Министерство науки и высшего образования РФ

Пермский национальный исследовательский политехнический университет

Электротехнический факультет

Кафедра Информационные технологии и автоматизированные системы

Курсовая работа по дисциплине

«Алгоритмические языки программирования»

Тема: «АНАЛИЗ ТЕКСТА: СОСТАВЛЕНИЕ КАРТЫ ЛИТЕРАТУРНОГО ПРОИЗВЕДЕНИЯ»

Выполнили студенты гр. АСУ2-19 1м

Гребеньщикова Елизавета Витальевна

|  |
| --- |
|  |
| (подпись) |

Орлова Екатерина Дмитриевна

|  |
| --- |
|  |
| (подпись) |

Фоминых Полина Юрьевна

|  |
| --- |
|  |
| (подпись) |

Проверил к.т.н, доцент кафедры ИТАС,

Курушин Даниил Сергеевич

|  |
| --- |
|  |
| (подпись) |

Пермь, 2019

**Оглавление**

[Введение 3](#_Toc26389282)

[Подходы к анализу текста 4](#_Toc26389283)

[Концепция алгоритма 6](#_Toc26389284)

[Реализация алгоритма 9](#_Toc26389285)

[Выводы 10](#_Toc26389286)

[Список использованной литературы 11](#_Toc26389287)

[Приложение 1 12](#_Toc26389288)

# Введение

В современном мире идет стремительное развитие робототехники. При этом разработки в это области имеют достаточно большой спектр применения: от медицинских учреждений и промышленности, до решения проблем в ЖКХ. Одним из перспективных направлений развития робототехники является компьютерная лингвистика. Необходимость новых разработок в данном направлении обуславливается тем, что робототехника, так или иначе, взаимодействует с человеком. Основным средством коммуникации человека является речь. Таким образом, обучение роботов распознавать и анализировать текст является важной задачей.

Проблему обработки естественного языка принято делить на 4 уровня: морфологический, синтаксический, семантический и прагматический. Существует достаточно большое количество парсеров, которые позволяют анализировать текст на морфологическом уровне (анализ формы слова и его грамматические категории) и на синтаксическом уровне (анализ взаимоотношений между словами). Такой анализ позволяет обрабатывать большое количество данных, выделять необходимую информацию и преобразовывать ее.

Данная работа является примером использования морфологических анализаторов для создания карты литературного произведения. Актуальность работы состоит в том, что данные разработки можно использовать в системе анализа языка для обучения роботов. Целью работы является анализ текста и визуализация перемещений персонажей литературного произведения по известной карте местности.

## Подходы к анализу текста

Существует два основных подхода к анализу текста: нейронные сети и корпусная лингвистика. Нейронные сети сейчас являются одним из самых популярных методов интеллектуального анализа данных. Они способны решать самые сложные задачи обработки естественного языка. Почти во всех предметных областях они показывают более качественные результаты, чем другие методы машинного обучения. Для анализа текстовых данных чаще всего используются сверточные и рекуррентные нейронные сети (СНС и РНС).

Сверточные нейронные сети позволяют находить короткие зависимости между словами. В основном они используются в задачах тематической классификации, определение парафраза и извлечение отношений между словами. Но иногда, таких зависимостей недостаточно. Например, для того чтобы система интеллектуального ввода могла предсказать следующее слово, необходимо запоминать более длительные зависимости. Эту проблему решили с помощью РНС, в которой выходные данные с предыдущего шага подаются как входные данные для текущего шага. Чаще применяются именно РНС, так как они дают более точные результаты [1]. Существенным недостатком данного подхода является то, что при работе с длинными последовательностями обучение нейронных сетей требует больших вычислительных мощностей и большого объема данных. Еще одним недостатком является то, что обученная сеть, решая определенную задачу на одном корпусе текстов, может не справиться с той же самой задачей на другом корпусе.

Помимо нейронных сетей для анализа текста так же применяют корпусную лингвистику. Корпусная лингвистика – это изучение и анализ данных, полученных из корпуса. В рамках данного подхода под «корпусом» понимают «унифицированный, структурированный и размеченный массив языковых (речевых) данных в электронном виде, предназначенный для определенных гуманитарных изысканий» [2]. Он позволяет производить поиск лексических и грамматических языковых единиц и отбор слов, словоформ, грамматических категорий и словосочетаний. В качестве модели языка в корпусной лингвистике принимается сам язык. Основная задачи корпусного лингвиста – автоматическое извлечение данных, обработка и анализ информации, а также её интерпретация, вывод языковых закономерностей [3]. Основным преимуществом применения корпусного анализа текста является объективность исследования: такие функции как подбор, разметка, анализ текстов и выявление соответствий выполняются автоматически. Наиболее востребованными инструментами современной компьютерной лингвистики являются NLTK, Pattern, PyMorphy2 [4].

# Концепция алгоритма

В данной работе для анализа текста была выбран подход, базирующийся на корпусной лингвистике. Решение поставленной задачи будет происходить поэтапно с использованием языка программирования Python. Первым этапом анализа текста является его морфологический разбор. Существует несколько морфологических анализаторов русского языка, для данной работы был выбран PyMorphy2, работающий с морфологическим словарем OpenCorpora (http://opencorpora.org/) [5]. Преимущества словаря OpenCorpora в его открытости, но есть существенный недостаток: статистическая разреженности, ввиду чего определение морфологии некоторых слов не всегда корректно.

В процессе морфологического синтеза, по исходной словоформе и тегам выполняется поиск нормальной формы слова, а затем перебор всех возможных пар ⟨окончание, теги⟩ в найденной лексеме, пока не будет найдена пара с заданными морфологическими тегами. После этого от нормальной формы отсекается её окончание, а найденное окончание приписывается к полученной псевдооснове[6].

При разборе текста каждому слову присваиваются тэги. Один из тэгов «Score» показывает вероятность корректности разбора, выводится наиболее вероятный вариант. Функционал данной библиотеки позволяет приводить слово к нормальной форме и возвращать грамматическую информацию о слове. С помощью PyMorphy2 будут выделены ключевые слова, такие как имена героев произведения, названия населенных пунктов, слова, обозначающие перемещения персонажа, например, «приехал», «уехал» и т.д. Также необходимо будет учесть порядок посещаемых мест.

Далее необходимо решить задачу поиска имен и населенных пунктов. Для решения задач распознавания и извлечения информации из текстов используются три главных подхода: основанный на правилах (инженерный), основанный на машинном обучении и гибридный. Гибридный метод наиболее распространен. Инженерный подход опирается на тот факт, что извлекаемая информация употребляется в рамках определённых языковых конструкций. Например, название города пишется с большой буквы и нередко предваряется словами город, гор. или г. Подобная лингвистическая информация обычно вручную описывается в виде формальных шаблонов распознаваемых конструкций и правил их обработки. Затем правила применяются к анализируемому тексту: в нем ищутся описанные шаблонами фрагменты, из которых извлекается искомая информация. В рамках подхода, основанного на машинном обучении, применяются методы обучения с учителем, методы без учителя, методы частичного обучения с учителем. Чаще всего применяется обучение с учителем, которое подразумевает построение математической и программной модели, которая умеет отличать искомые данные от всех остальных. Построение такого машинного классификатора (т. е. обучение модели) происходит на специально размеченном вручную текстовом корпусе (обучающей выборке), в котором значимым объектам, их атрибутам, отношениям, фактам приписаны соответствующие метки. Метки кодируют признаки для распознавания этих данных.

По сути, обучение модели заключается в выявлении на основе частных данных, вошедших в обучающую выборку, общих закономерностей и зависимостей, которые присущи реальным данным. После обучения полученный классификатор применяется к текстам, при этом каждому извлечённому слову или словосочетанию может ставится в соответствие вероятность того, являются ли они искомыми данными или нет [6].

В работе применяется метод машинного обучения с учителем при помощи библиотеки PyMorphy2. Необходимо произвести разбор каждого слова в тексте и определить, есть ли тэг «Name» у этих слов. Для выделения населенных пунктов понадобилась библиотека для извлечения структурированной информации из текстов на русском языке «natasha». Функционал «natasha» содержит правила для извлечения имён, адресов, дат и сумм денег. Есть ещё правила для названий организаций и географических объектов, но у них не очень высокое качество. Слова, обозначающие перемещения были введены вручную и переведены в нормальную форму.

Следующий этап состоял в определении координат географических точек. Для этого была использована библиотека geocoder. Данная библиотека позволяет определять широту и долготу по наименованию местности (реальной).

На следующем этапе, воспользовавшись библиотекой folium, на карте мира выделены местности, которые упоминаются в литературном произведении. Библиотека folium – инструмент для визуализации геоданных или других данных, которые используют местоположение и координаты. Для построения передвижений персонажей нужно найти ключевое слово, обозначающее передвижение персонажа, и проверить несколько слов вправо и влево от этого слова. Если в этом диапазоне есть слова из списка выделенных персонажей и слова из списка географических объектов, то точка на карте передвижения данного персонажа найдена. Последний этапом работы было векторное отображение на карте перемещений персонажей от одной локации к другой.

# Реализация алгоритма

В данной курсовой работе мы анализировали произведение французского писателя [Жюля Верна](https://ru.wikipedia.org/wiki/%D0%92%D0%B5%D1%80%D0%BD,_%D0%96%D1%8E%D0%BB%D1%8C) «Вокруг света за 80 дней», повествующее о путешествии эксцентричного и флегматичного англичанина Филеаса Фогга и его слуги-француза Жана Паспарту вокруг света, предпринятом в результате одного пари. Это произведение выбрано неслучайно, в нем присутствует большое количество локаций, что позволит наилучшим образом продемонстрировать возможности представленного алгоритма. Визуализация перемещений была проведена для главного героя - Филеаса Фогга.

Приведенный ранее алгоритм был полностью реализован: исходный текст произведения был разбит на слова, в нем были выделены города, которые посещал главный герой, далее они были определены их координаты и отражены на карте согласно последовательности их посещения. Алгоритм смог выделить следующие города: Лондон, Аден, Индия, Калькутта, Гонконг, Шанхай, Иокогама, Нью-Йорк и Ливерпуль. Результат работы программы представлены на рис.1. Последовательность цифр на карте соответствует последовательности перемещений героя.

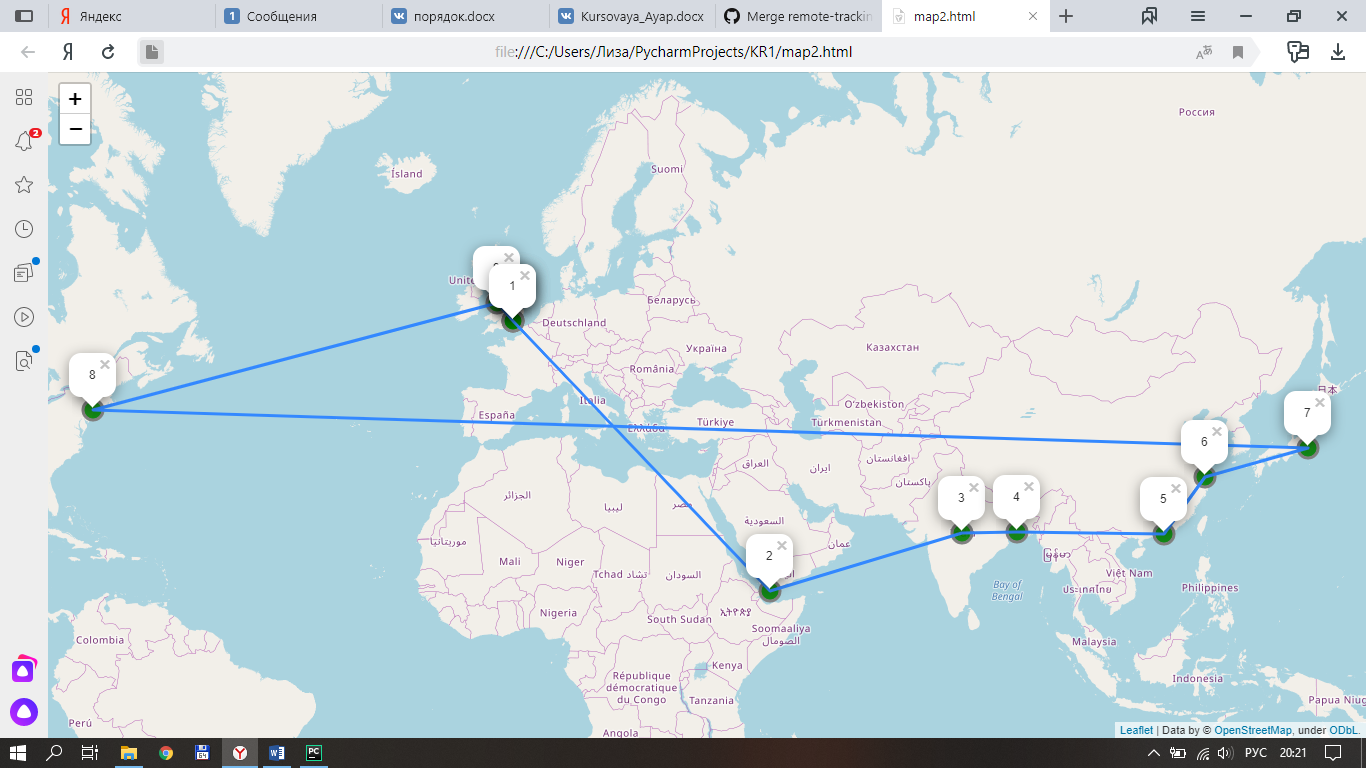


Рис.1. Карта перемещения [Филеаса Фогга](https://ru.wikipedia.org/wiki/%D0%A4%D0%B8%D0%BB%D0%B5%D0%B0%D1%81_%D0%A4%D0%BE%D0%B3%D0%B3) по алгоритму

Полученные данные практически точно описывают перемещение персонажа. На рис.2. представлен маршрут персонажа согласно роману. Код программы представлен в Приложении 1.



Рис.2. Карта перемещения [Филеаса Фогга](https://ru.wikipedia.org/wiki/%D0%A4%D0%B8%D0%BB%D0%B5%D0%B0%D1%81_%D0%A4%D0%BE%D0%B3%D0%B3) по роману

# Выводы

В работе была реализована программа для анализа текста литературного произведения, основанная на корпусно-лингвистическом подходе, с помощью которого можно составить карту перемещений персонажей литературного произведения на известной местности. Применение данного подхода позволит эффективно обработать текст, провести необходимый статистический анализ и в полном объеме представить полученные результаты.

В ходе работы было проанализировано произведение [Жюля Верна](https://ru.wikipedia.org/wiki/%D0%92%D0%B5%D1%80%D0%BD,_%D0%96%D1%8E%D0%BB%D1%8C) «Вокруг света за 80 дней». Был построен маршрут перемещения главного героя романа - [Филеаса Фогга](https://ru.wikipedia.org/wiki/%D0%A4%D0%B8%D0%BB%D0%B5%D0%B0%D1%81_%D0%A4%D0%BE%D0%B3%D0%B3). Полученные результаты работы алгоритма практически полностью совпадают с действием романа.

# Список использованной литературы

1. M. Usama et al., Equipping recurrent neural network with CNN-style attention mechanisms for sentiment analysis of network reviews // Computer Communications. – 2019. – vol. 148, – P. 98–106.

2. Захаров В.П. Поисковые системы Интернета как инструмент лингвистических исследований / В.П. Захаров // Русский язык в Интернете: Сб. статей. – Казань: Отечество, 2003. – C. 48–59.

3. Инструменты корпусной лингвистики / А. М. Амиева, В. В. Филимонов, А. П. Сергеев [и др.] // 2я Международная конференция студентов, аспирантов и молодых ученых «Информационные технологии, телекоммуникации и системы управления» – сборник докладов. – Екатеринбург: [УрФУ], 2016. – С. 251-260.

4. Райскина В.А. Применение современных методов корпусной лингвистики при анализе текста (на примере корпуса BFM) // Молодежный научный форум: Гуманитарные науки: электр. сб. ст. по мат. XXV междунар. студ. науч.-практ. конф. № 6(24). URL:[https://nauchforum.ru/archive/MNF\_humanities/6(24).pd](https://nauchforum.ru/archive/MNF_humanities/6(24).pdf)f

5. Паничева П.В., Протопопова Е.В., Митрофанова О.А., Мирзагитова А.Р. Разработка лингвистического комплекса для морфологического анализа русскоязычных корпусов текстов на основе Pymorphy и NLTK // Труды международной конференции «Корпусная лингвистика – 2015». СПб., 2015. – С. 361-373.

6. Автоматическая обработка текстов на естественном языке и анализ данных. Большакова Е.И., Воронцов К.В., Ефремова Н.Э., Клышинский Э.С., Лукашевич Н.В., Сапин А.С. // М.: Изд-во НИУ ВШЭ, 2017. – 269с.

# Приложение 1

Код программы

import pymorphy2  
import geocoder  
import natasha  
from collections import OrderedDict  
  
from folium import Popup  
  
morph = pymorphy2.MorphAnalyzer()  
#\_\_\_\_\_\_\_\_\_передвижение  
with open('2.txt') as movement: # ключевые слова передвижения  
 lst2 = []  
 for line in movement:  
 lst2 = line.split()  
#print(lst2)  
phileasfogg = {}  
k=0  
with open('Верн Жюль. Вокруг света в восемьдесят дней - royallib.com.txt') as f:  
 ls = list() # запишем весь текст разбитый на слова  
 for line in f:  
 lst = line.split()  
 words = [] # слова из одной строки  
 for word in lst:  
 p = morph.parse(word)[0] # делаем разбор  
 words.append(p.normal\_form)  
 ls += words  
for i in range(len(ls)-1):  
 for move in lst2:  
 if ls[i] == move:  
 ls1 = ls[i-25:i+25]  
 p1 = False  
 p2 = False  
 for j in range(len(ls1)-1):  
 p = morph.parse(ls1[j])[0]  
 if ('Name' in p.tag) and ((p.normal\_form == 'филеасу') or (p.normal\_form == 'фогга')):  
 p1 = True  
 elif ('Geox' in p.tag):  
 p2 = True  
 g = [ls1[j]]  
 if p1 == True and p2 == True:  
 phileasfogg[k] = g  
 k += 1  
location1= []  
from geopy.geocoders import Nominatim  
for key in phileasfogg.keys():  
 if key != 0 or key != 1:  
 try:  
 phileasfogg[key]  
 geolocator = Nominatim(user\_agent="specify\_your\_app\_name\_here")  
 location = geolocator.geocode(phileasfogg[key])  
 location1 += [list([location.latitude, location.longitude])]  
 phileasfogg[key].append([location.latitude, location.longitude])  
 except:  
 phileasfogg[key] = ['no']  
 else:  
 phileasfogg[key] = ['No']  
for key in phileasfogg.keys():  
 print(phileasfogg[key])  
location1.insert(0,[51.5073219, -0.1276474]) # координаты лондона  
location1.append([51.5073219, -0.1276474])  
# print(location1)  
location1.remove([22.34921555, 114.185797810004]) # "ошибка" алгоритма'''  
location12=[] # убрать повторения для корректной нумерации  
for i in location1:  
 if i not in location12:  
 location12.append(i)  
location12.insert(0,[51.5073219, -0.1276474]) # координаты лондона  
location12.append([51.5073219, -0.1276474])  
#отображение на карте  
import folium  
i=0  
map1 = folium.Map(location=[51.5073219, -0.1276474], popup = str(i), zoom\_start = 5)  
for coordinates in location12:  
 if i==10:  
 i=1  
 folium.CircleMarker(location=coordinates, popup=Popup(str(i), parse\_html=True, show=True), tooltip=str(i), fill\_color="green", color="gray", fill\_opacity = 0.9).add\_to(map1)  
 i=i+1  
folium.PolyLine([location12]).add\_to(map1)  
map1.save("map2.html")