MIPT
MOSCOW INSTITUTE OF PHYSICS AND TECHNOLOGY

АТП кафедра Алгоритмов и Технологий Программирования

НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ

# Автоматическая обработка текстов
# Морфологический анализ

Лекция 2

Емельянов А. А.
login-const@mail.ru

2018

https://github.com/king-menin/nlp-course

# Что такое морфология

- **Морфоло́гия**[1] (от др.-греч. μορφή — «форма» и λόγος — «слово, учение») — раздел грамматики, основными объектами которого являются слова естественных языков, их значимые части и морфологические признаки. В задачи морфологии, таким образом, входит определение слова как особого языкового объекта и описание его внутренней структуры.

- **Морфология**[2] — система форм изменения слов в каком-л. языке, а также раздел грамматики, изучающий формы слов.

1. Википедия. URL: https://ru.wikipedia.org/wiki/%D0%9C%D0%BE%D1%80%D1%84%D0%BE%D0%BB%D0%BE%D0%B3%D0%B8%D1%8F_(%D0%BB%D0%B8%D0%BD%D0%B3%D0%B2%D0%B8%D1%81%D1%82%D0%B8%D0%BA%D0%B0)
2. Малый Академический Словарь. URL: http://rus-yaz.niv.ru/doc/small-academic-vocabulary/fc/slovar-204-22.htm#zag-29544

# Морфологический анализ

**Основные морфологического анализа**:

- **Разбор слова**
  - Лемматизация – определение нормальной формы слова (леммы)
  - Определение грамматических характеристик слова (POS-tagging, частеречная разметка)
  - Стемминг – определение (псевдо)основны слова (стема)
- **Синтез слова** — генерация слова по заданным грамматическим характеристикам

# Применение морфологического анализа

- **Для классификации / кластеризации** для отбора признаков
  - **Лемматизация и стемминг** помогают сократить количество признаков (одно слово – один признак)
  - **Фильтрация** по частям речи тоже помогает сократить количество признаков
  - **Извлечение групп** [англ. chunking] (именных групп, глагольных групп) помогает добавить “умные” признаки
- Для более **сложных задач** обработки текста и речи в качестве предобработки:
  - **Машинный перевод**
  - **Распознавание и генерация речи**
  - **Поиск**

# Части речи и их грамматические характеристики

- По документации MyStem[1]
- Общая документация: universaldependencies.org[2]

| | | | |
|---|---|---|---|
| A | прилагательное | падеж, число, форма, степень сравнения, род | горячий, холодный |
| ADV | наречие | | кисло, сладко |
| ADVPRO | местоименное наречие | | почему, поэтому |
| ANUM | числительное-прилагательное | падеж, число, род | первый, третий |
| APRO | местоимение-прилагательное | падеж, число, род | мой, твой |
| COMP | часть композита | | |
| CONJ | союз | | и, но |
| INTJ | междометие | | ах, ну |
| NUM | числительное | падеж | двадцать, пять |
| PART | частица | | бы, же |
| PR | предлог | | в, на |
| S | существительное | род, число, падеж, одушевленность | гусь, топор |
| SPRO | местоимение-существительное | лицо, число, падеж | ты, вы |
| V | глагод | лицо, число, время, вид, репрезентация, залог, переходность | идти, смотреть |

1. Mystem: URL: https://tech.yandex.ru/mystem/doc/grammemes-values-docpage/
2. Общая документация http://universaldependencies.org/

# Стемминг

- Слова состоят из морфем: word = stem + affixes. **Стемминг** позволяет отбросить аффиксы (чаще всего – только суффиксы).
  - павлиний, павлиньи, павлиньим ==⇒ павлин
  - пакет, пакетом, пакеты ==⇒ пакет
- Проблемы: **морфологическая неоднозначность**
  - Существительное или глагол: стали, стекло, течь, белила, падали
  - Прилагательное или существительное: мороженое, простой
  - Существительное или существительное: черепах Новые слова

# Стемминг. Алгоритм Портера

**Алгоритм Портера**

- Алгоритм Портера состоит из 5 циклов команд, на каждом цикле – операция удаления / замены суффикса. Возможны вероятностные расширения алгоритма.
- Широко используется (использовался) в информационном поиске.
- Ошибки:
  - белый, белка, белье ==⇒ бел
  - трудность ==⇒ трудност, трудный ==⇒ труд
  - быстрый, быстрее ==⇒ быст, побыстрее ==⇒ побыст

1. Russian stemming algorithm. URL: http://snowball.tartarus.org/algorithms/russian/stemmer.html
2. The Porter Stemming Algorithm (en). URL: https://tartarus.org/martin/PorterStemmer/

# Стемминг. Использование

In [59]:
```
from nltk.stem.snowball import RussianStemmer

stemmer = RussianStemmer()
words = ['распределение', 'приставить', 'сделала', 'словообразование']
for w in words:
    stem = stemmer.stem(w)
    print(stem)
```

```
распределен
пристав
сдела
словообразован
```

# Лемматизация. Использование

In [56]:
```
sent = 'Действительно, на его лице не отражалось никаких чувств – ни проблеска сочувствия не было на нем, а ведь боль просто невыносима'
```

In [9]:
```
sent = 'У страха глаза велики .'
```

In [10]:
```
from pymorphy2 import MorphAnalyzer

m = MorphAnalyzer()
lemmas1 = [m.parse(word)[0].normal_form for word in sent.split()]
print(' '.join(lemmas1))
```

```
у страх глаз велик
```

In [11]:
```
from pymystem3 import Mystem

m = Mystem()
lemmas2 = m.lemmatize(sent)
print(''.join(lemmas2))
```

```
у страх глаз большой
```

1. pymorphy2. https://pymorphy2.readthedocs.io/en/latest/
2. Mystem. URL: https://tech.yandex.ru/mystem/

# Sequence labeling

- Требуется каждому слову в предложении приписать ту или иную метку. Примеры задач:
  - part-of-speech tagging (определение частей речи)
  - named entity recognition (извлечение именованных сущностей)
  - semantic role labeling (определение семантических ролей)

# Sequence labeling. Подходы

- **Модель на правилах**
  - Поиск в словаре

# Sequence labeling. Подходы

- **Модель на правилах**
  - Поиск в словаре

- **Отдельные классификаторы** для каждого слова

# Sequence labeling. Подходы

- **Модель на правилах**
  - Поиск в словаре
- **Отдельные классификаторы** для каждого слова
- **Скрытая марковская модель** (HMM)

# Sequence labeling. Подходы

- **Модель на правилах**
  - Поиск в словаре
- **Отдельные классификаторы** для каждого слова
- **Скрытая марковская модель** (HMM)
- **Марковская модель максимальной энтропии** (MEMM)

# Sequence labeling. Подходы

- **Модель на правилах**
  - Поиск в словаре
- **Отдельные классификаторы** для каждого слова
- **Скрытая марковская модель** (HMM)
- **Марковская модель максимальной энтропии** (MEMM)
- **Conditional random fields** (CRF)

# Sequence labeling. Подходы

- **Модель на правилах**
  - Поиск в словаре
- **Отдельные классификаторы** для каждого слова
- **Скрытая марковская модель** (HMM)
- **Марковская модель максимальной энтропии** (MEMM)
- **Conditional random fields** (CRF)
- **Нейронные сети**

# Отдельные классификаторы: наивный Байес

- По каким признакам можно определить часть речи слова?
  - Первое/последнее слово в предложении
  - Предыдущее/следующее слово
  - Есть ли дефис?
  - Префиксы/Суффиксы

# Отдельные классификаторы: наивный Байес

$$argmax_{tag} p(tag|features) =$$

$$argmax_{tag} p(features|tag) p(tag) =$$

$$argmax_{tag} \prod_{i=1}^{n} p(f_i|tag) p(tag)$$

- Здесь предполагается, что свойства независимы.

# Отдельные классификаторы

- Наивный Байес

# Отдельные классификаторы

- Наивный Байес
- Линейные модели
  - Линейная регрессия
  - SVM

1. A. Berger, S. Della Pietr, V. Della Pietra. A Maximum Entropy Approach to Natural Language Processing. Computational Linguistics, no. 22, 39–71, 1996.

# Отдельные классификаторы

- Наивный Байес
- Линейные модели
  - Линейная регрессия
  - SVM
- Логистическая регрессия
  - Эта модель также называется моделью максимальной энтропии [1].

1. A. Berger, S. Della Pietr, V. Della Pietra. A Maximum Entropy Approach to Natural Language Processing. Computational Linguistics, no. 22, 39–71, 1996.

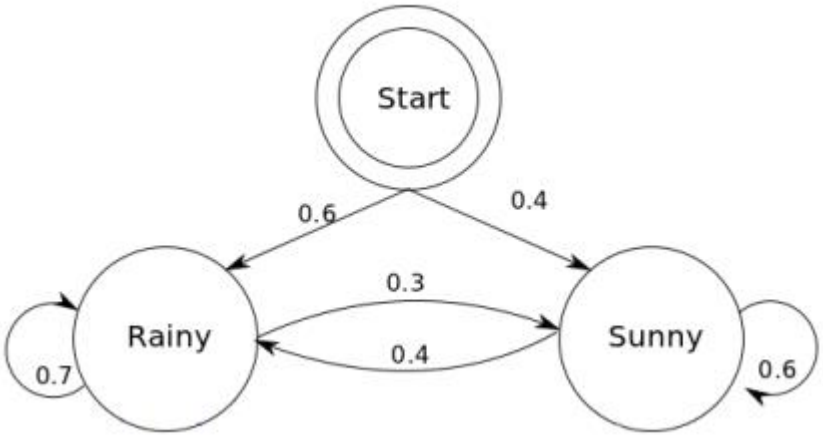
# Отдельные классификаторы

- Наивный Байес
- Линейные модели
  - Линейная регрессия
  - SVM
- Логистическая регрессия
  - Эта модель также называется моделью максимальной энтропии [1].
- Решающие деревья
  - Случайный лес
  - Бустинг

1. A. Berger, S. Della Pietr, V. Della Pietra. A Maximum Entropy Approach to Natural Language Processing. Computational Linguistics, no. 22, 39–71, 1996.

# Марковская модель

- задается следующим набором:
  - $Q = q_1, q_2, \ldots, q_N$: множество состояний;
  - q0: начальное состояние;
  - $A = (a_{ij})$: $(N + 1) \times (N + 1)$ - матрица переходных вероятностей;



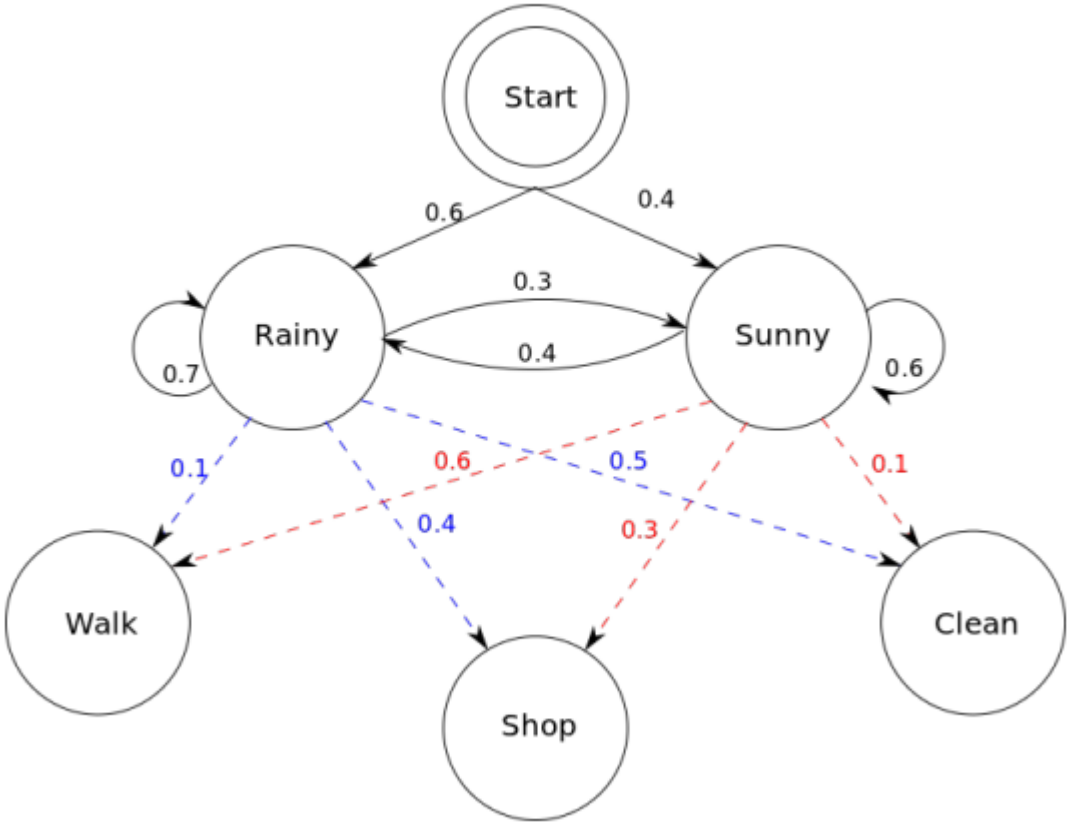
# Использование языковой модели для генерации псевдослучайного текста

In [42]:
```
import markovify
with open("positive.txt", "r", encoding="utf-8") as file:
    positive_plain = file.read()
text_model = markovify.Text(positive_plain)
```

In [44]:
```
text = text_model.make_short_sentence(140)
```

```
@Dimas_writter И что будет после этого они не захотят вступать в еврос
оюз 100% ахаххаха.
```

# Скрытая марковскоя модель



# Скрытая марковскоя модель

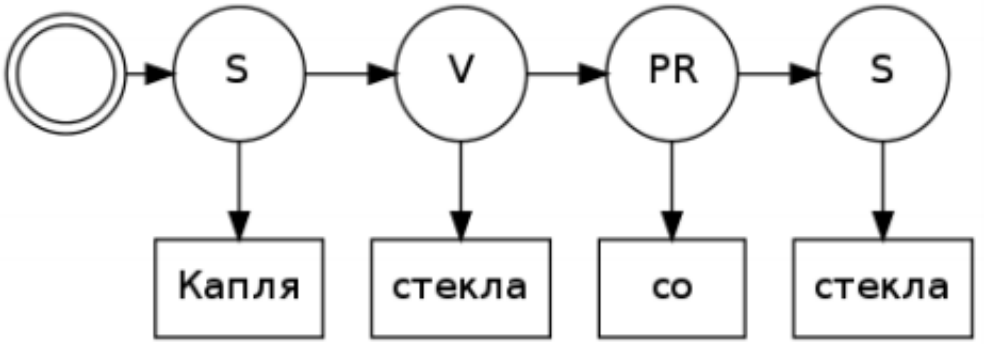
- $Q = q_1, q_2, \ldots, q_N$: множество состояний;
- q0: начальное состояние;
- $A = (a_{ij})$: $(N + 1) \times (N + 1)$ - матрица переходных вероятностей;
- $O = o_1, o_2, \ldots, o_N$: последовательность наблюдаемых;
- $B = b_i(o_t)$: набор выходных вероятностей.

- Наблюдаем внешние события, но не внутреннее состояние модели.

# Скрытая марковскоя модель

- В наших задачах скрытыми состояниями будут последовательности тегов, а наблюдаемыми — слова предложения.

- То есть, появление очередного слова текста будет зависеть от текущего морфологического тега, а появление очередного тега — от предыдущих тегов.



# Задачи, связанные с HMM

- **Оценка.**
  - Найти вероятность данной последовательности наблюдений.
  - Решается с помощью **алгоритма Forward-Backward**.
- **Декодирование**.
  - Найти наиболее вероятную последовательность тегов.
  - Решается с помощью **алгоритма Витерби**.
- **Обучение.**
  - Подобрать параметры модели для данного выхода.
  - Решается с помощью **алгоритма Баума-Велша**.

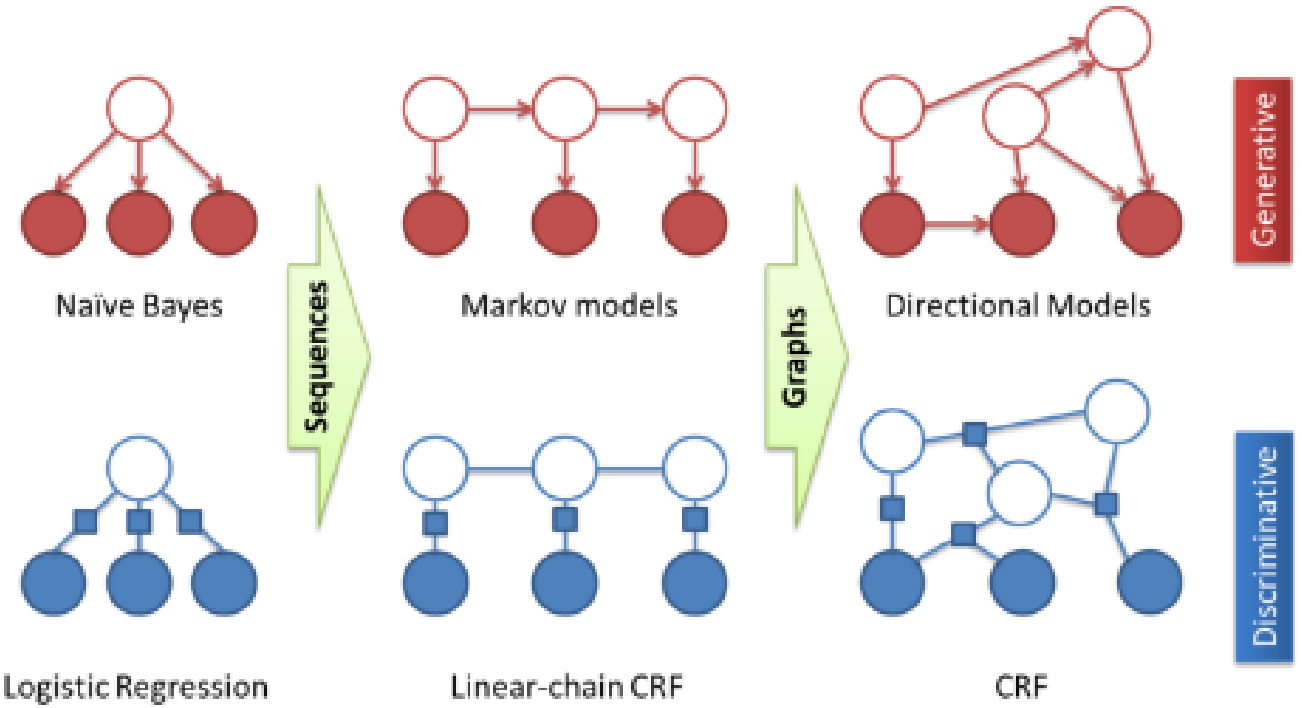
# Использование HMM для POS-tagging

In [42]:

```
import nltk
from nltk.corpus import treebank

# Train data - pretagged
train_data = treebank.tagged_sents()[:3000]

print train_data[0]

# Import HMM module
from nltk.tag import hmm

# Setup a trainer with default(None) values
# And train with the data
trainer = hmm.HiddenMarkovModelTrainer()
tagger = trainer.train_supervised(train_data)
```

nltk.hmm

# Условные случайные поля (CRF)




Adapted from C. Sutton, A. McCallum, "An Introduction to Conditional Random Fields", ArXiv, November 2010

# Использование CRF для POS-tagging

```
In [13]: train, test = sentences[:-100], sentences[-100:]

In [14]: from nltk.tag import CRFTagger

In [15]: ct = CRFTagger()

In [16]: ct.train(train ,'model.crf.tagger')

In [17]: ct.evaluate(test)
Out[17]: 0.9566528458349038
```

# Морфологический анализ в python

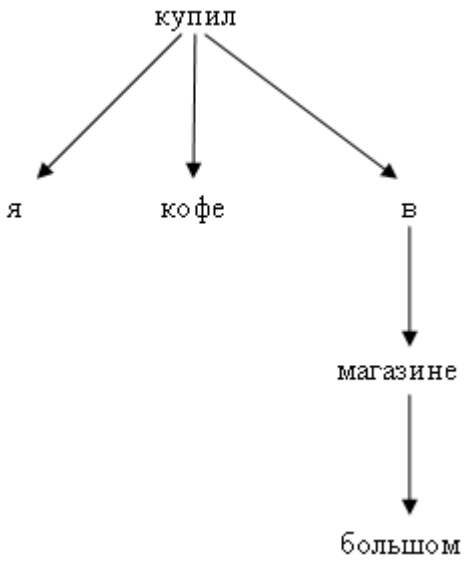
In [21]:
```
word = 'ГАИ'
```

In [22]:
```
m = MorphAnalyzer()
m.parse(word)
```

Out[22]:
```
[Parse(word='гаи', tag=OpencorporaTag('NOUN,anim,masc,Name plur,nomn'), normal_form
='гай', score=0.1111111111111111, methods_stack=((<DictionaryAnalyzer>, 'гаи', 41,
6),)),
 Parse(word='гаи', tag=OpencorporaTag('NOUN,inan,masc plur,nomn'), normal_form='гай',
score=0.1111111111111111, methods_stack=((<DictionaryAnalyzer>, 'гаи', 495, 7),)),
 Parse(word='гаи', tag=OpencorporaTag('NOUN,inan,masc plur,accs'), normal_form='гай',
score=0.1111111111111111, methods_stack=((<DictionaryAnalyzer>, 'гаи', 495, 10),)),
 Parse(word='гаи', tag=OpencorporaTag('NOUN,inan,femn,Sgtm,Fixd,Abbr,Orgn sing,nom
n'), normal_form='гаи', score=0.1111111111111111, methods_stack=((<DictionaryAnalyzer
>, 'гаи', 645, 0),)),
 Parse(word='гаи', tag=OpencorporaTag('NOUN,inan,femn,Sgtm,Fixd,Abbr,Orgn sing,gen
t'), normal_form='гаи', score=0.1111111111111111, methods_stack=((<DictionaryAnalyzer
>, 'гаи', 645, 1),)),
 Parse(word='гаи', tag=OpencorporaTag('NOUN,inan,femn,Sgtm,Fixd,Abbr,Orgn sing,dat
v'), normal_form='гаи', score=0.1111111111111111, methods_stack=((<DictionaryAnalyzer
>, 'гаи', 645, 2),)),
 Parse(word='гаи', tag=OpencorporaTag('NOUN,inan,femn,Sgtm,Fixd,Abbr,Orgn sing,acc
s'), normal_form='гаи', score=0.1111111111111111, methods_stack=((<DictionaryAnalyzer
>, 'гаи', 645, 3),)),
 Parse(word='гаи', tag=OpencorporaTag('NOUN,inan,femn,Sgtm,Fixd,Abbr,Orgn sing,abl
t'), normal_form='гаи', score=0.1111111111111111, methods_stack=((<DictionaryAnalyzer
>, 'гаи', 645, 4),)),
 Parse(word='гаи', tag=OpencorporaTag('NOUN,inan,femn,Sgtm,Fixd,Abbr,Orgn sing,loc
t'), normal_form='гаи', score=0.1111111111111111, methods_stack=((<DictionaryAnalyzer
```

# Синтаксический анализ

- Я купил кофе в большом магазине




- Все слова в предложении связаны отношением типа "хозяин-слуга", имеющим различные подтипы
- Узел дерева – слово в предложении
- Дуга дерева – отношение подчинения

# Синтаксический анализ: SyntaxNet

- SyntaxNet – архитектура синтаксического парсера. Доступны обученные модели для более чем 40 языков, в том числе, для русского.
- D. Chen and C. D. Manning. A Fast and Accurate Dependency Parser using Neural Networks. EMNLP. 2014.

# Синтаксический анализ: обработка conll файлов

In [91]:

```
from nltk import DependencyGraph
import codecs

processed_sentences = []
sentence = []
for line in codecs.open('data.conll', 'r', 'utf-8'):
    if len(line) == 1:
        processed_sentences.append(sentence)
        sentence = []
    else:
        word = line.split("\t")
        sentence.append(word)

deps = []
for sentence in processed_sentences:
    s = u""
    for line in sentence:
        s += u"\t".join(line) + u'\n'
    deps.append(s)
```

# Синтаксический анализ: обработка conll файлов

- Синтаксические деревья:

In [93]:
```
for sent_dep in deps:
    graph = DependencyGraph(tree_str=sent_dep)
    tree = graph.tree()
    print(tree.pretty_print())
```


None

# Домашнее задание 1

- **Целью этого задания является знакомство морфологическим анализом, задачей извлечения сущностей из текста и регулярных выражений.**
- **Deadline (получение полных баллов):** 23:59 28.04.2018
- **Адрес:** login-const@mail.ru
- Задание состоит из двух частей:
  - Генерация текста по шаблону
  - Извлечение телефонных номеров из текста.
- **Текс условия доступен по** ссылке.

# СПАСИБО ЗА ВНИМАНИЕ