле существуют следующие методы:

* \_\_init\_\_(self) - инициализирует объект парсера,
* get\_url(self) – возвращает ссылку объекта,
* get\_html(self) – возвращает код страницы,
* getCategories(self) – возвращает и записывает в БД все категории
* getSubCategories(self) – парсит категории и возвращает по ним подкатегории
* getProducts(self) – по полученным подкатегориям считывает информацию о товарах и записывает их в БД
* getPages(self) – возвращает количество страниц на страницах, где размещены товары
* isPaginable(self) – указывает на то, что страница может быть пропарсена, или имеет другую структуру.

from bs4 import BeautifulSoup

import requests

import csv

import os

import json

class MVideo(object):

# Поля класса парсера МВидео

MainUrl = 'http://mvideo.ru'

Categories = {}

Products = {}

ProductsCount = 0

CategoriesColumnNames = ['Категория','Url']

ProductsColumnNames = ['Продукт','Цена(руб.)','Категория','Url']

url = 'https://www.mvideo.ru/catalog'

CategoriesFileName = 'files/categories.csv'

SubCategoriesFileName = 'files/subcategories.csv'

ProductsFileName = 'files/products.csv'

def \_\_init\_\_(self):

self.url = self.url

def get\_url(self):

return self.url

def get\_html(self):

response = requests.get(self.get\_url())

return response.text

def GetCategories(self):

if(os.path.isfile(self.CategoriesFileName)):

print("Считываем с файл '"+self.CategoriesFileName+"'")

with open(self.CategoriesFileName, 'r',encoding='utf-8', newline='') as openedCsv:

reader = csv.reader(openedCsv, delimiter='|')

for item in reader:

if(not item[0]=='Категория'):

CategoryName = item[0]

CategoryUrl = item[1]

AbsoluteUrl = CategoryUrl

self.Categories[CategoryName] = AbsoluteUrl

if 'Акции' in self.Categories.keys():

del self.Categories['Акции']

else:

soup = BeautifulSoup(self.get\_html(),'lxml')

Categoriessoup = soup.find("div",{"class":"js-catalog-container"}).find\_all("div",{"class":"c-catalog-item"})

self.CategoriesColumnNames.append('Категория')

self.CategoriesColumnNames.append('Url')

for item in Categoriessoup:

CategoryName = item.find('h3').find('a').getText().strip()

CategoryUrl = item.find('h3').find('a').get('href').strip()

AbsoluteUrl = self.url + CategoryUrl

self.Categories[CategoryName] = AbsoluteUrl

del self.Categories['Акции']

filename = self.CategoriesFileName

columnsName = ['Категория','Url']

dictionary = self.Categories

self.writeCSV(filename,columnsName,dictionary)

return self.Categories

def GetSubCategories(self):

SubCategory = {}

if(os.path.isfile(self.SubCategoriesFileName)):

print("Считываем с файл '"+self.SubCategoriesFileName+"'")

with open(self.SubCategoriesFileName, 'r',encoding='utf-8', newline='') as openedCsv:

reader = csv.reader(openedCsv, delimiter='|')

for item in reader:

if(not item[0]=='Подкатегория'):

SubCategoryName = item[0]

SubCategoryURL = item[1]

SubCategory[SubCategoryName] = SubCategoryURL

else:

SubCategoryName = ''

SubCategoryURL = ''

self.Categories = self.GetCategories()

for i,item in self.Categories.items():

self.url = item

soup = BeautifulSoup(self.get\_html(),'lxml')

try:

objSubCategory = soup.find('ul',{'class':'sidebar-categories-list'}).find\_all('li')

for i in objSubCategory:

SubCategoryName = i.find('a').getText()

SubCategoryURL = self.url + i.find('a').get('href')

SubCategory[SubCategoryName] = SubCategoryURL

except Exception:

print('Что-то пошло не так')

finally:

print("Подкатегория '" + SubCategoryName + "' готова к парсингу")

columnsName = ['Подкатегория','Url']

dictionary = SubCategory

filename = self.SubCategoriesFileName

self.writeCSV(filename,columnsName,dictionary)

return SubCategory

def GetProducts(self):

if(os.path.isfile(self.SubCategoriesFileName)):

with open(self.SubCategoriesFileName, 'r', encoding='utf-8', newline='') as CsvReader:

reader = csv.reader(CsvReader, delimiter='|')

for row in reader:

if(not row[0]=='Подкатегория'):

try:

#! Количество страниц по подкатегориям.

totalPages = self.GetPages(row[1])

if(not totalPages==None):

for i in range(0,totalPages+1):

if(not totalPages==0):

pagePart = '/page=' if 'f' in row[1].split('/') else '/f/page='

if(i<=totalPages):

i += 1

self.url = row[1] + pagePart + str(i)

else:

self.url = row[1]

#print(self.url,'total pages =',totalPages)

soup = BeautifulSoup(self.get\_html(),'lxml')

try:

productsDict = dict.fromkeys(['ProductName','ProductLocalPrice','ProductCategoru'], 'None')

productsSoup = soup.find('div',{'class':'o-plp-container\_\_product-listing'}).find\_all('div',{'class':'product-tiles-list-wrapper'})

for products in productsSoup:

productsList = products.find\_all('div',{'class':'c-product-tile\_\_description'})

productsDict = {}

for index,productObject in enumerate(productsList):

#! Блок, отвечающий за, непостредственно, парсинг

ProductDictObject = json.loads(productObject.find('h4').find('a').get('data-product-info'))

ProductName = ProductDictObject['productName']

ProductPrice = ProductDictObject['productPriceLocal']

ProductCategory = ProductDictObject['productCategoryName']

ProductUrl = self.MainUrl + productObject.find('h4').find('a').get('href').strip()

productsDict[index] = {'ProductName':ProductName,'ProductPrice':ProductPrice,'ProductCategory':ProductCategory,'ProductUrl':ProductUrl}

self.writeCSVProducts(self.ProductsFileName,self.ProductsColumnNames,productsDict)

except Exception:

print('Error on parsing the certain product!')

except Exception:

print('Error on scraping the subcategories!')

else:

self.GetSubCategories()

self.GetProducts()

def GetPages(self,html):

self.url = html

html = self.get\_html()

soup = BeautifulSoup(html,'lxml')

optionsPagination = {'class':'c-pagination'}

lastPageOption = {'class':'c-pagination\_\_num c-btn c-btn\_white'}

try:

pages = 0

if(self.isPaginatinable(soup)==True):

try:

pages = soup.find('div',attrs=optionsPagination).find('a',attrs=lastPageOption).getText()

except Exception:

pages = 0

else:

print('Структура страницы отличается от алгоритма для парсинга.')

return None

finally:

return int(pages)

def isPaginatinable(self,soup):

optionsCheck = {'class':'product-tiles-list-wrapper'}

try:

check = soup.find('div',attrs=optionsCheck)

except Exception:

check = None

print('error on accessing to pagination.')

if(not check==None):

return True

else:

return False

def writeCSV(self,filename=CategoriesFileName,columnsNames=CategoriesColumnNames,dictionary=Categories):

if(os.path.isfile(filename)):

print('Файл уже существует, откройте его')

else:

with open(filename, 'w',newline='',encoding='utf-8') as file:

writer = csv.writer(file,delimiter="|")

writer.writerow(columnsNames)

for key,item in dictionary.items():

if('promo' not in str(item)):

writer.writerow([str(key).strip().replace('\"','') ,str(item).strip().split('?')[0]])

res = "Файл '" + filename.split('/')[1] + "'записан. Путь: '" + filename.split('/')[0] + "/'"

print(res)

def writeCSVProducts(self,filename,columnsNames,dictionary):

if(not os.path.isfile(filename)):

with open(filename, 'a',newline='',encoding='utf-8') as file:

writer = csv.writer(file,delimiter="|")

writer.writerow(columnsNames)

with open(filename, 'a',newline='',encoding='utf-8') as file:

writer = csv.writer(file,delimiter="|")

for key,item in dictionary.items():

writer.writerow([str(item['ProductName']),str(item['ProductPrice']),str(item['ProductCategory']),str(item['ProductUrl'])])

res = "Файл '" + filename.split('/')[1] + "'записан. Путь: '" + filename.split('/')[0] + "/" + filename.split('/')[1] + "'"

print(res)

Заключение

По завершению практической работы, был изучен язык программирования Python, использованы как внутренние, так и его внешние, основные модули для создания методов сбора, обработки и сохранения данных. Была изучена предметная область и выявлены проблемные составляющие. Также был разработан модуль парсинга веб-сайта по объектно-ориентированному подходу. Все файлы программного приложения размещены на GitHub в открытом доступе. Программа будет дополняться новыми функциональными возможностями, чтобы решить проблему курсовой работы.

Литература

1. Марк Лутц. Изучаем Python. 4 – издание / Символ-Плюс, 2011. – 1272 с.
2. Уэс Маккини. Python и анализ данных / ДМК Пресс, 2013. – 484 с.
3. Дж. Вандер Плас. Python для сложных задач. Наука о данных и машинное обучение / Питер, 2017. – 576 с.
4. Кэти О’Нил. Data Science. Инсайдерская информация для новичков. Включая язык R. / Райчел Шатт / Питер, 2013. – 368 с.
5. Андрей Ширшов. Парсим на Python: Pyparsing для новичков / [https://habr.com/ru/post/239081/], 2014.
6. Адам Геитгей. Natural Language Processing / [<https://medium.com/@ageitgey/natural-language-processing-is-fun-9a0bff37854e>], 2018.
7. Hector Garcia-Molina. Database Systems: The Complete Book: книга/ Jeffrey D. Ullman, Jennifer Widom. – 2-е издание. – Pearson, 2008. – 1248 с.
8. Terry Halpin. Information Modeling and Relational Databases: книга / Terry Halpin, Tony Morgan. – 2-е издание. – Morgan Kaufmann, 2008. – 976 с.
9. Бернхард Тальхайм. Entity-Relationship Modeling: Foundations of Database Technology: книга. – Springer, 2000. – 628 с.
10. Christian Mancas. Conceptual Data Modeling and Database Design: академическое пособие. - Apple Academic Press, 2015. – 698 с.