**Санкт-Петербургский политехнический университет Петра Великого**

**Институт компьютерных наук и технологий**

**Высшая школа интеллектуальных систем и суперкомпьютерных технологий**

**Р а с ч е т н а я р а б о т а № 1**

**Часть 2. Задание words**

по дисциплине «Теория вероятностей и математическая статистика»

Выполнила студент гр. 3530901/90001 \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Фам Ба Нам

(подпись)

Преподаватель \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ К.В. Никитин

(подпись)

Санкт-Петербург 2021

**Постановка задачи**

Ведущий загадывает слово. Затем в начале каждого хода ведущий сообщает некоторую информацию о слове:

- он выбирает наугад одну из букв в слове и сообщает ее без уточнения ее позиции в слове;

- он выбирает наугад одну из букв в слове и сообщает ее позицию и определенную информацию о ней (гласная/согласная/соединительная, звонкая/глухая, заглавная/строчная);

Игрок на основании информации, полученной от ведущего, строит гипотезы о том, какое это слово. Количество букв в слове в общем случае неизвестно.

1. Попытаться вначале угадать слово самостоятельно, используя как можно меньше подсказок.

2. Используя предоставленную базу слов, вычислить для каждого из слов последовательно вероятности того, что загадано именно это слово. Определить количество подсказок, начиная с которого можно с определенной уверенностью утверждать о том, какое слово загадано.

Построить распределения числа слов-кандидатов как функцию от номера подсказок. По логике количество слов-кандидатов с каждым шагом должно уменьшаться.

Загаданное слово принадлежит одному из классов:

Город России (cities\_rus.txt)

Страна мира (countries.txt)

Столица (capitals.txt)

Русское имя (names\_all.txt).

Река (rivers.txt)

Русское существительное (имя нарицательное) (russian\_nouns.txt)

В скобках выше приведены названия текстовых файлов, в которых

**Задание**

1а. После каждого опыта необходимо вычислить ряд распределения апостериорных вероятностей гипотез – какое слово загадано. Представить соответствующие результаты визуально на графике в форме изменения с течением опытов диаграмм распределений вероятностей гипотез. Сузить количество гипотез можно по аналогии с п. 3a. 9

Зам. В данной задаче количество гипотез равно количеству слов в соответствующем файле со словами.

1b. Определять после каждого опыта, какое слово имеет наибольшую вероятность. Визуализировать эволюцию изменения наиболее вероятного слова.

1с. Построить зависимость числа превалирующих гипотез от числа проведенных опытов.

2a. Рассмотреть каждую букву в слове по отдельности и в качестве гипотез выдвигать то, что это за буква (сам символ, заглавная/строчная, гласная/согласная и т.д.). Для гипотезы о том, что за символ общее количество гипотез равно общему количеству возможных символов, для гипотез о классах буквы количество гипотез равно количеству классов.

Вычислить для каждой из букв распределения вероятностей гипотез после каждого опыта. Представить результаты визуально по аналогии с п. 1а.

2b. Определить для каждой буквы наиболее вероятную гипотезу на каждом шаге и визуализировать эволюцию этой гипотезы.

2с. Объединить результаты для всех букв, получить наиболее вероятное слово и сравнить его с полученным п.1. Провести анализ сравнения.

Зам. Возможной проблемой в данном пункте будет то, что общее количество букв не задано. Возможным способом ее решения является отбрасывание всех букв, у которых номер в слове больше, чем номер последней буквы, сообщенной ведущим.

3a. Определить приближенно частоту тех или иных сообщаемых символов. Исходя из этого можно оценить примерно, сколько каждого из символов в слове. Затем сопоставьте эту информацию с тем, что ведущий говорит о позициях букв и их типе. В результате можно сузить количество возможных слов, используемых в качестве гипотез.

3b. Составьте для каждого слова из базы теоретические профили, содержащие в себе некоторую статистику – количество символов каждого типа, позиция с учетом типа символа. Далее из п. 3а определите аналогичный экспериментальный профиль.

3c. Сопоставить теоретические профили с экспериментальным и найдите их наиболее правдоподобное соответствие в форме слова (слов). Сравнить с полученными результатами в п. 1 и 2. Провести анализ сравнения.

3d. Привести графики изменения экспериментальных профилей для различного количества опытов

**Теоретические основы**

Формула Байеса имеет следующий вид: .

1. Полученная информация на каждом шаге о загаданном шаге сужает количество гипотез. В связи с этим условная вероятность будет варьироваться в зависимости от информации, сказанной ведущим.

Вероятность будет равна 1, если назван номер буквы и ее классы.

Вероятность , если сказана просто буква.

2. Далее необходимо рассмотреть каждую букву в слове по отдельности, в качестве гипотез были взяты все возможные вариации буквы, которая может находится на какой-либо позиции. Общее количество гипотез равно m\*n, где m – мощность алфавита, n – количество букв в слове.

# Описание решения

1. Полученная информация на каждом шаге о загаданном слове сужает количество гипотез. В связи с этим условная вероятность, варьироваться в зависимости от информации, сказанной ведущим:
   1. если сказана просто буква, Р(Hi/A) будет равна количеству названных букв в слове, разделенных на количество букв в слове(гипотезе).
   2. если назван номер буквы и ее классы, то если указанная буква в гипотезе того же класса, то вероятность такого условия будет равна 1
2. Далее необходимо рассмотреть каждую букву в слове по отдельности, в качестве гипотез были взяты все возможные вариации буквы, которая может находиться на какой-либо позиции: итого их получилось n \* m, где n – количество букв в слове, m – мощность алфавита. В таком случае условная вероятность будет иметь вид:
   1. если названная буква ведущим совпадает с гипотезой, то Р(Hi/A) будет равна 1/n
   2. если названная буква ведущим не совпадает с гипотезой, то Р(Hi/A) будет равна 1/ (n \* m), т.к. вероятность того, что не смотря на букву, названную ведущим на данной позиции все равно будет стоять буква гипотезы будет равна вероятности того, что ведущий имел ввиду другой номер буквы, т.е. 1/n, и вероятности того, что на данном месте стоит именно эта буква, т.е. 1/m
   3. в случае, если названа позиция буквы и ее класс то Р(Hi/A) будет равно 1

# Реализация

## 1a Вычисление ряда распределения

В листинге 1.1 после каждого опыта необходимо вычислить ряд распределения апостериорных вероятностей гипотез – какое слово загадано. Представить соответствующие результаты визуально на графике в форме изменения с течением опытов диаграмм распределений вероятностей гипотез.

Листинг 1.1

import itertools

import codecs

from scipy.special import comb

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

import math

voiceless=['п', 'ф', 'т', 'с', 'ш', 'к','ч', 'щ', 'ц', 'х']

voiced=['б', 'в', 'д', 'з', 'ж', 'г', 'л', 'м', 'н', 'р', 'й']

vowels=['а', 'о', 'и', 'е', 'ё', 'э', 'ы', 'у', 'ю', 'я']

signs=['ь','ъ']

def letter\_info(letter):

    s=''

    if letter=='':

        return ''

    if letter in signs:

        return 's'

    if letter.isupper():

        s='u'

    else:

        s='l'

    letter\_l=letter.lower()

    if letter\_l in voiceless:

        s=s+'cs'

    elif letter\_l in voiced:

        s=s+'cd'

    elif letter\_l in vowels:

        s=s+'v'

    return s

#Это вероятность получить букву из слова

def pi\_letter(word,letter):

    return len([a for a in word if a.lower()==letter])/len(word)

def pi\_info(word,position,info):

    if position>=len(word):

        return 0

    if info!=letter\_info(word[position]):

        return 0

    else:

        return 1/len(word)

f=codecs.open('task\_1\_words.txt',encoding='utf-8')

n\_exp=int(f.readline().split(': ')[1].split(', ')[1].split(' = ')[1])

f.readline()

info\_list=[]

for i in range(n\_exp):

    dct={}

    s=f.readline().split(': ')

    if len(s)==2:

        dct['letter']=s[1][1]

    else:

        dct['position']=int(s[1])-1

        x=''

        for item in s[2].rstrip().split(' '):

            if item=='заглавная':

                x='u'

            if item=='строчная':

                x='l'

            if item=='гласная':

                x=x+'v'

            if item=='звонкая':

                x=x+'cd'

            if item=='глухая':

                x=x+'cs'

        dct['info']=x

    info\_list.append(dct)

f.close()

list\_of\_words=[]

with open('names\_all.txt ','r',encoding='utf-8') as f:

    for line in f:

        list\_of\_words.append(line.rstrip())

probs={}

for word in list\_of\_words:

    probs[word]=1/len(list\_of\_words)

possible\_words=[word for word in list\_of\_words]

step=0

max\_prob=max(probs.values())

print(probs.values())

for item in info\_list:

    step+=1

    if 'letter' in item:

        for word in possible\_words:

            probs[word]\*=pi\_letter(word,item['letter'])

    else:

        for word in possible\_words:

            probs[word]\*=pi\_info(word,item['position'],item['info'])

    s=sum(probs.values())

    for word in possible\_words:

        try:

            probs[word]/=s

        except:

            print("ngoai le")

    possible\_words=[word for word in possible\_words if probs[word]!=0]

    if max(probs.values())/max\_prob>1.5:

        print(step)

        plt.plot(list(probs.values()))

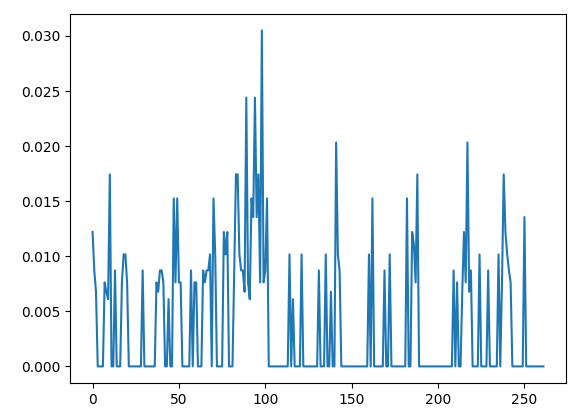
        plt.show()

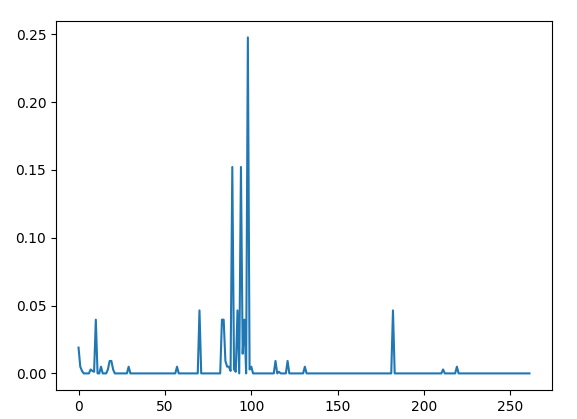
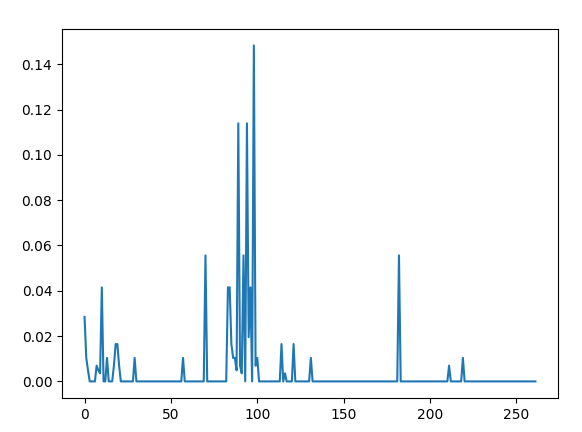
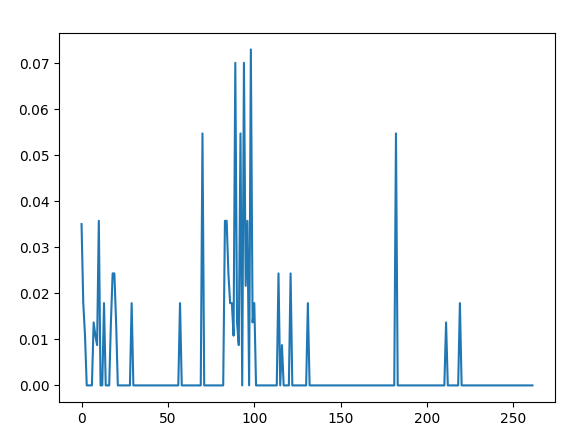
        max\_prob=max(probs.values())

    print(len(possible\_words))

    if len(possible\_words)==1:

        break





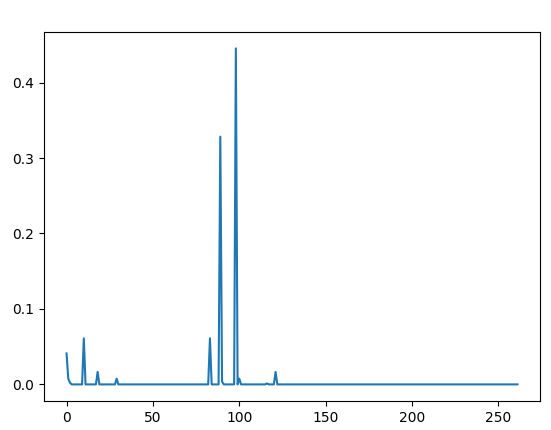


График выводится, только если достаточно сильно поменялась максимальная вероятность гипотез. Когда остается одно слово, алгоритм заканчивает работу.

## 1b Визуализация эволюции изменения наиболее вероятного слова

Определим после каждого опыта, какое слово имеет наибольшую вероятность.

**Листинг 2.1**

possibleWords = listOfWords

list\_of\_codes = []

best\_words = {'':0}

for item in infoList:

    if 'letter' in item:

        for word in possibleWords:

            probs[word] = piLetter(word, item['letter'])

    else:

        for word in possibleWords:

            probs[word] = piInfo(word, item['position'], item['info'])

    s = sum(probs.values())

    if s == 0.0 : continue

    for word in possibleWords:

        try:

            probs[word] /= s

        except:

            print("0")

    possibleWords = [word for word in possibleWords if probs[word] != 0]

    word = max(probs, key = probs.get)

    if word in best\_words:

        list\_of\_codes.append(best\_words[word])

    else:

        best\_words[word] = max(best\_words.values()) + 1

        list\_of\_codes.append(best\_words[word])

    if len(possibleWords) == 1: break

fig, ax = plt.subplots()

ax.set\_yticks(range(len(best\_words)))

ax.set\_yticklabels([['$'+key+'$' for key, value in best\_words.items() if value == i][0] for i in range(len(best\_words))])

ax.plot(list\_of\_codes)

ax.set\_title('Maximum probability hypothesis')

plt.show()

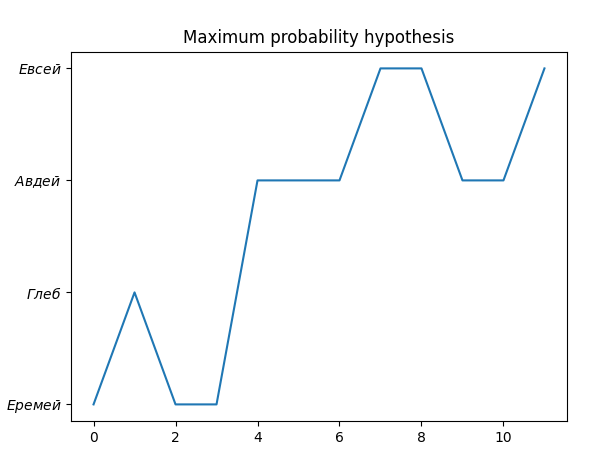


График 2.1

Построим зависимость числа превалирующих гипотез от числа проведенных опытов.

**Листинг 2.2**

possibleWords = listOfWords

list\_of\_nums = []

for item in infoList:

    if 'letter' in item:

        for word in possibleWords:

            probs[word] = piLetter(word, item['letter'])

    else:

        for word in possibleWords:

            probs[word] = piInfo(word, item['position'], item['info'])

    s = sum(probs.values())

    if s == 0.0 : continue

    for word in possibleWords:

        probs[word] /= s

    possibleWords = [word for word in possibleWords if probs[word] != 0]

    num\_of\_hyps = 0

    total\_prob = 0

    for prob in sorted(probs.values(), reverse = True):

        num\_of\_hyps += 1

        total\_prob += prob

        if total\_prob > 0.99:

            break

    list\_of\_nums.append(num\_of\_hyps)

    if len(possibleWords) == 1:

        break

fig, ax = plt.subplots()

ax.plot(list\_of\_nums)

ax.set\_yscale('log')

ax.set\_title('Number of best hypotheses')

plt.show()

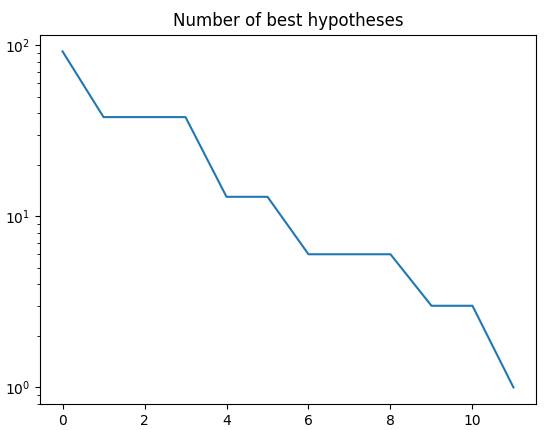


График 2.2

Будем рассматривать каждую букву в слове по отдельности и в качестве гипотез выдвигать то, что это за буква (сам символ, заглавная/строчная, гласная/согласная и т.д.). Для гипотезы о том, что за символ общее количество гипотез равно общему количеству возможных символов, для гипотез о классах буквы количество гипотез равно количеству классов.

**2а Вычислим для каждой из букв распределения вероятностей гипотез после каждого опыта. Представим результаты визуально по аналогии с пунктом 1.**

Проблема с неизвестным количеством букв решается следующим образом: Количество букв не превышает максимальную длину слова в словаре. Далее, если на каком-либо этапе оказалось, что вероятность иметь на позиции '' (то есть не иметь никакого символа) равна 1, то процедура останавливается, следующие позиции нас не интересуют.

**Листинг 2.3**

def symbolAtPosition(word, position):

    if position < len(word):

        return word[position]

    return ''

probs = {}

N = max([len(word) for word in listOfWords])

for position in range(N):

    hyps = ['а', 'б', 'в', 'г', 'д', 'е', 'ё', 'ж', 'з', 'и', 'й', 'к', 'л', 'м',

          'н', 'о', 'п', 'р', 'с', 'т', 'у', 'ф', 'х', 'ц', 'ч', 'ш', 'щ', 'ь',

          'ы', 'ъ', 'э', 'ю', 'я']

    hyps = hyps + [x.upper() for x in hyps] + ['']

    for word in listOfWords:

        probs[word]=1 / len(listOfWords)

    possibleWords = listOfWords

    prob\_letter = {}

    num\_of\_possible\_letters = 67

    for hyp in hyps:

        prob\_letter[hyp] = sum([probs[word] for word in possibleWords if symbolAtPosition(word, position) == hyp])

    hyps=[h for h in hyps if prob\_letter[h] != 0]

    step = 0

    x = [prob\_letter[let] for let in hyps]

    plt.plot(x)

    plt.title('Position ' + str(position + 1))

    plt.xticks(range(len(x)), ['$' + let + '$' for let in hyps[:-1]] + [''])

    for item in infoList:

        step += 1

        if 'letter' in item:

            for word in possibleWords:

                probs[word] = piLetter(word, item['letter'])

        else:

            for word in possibleWords:

                probs[word] = piInfo(word, item['position'], item['info'])

        s = sum(probs.values())

        if(s == 0 ) : continue

        for word in possibleWords:

            probs[word] /= s

        for hyp in hyps:

            prob\_letter[hyp] = sum([probs[word] for word in possibleWords if symbolAtPosition(word, position) == hyp])

        possibleWords = [word for word in possibleWords if probs[word] != 0]

        if len([x for x in prob\_letter.values() if x != 0]) / num\_of\_possible\_letters < 0.6:

            plt.plot([prob\_letter[let] for let in hyps])

            num\_of\_possible\_letters = len([x for x in prob\_letter.values() if x != 0])

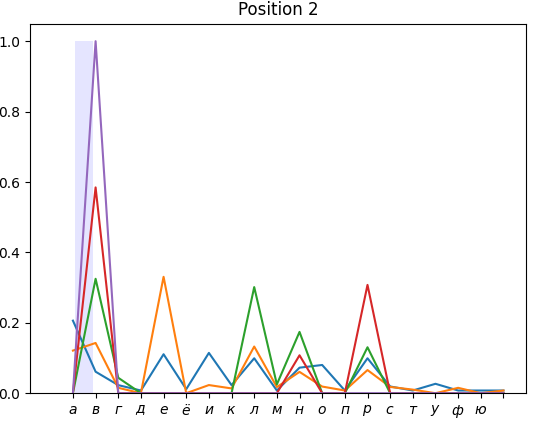
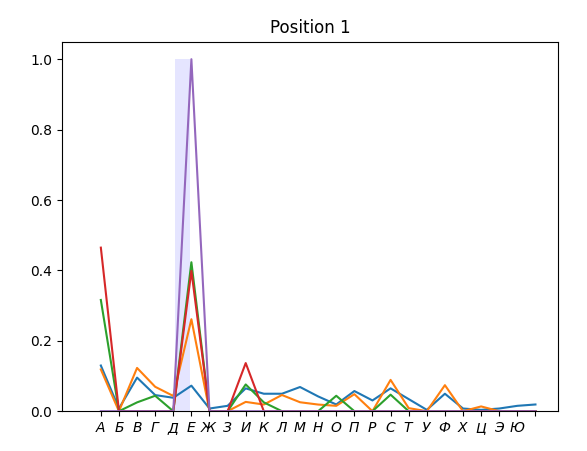
        if len(possibleWords) == 1: break

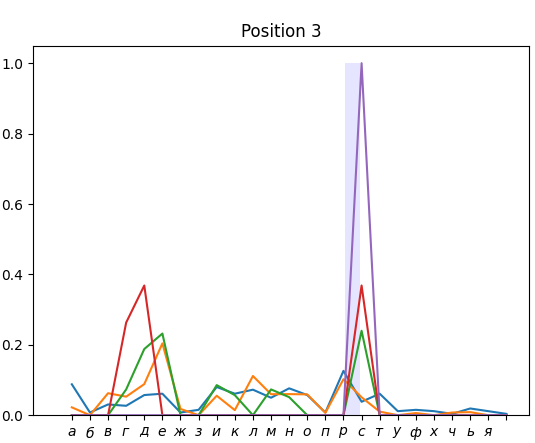
        if num\_of\_possible\_letters == 1: break

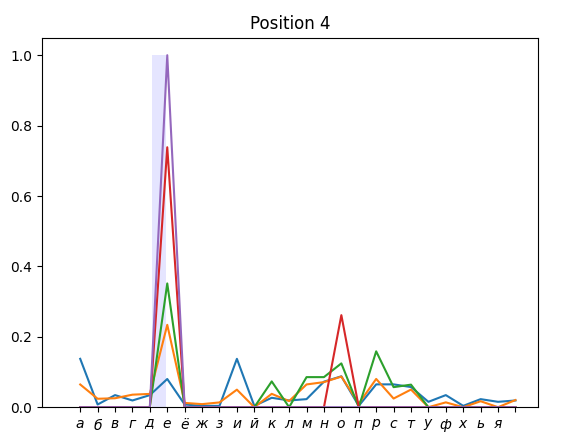
    plt.bar([k - 0.5 for k in range(len(hyps))], [prob\_letter[let] for let in hyps], alpha = 0.1, color = 'blue')

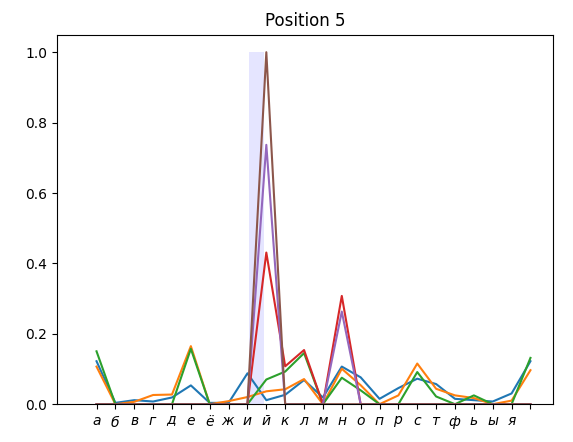
    plt.show()

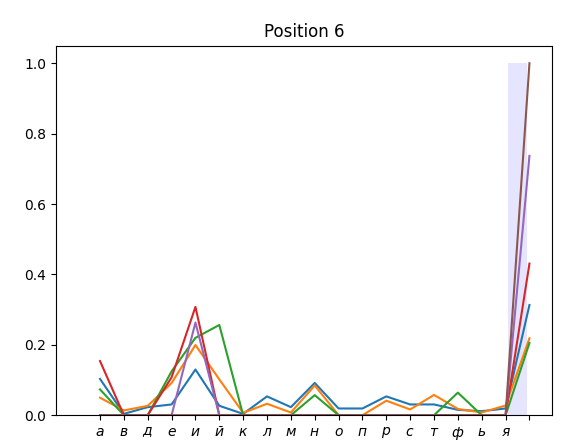
    if prob\_letter[''] == 1: break











Гипотеза о типе символа: 1) заглавная гласная (uv) 2) заглавная согласная звонкая (ucd) 3) заглавная согласная глухая (ucs)

4) строчная гласная (lv) 5) строчная согласная звонкая (lcd) 6) строчная согласная глухая (lcs) 7) знак (s) 8) нет символа ()

**Листинг 2.4**

probs = {}

hyps = ['uv', 'ucd', 'ucs', 'lv', 'lcd', 'lcs', 's', '']

for position in range(N):

    for word in listOfWords:

        probs[word] = 1 / len(listOfWords)

    possibleWords = listOfWords

    probLetterType = {}

    num\_of\_possible\_letter\_types = 8

    for hyp in hyps:

        probLetterType[hyp]=sum([probs[word] for word in possibleWords if letterInfo(symbolAtPosition(word, position)) == hyp])

    step = 0

    plt.plot([probLetterType[hyp] for hyp in hyps])

    plt.title('Position ' + str(position + 1))

    plt.xticks(range(8), hyps)

    for item in infoList:

        step += 1

        if 'letter' in item:

            for word in possibleWords:

                probs[word] = piLetter(word, item['letter'])

        else:

            for word in possibleWords:

                probs[word] = piInfo(word, item['position'], item['info'])

        s = sum(probs.values())

        if s == 0 : continue

        for word in possibleWords:

            probs[word] /= s

        for hyp in hyps:

            probLetterType[hyp] = sum([probs[word] for word in possibleWords if letterInfo(symbolAtPosition(word, position)) == hyp])

        possibleWords = [word for word in possibleWords if probs[word] != 0]

        if len([x for x in probLetterType.values() if x != 0]) < num\_of\_possible\_letter\_types:

            plt.plot([probLetterType[hyp] for hyp in hyps])

            num\_of\_possible\_letter\_types = len([x for x in probLetterType.values() if x != 0])

        if len(possibleWords) == 1: break

        if num\_of\_possible\_letter\_types == 1: break

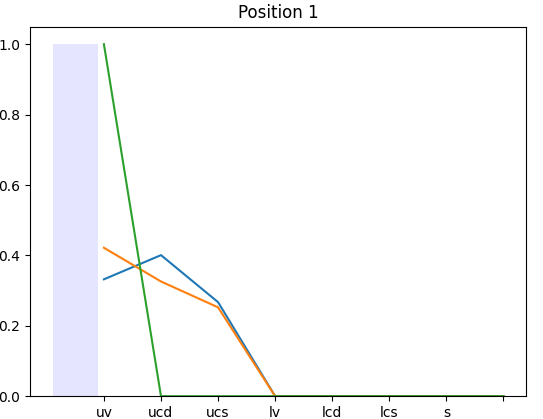
    plt.bar([k - 0.5 for k in range(8)], [probLetterType[hyp] for hyp in hyps], alpha = 0.1, color= 'blue')

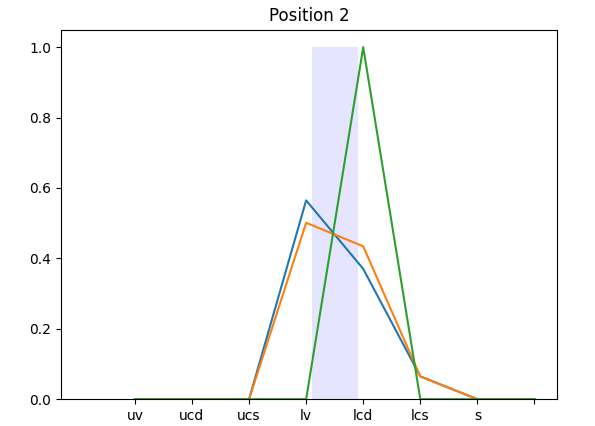
    plt.show()

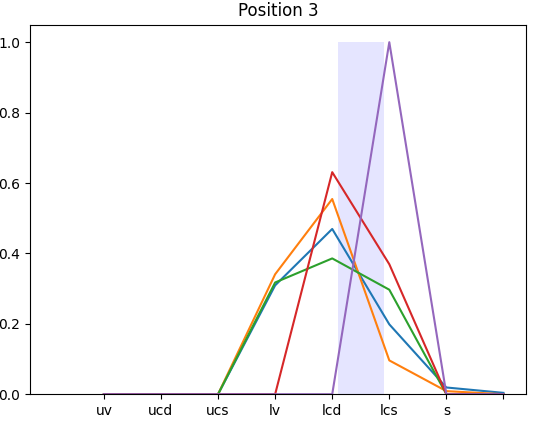
    if probLetterType[''] == 1: break

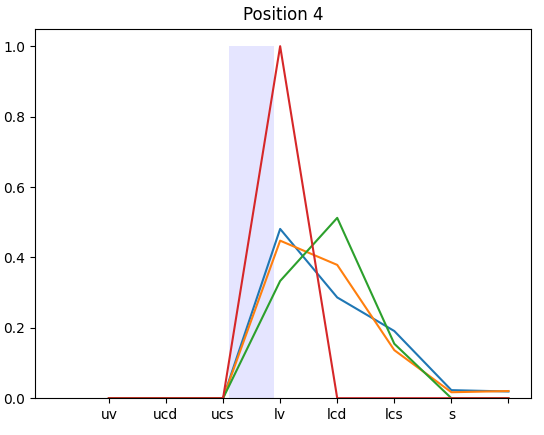
# ---2b---

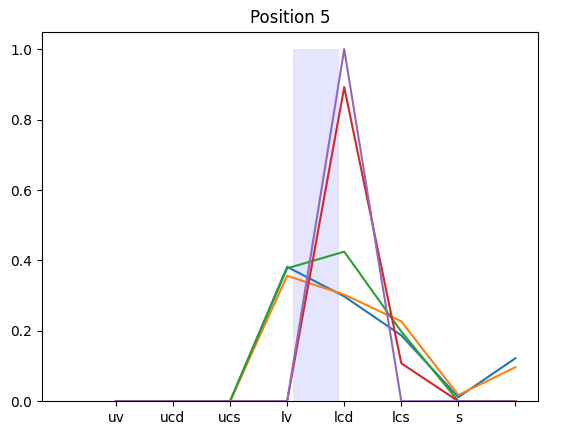
Результаты:

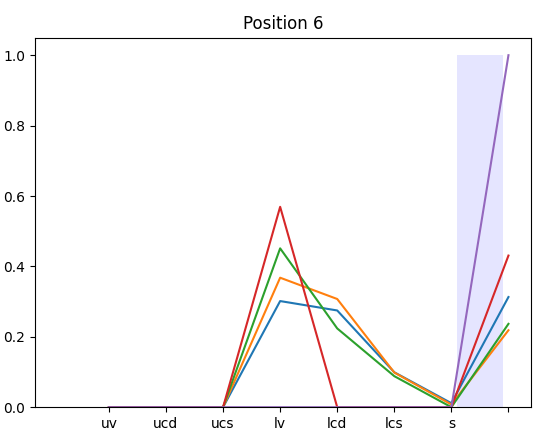












**2b Определим для каждой буквы наиболее вероятную гипотезу на каждом шаге и визуализировать эволюцию этой гипотезы.**

**Листинг 2.5**

probs = {}

for position in range(N):

    hyps = ['а', 'б', 'в', 'г', 'д', 'е', 'ё', 'ж', 'з', 'и', 'й', 'к', 'л', 'м',

      'н', 'о', 'п', 'р', 'с', 'т', 'у', 'ф', 'х', 'ц', 'ч', 'ш', 'щ', 'ь',

      'ы', 'ъ', 'э', 'ю', 'я']

    hyps = hyps + [x.upper() for x in hyps] + ['']

    for word in listOfWords:

        probs[word]= 1 / len(listOfWords)

    possibleWords = listOfWords

    prob\_letter = {}

    for hyp in hyps:

        prob\_letter[hyp] = sum([probs[word] for word in possibleWords if symbolAtPosition(word, position) == hyp])

    hyps=[h for h in hyps if prob\_letter[h] != 0]

    step = 0

    best\_hyps = []

    for item in infoList:

        step += 1

        if 'letter' in item:

            for word in possibleWords:

                probs[word] = piLetter(word, item['letter'])

        else:

            for word in possibleWords:

                probs[word] = piInfo(word, item['position'], item['info'])

        s = sum(probs.values())

        if s == 0 : continue

        for word in possibleWords:

            probs[word] /= s

        for hyp in hyps:

            prob\_letter[hyp] = sum([probs[word] for word in possibleWords if symbolAtPosition(word, position) == hyp])

        possibleWords = [word for word in possibleWords if probs[word] != 0]

        best\_hyps.append(hyps.index(max(prob\_letter, key = prob\_letter.get)))

        if len(possibleWords) == 1: break

    plt.plot(best\_hyps)

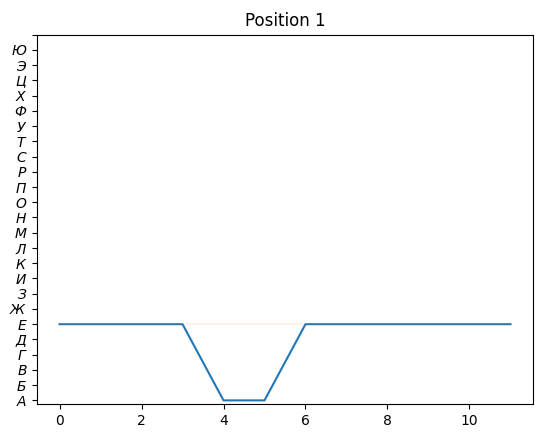
    plt.plot([best\_hyps[-1]] \* len(best\_hyps), alpha = 0.1)

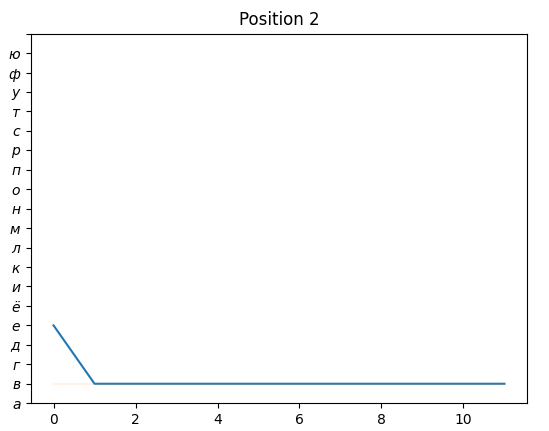
    plt.yticks(range(len(hyps) ), ['$' + hyp + '$' for hyp in hyps[:-1]] + [''])

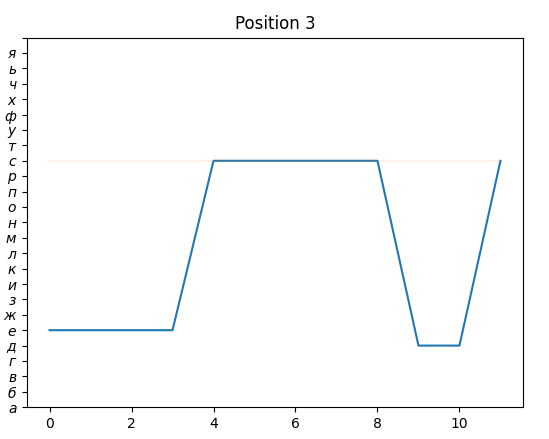
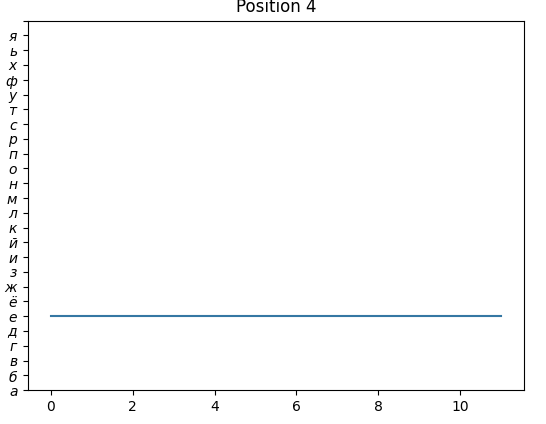
    plt.title('Position ' + str(position + 1))

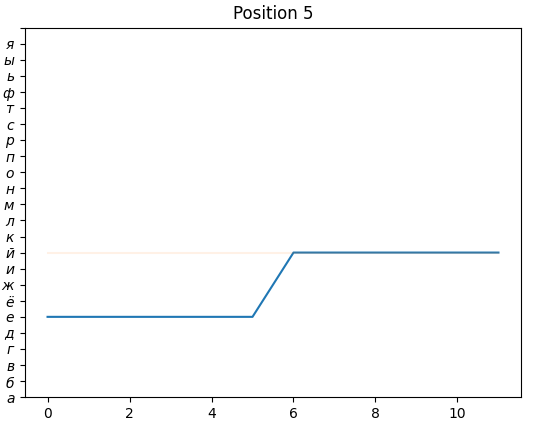
    plt.show()

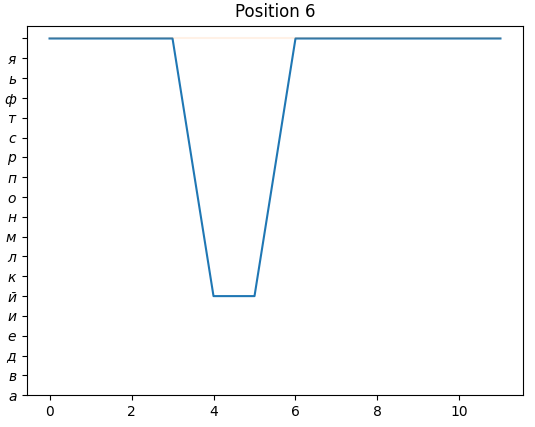
    if prob\_letter[''] == 1: break









Также получили имя " Евсей":

**Листинг 2.6**

probs = {}

hyps = ['', 's', 'uv', 'ucd', 'ucs', 'lv', 'lcd', 'lcs']

hypsRussian = ['$нет$ $символа$', '$знак$', '$заглавная$ $гласная$',

                '$заглавная$ $согласная$ $звонкая$', '$заглавная$ $согласная$ $глухая$',

                '$строчная$ $гласная$', '$строчная$ $согласная$ $звонкая$',

                '$строчная$ $согласная$ $глухая$']

dct={'':0, 's':1, 'uv':2, 'ucd':3, 'ucs':4, 'lv':5, 'lcd':6, 'lcs':7}

for position in range(N):

    for word in listOfWords:

        probs[word]= 1 / len(listOfWords)

    possibleWords = listOfWords

    probLetterType = {}

    num\_of\_possible\_letter\_types = 8

    for hyp in hyps:

        probLetterType[hyp] = sum([probs[word] for word in possibleWords if letterInfo(symbolAtPosition(word, position)) == hyp])

    step = 0

    best\_hyps = []

    for item in infoList:

        step += 1

        if 'letter' in item:

            for word in possibleWords:

                probs[word] = piLetter(word, item['letter'])

        else:

            for word in possibleWords:

                probs[word] = piInfo(word, item['position'], item['info'])

        s = sum(probs.values())

        if s == 0 : continue

        for word in possibleWords:

            probs[word] /= s

        for hyp in hyps:

            probLetterType[hyp] = sum([probs[word] for word in possibleWords if letterInfo(symbolAtPosition(word, position)) == hyp])

        possibleWords = [word for word in possibleWords if probs[word] != 0]

        best\_hyps.append(dct[max(probLetterType, key = probLetterType.get)])

        if len(possibleWords) == 1: break

        if num\_of\_possible\_letter\_types == 1: break

    plt.plot(best\_hyps)

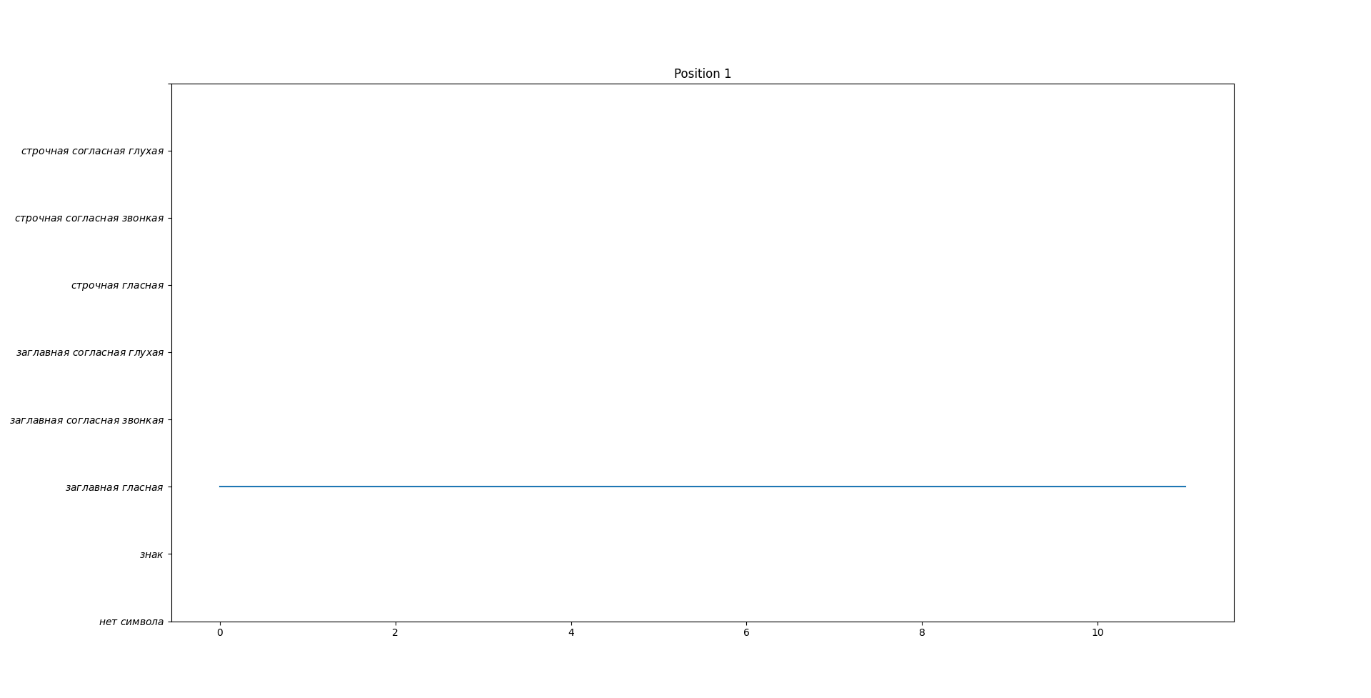
    plt.yticks(range(9), hypsRussian+[''])

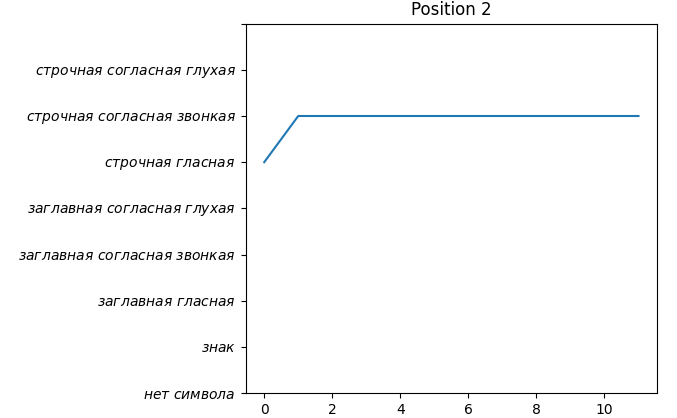
    plt.title('Position ' + str(position + 1))

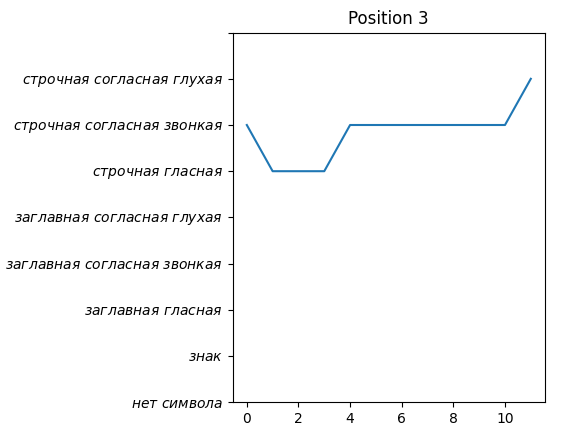
    plt.show()

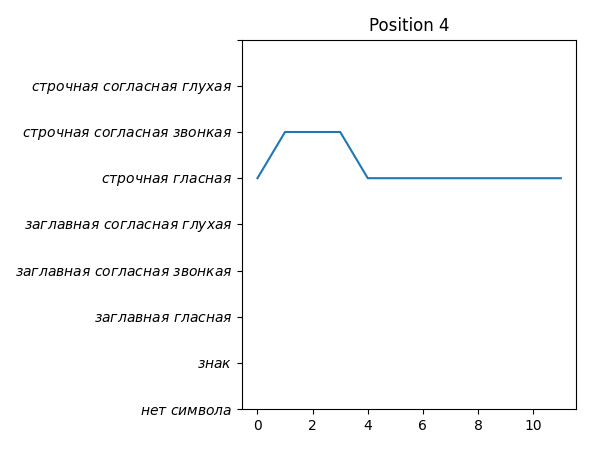
    if probLetterType[''] == 1: break

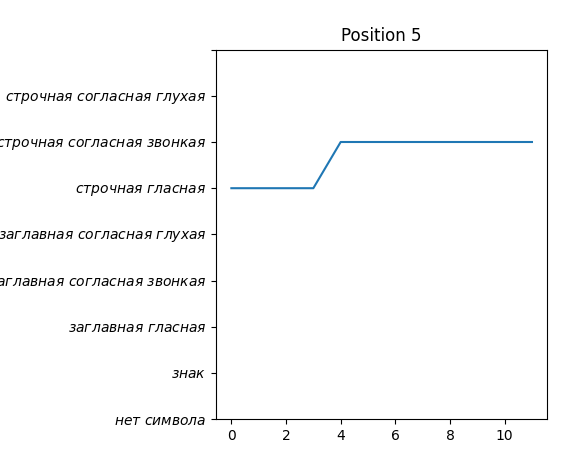
# ---3a---

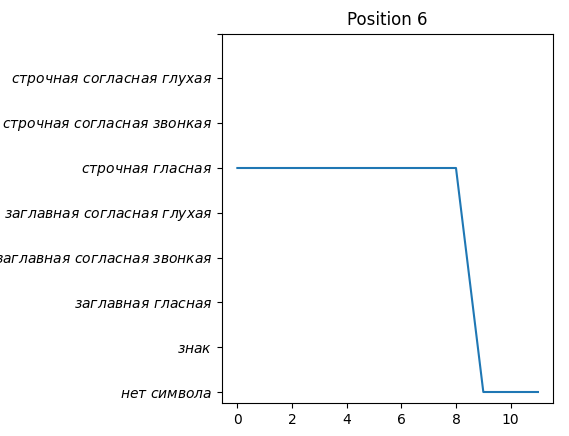












Получили типы букв как в имени " Евсей"

## 3a Определение частоты символов

Определим приближенно частоту тех или иных сообщаемых символов. Исходя из этого можно оценить примерно, сколько каждого из символов в слове. Затем сопоставим эту информацию с тем, что ведущий говорит о позициях букв и их типе. В результате можно сузить количество возможных слов, используемых в качестве гипотез.

**Листинг 3.1**

frLetters = {}

typesOfLetters = {}

for item in infoList:

    if 'letter' in item:

        if item['letter'] in frLetters:

            frLetters[item['letter']] += 1

        else:

            frLetters[item['letter']] = 1

    else:

        typesOfLetters[item['position']] = item['info']

frMin = min(frLetters.values())

lettersOfWord = {}

for letter in frLetters:

    lettersOfWord[letter] = int(frLetters[letter] / frMin)

Letters:

{'е': 1, 'й': 1, 'в': 1, 'с': 1}

Types:

{1: 'lcd', 4: 'cd', 3: 'lv', 2: 'lcs', 0: 'uv'}

**Листинг 3.2**

for word in listOfWords:

    letters= [word[i] for i in range(len(word))]

    n = len(letters)

    types = {}

    for i in range(n):

        types[i] = letterInfo(letters[i])

    check = types == typesOfLetters

    for l in lettersOfWord:

        if len([x for x in letters if x.lower() == l]) != lettersOfWord[l]:

            check = False

            break

    if check: print(word)

Подошло по профилю только одно слово – Евсей

**3d Приведём графики изменения экспериментальных профилей для различного количества опытов.**

**Листинг 3.3**

listOfLetters = []

for i in range(n\_exp):

    frLetters = {l: 0 for l in listOfLetters}

    typesOfLetters = {}

    for item in infoList[:i]:

        if 'letter' in item:

            if item['letter'] in frLetters:

                frLetters[item['letter']] += 1

            else:

                frLetters[item['letter']] = 1

                listOfLetters.append(item['letter'])

        else:

            typesOfLetters[item['position']] = item['info']

    if i in [2 \*\* i for i in range(20)]:

        print(str(i) + ' experiments:')

        print('Types: ', typesOfLetters)

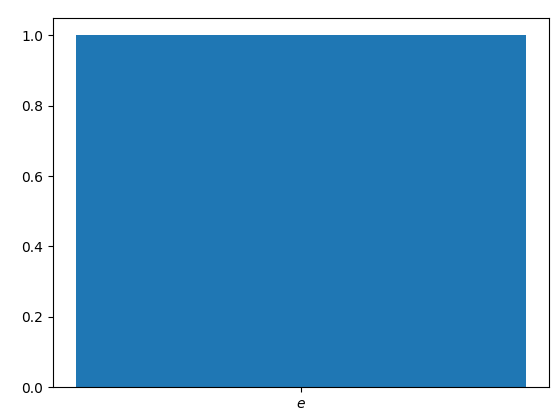
        plt.bar(range(len(frLetters)), [frLetters[l] for l in listOfLetters])

        plt.xticks(range(len(frLetters)),['$' + l + '$' for l in listOfLetters])

        plt.show()

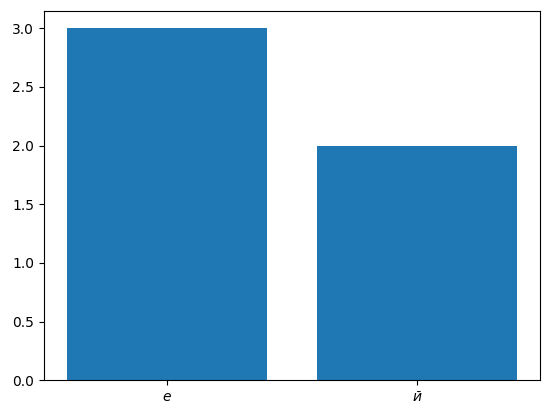
2 experiments:

Types: {1: 'lcd'}



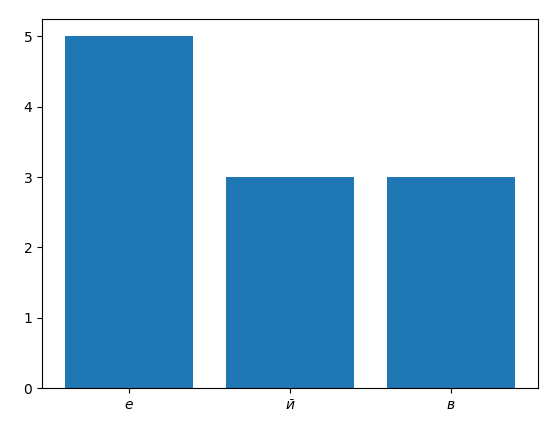
8 experiments:

Types: {1: 'lcd', 4: 'cd', 3: 'lv'}



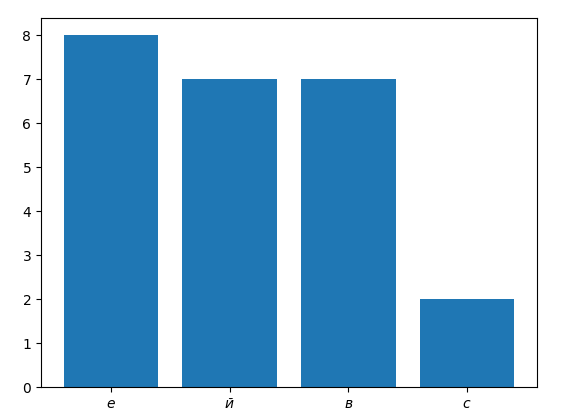
16 experiments:

Types: {1: 'lcd', 4: 'cd', 3: 'lv', 2: 'lcs'}



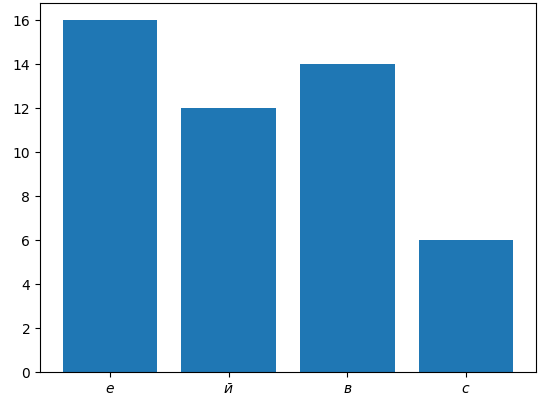
32 experiments:

Types: {1: 'lcd', 4: 'cd', 3: 'lv', 2: 'lcs'}



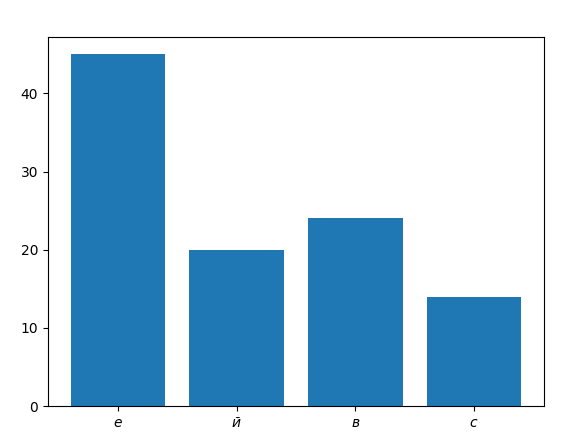
64 experiments:

Types: {1: 'lcd', 4: 'cd', 3: 'lv', 2: 'lcs', 0: 'uv'}



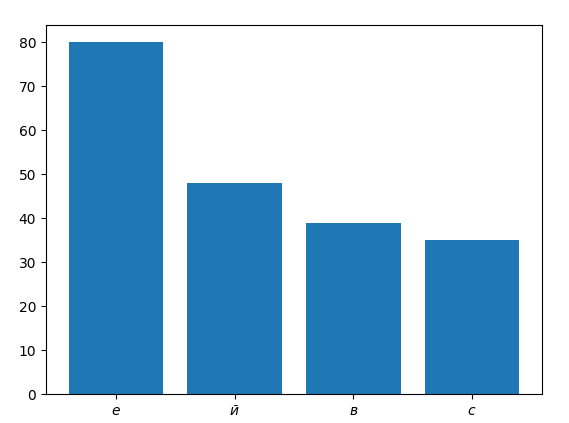
128 experiments:

Types: {1: 'lcd', 4: 'cd', 3: 'lv', 2: 'lcs', 0: 'uv'}



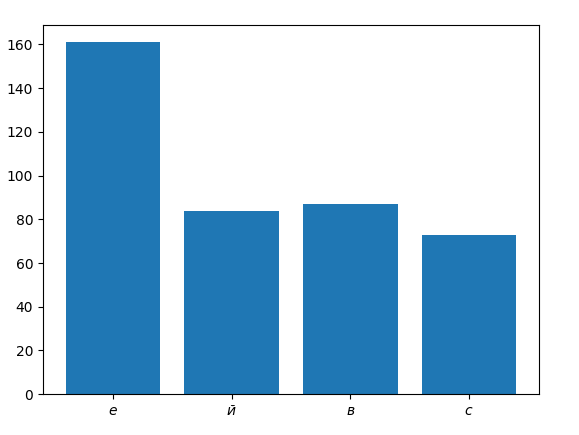
256 experiments:

Types: {1: 'lcd', 4: 'cd', 3: 'lv', 2: 'lcs', 0: 'uv'}



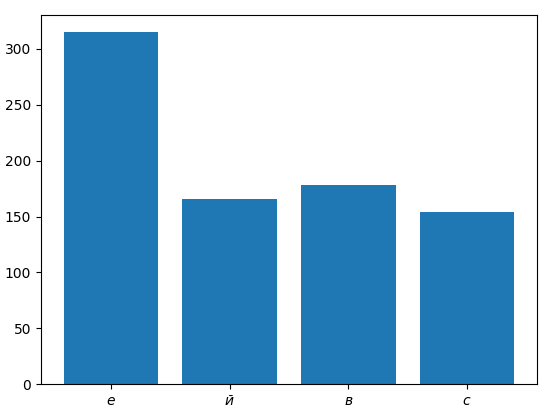
512 experiments:

Types: {1: 'lcd', 4: 'cd', 3: 'lv', 2: 'lcs', 0: 'uv'}



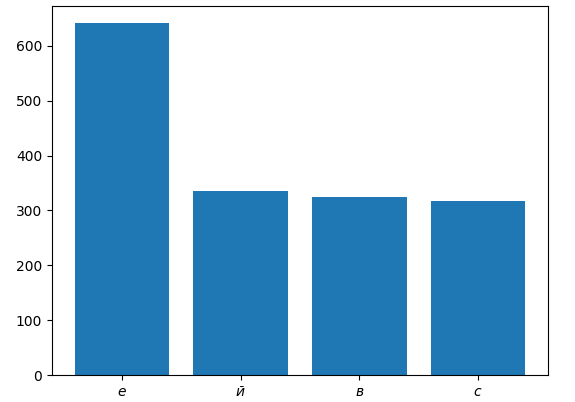
1024 experiments:

Types: {1: 'lcd', 4: 'cd', 3: 'lv', 2: 'lcs', 0: 'uv'}



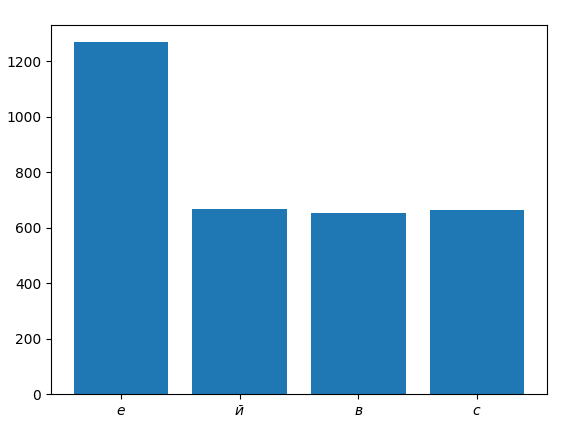
2048 experiments:

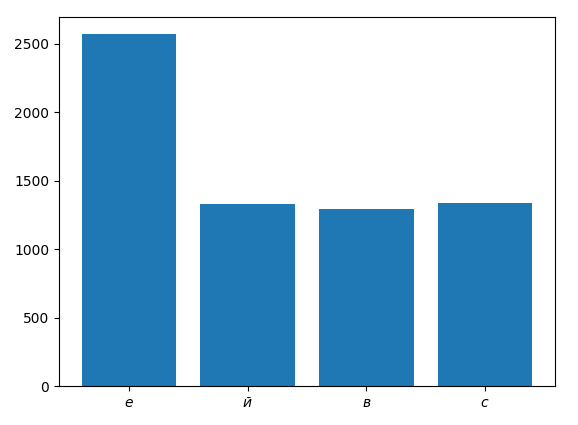
Types: {1: 'lcd', 4: 'cd', 3: 'lv', 2: 'lcs', 0: 'uv'}



4096 experiments:

Types: {1: 'lcd', 4: 'cd', 3: 'lv', 2: 'lcs', 0: 'uv'}





На последнем графике четко видно, что частоты букв е двухкратно превышают частоты других, что говорит о повторении букв в слове.

# Вывод

В ходе работы было получено загаданное имя (**Воздухоплаватель).** Получены все возможные вариации букв для каждой буквы слова, построены экспериментальные и теоретические профили. Все полученные результаты указывают на слово, полученное в пункте **2**, что говорит о правильности выполнения поставленной задачи, т.е. эксперимент совпал с ожидаемым результатом.