

Домашнее задание 5.

Осина Анна

Пляскин Павел

ИАД-2

Оглавление

| | |
|------------------|----|
| Задание 1..... | 2 |
| Задание 1-1..... | 2 |
| Задание 1-2..... | 5 |
| Задание 1-3..... | 8 |
| Задание 2..... | 11 |
| Задание 2-1..... | 11 |
| Задание 2-2..... | 12 |
| Задание 2-3..... | 14 |
| Задание 2-4..... | 17 |
| Задание 3..... | 19 |

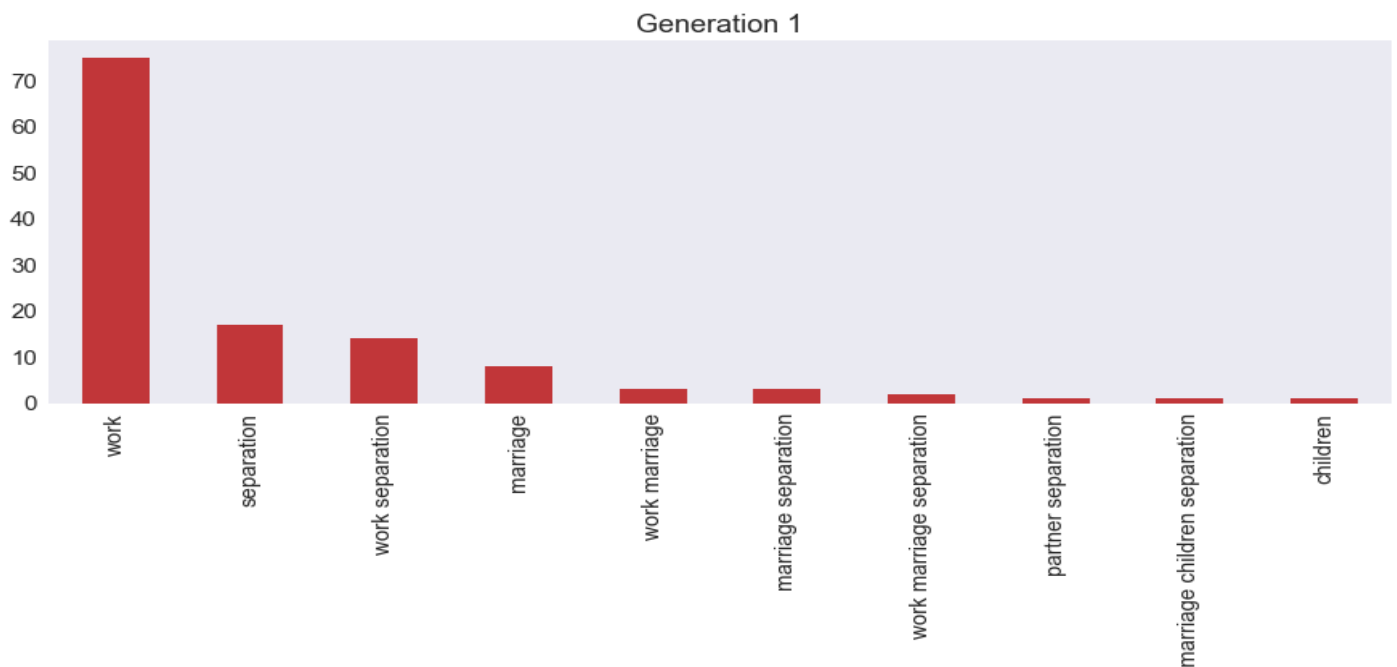
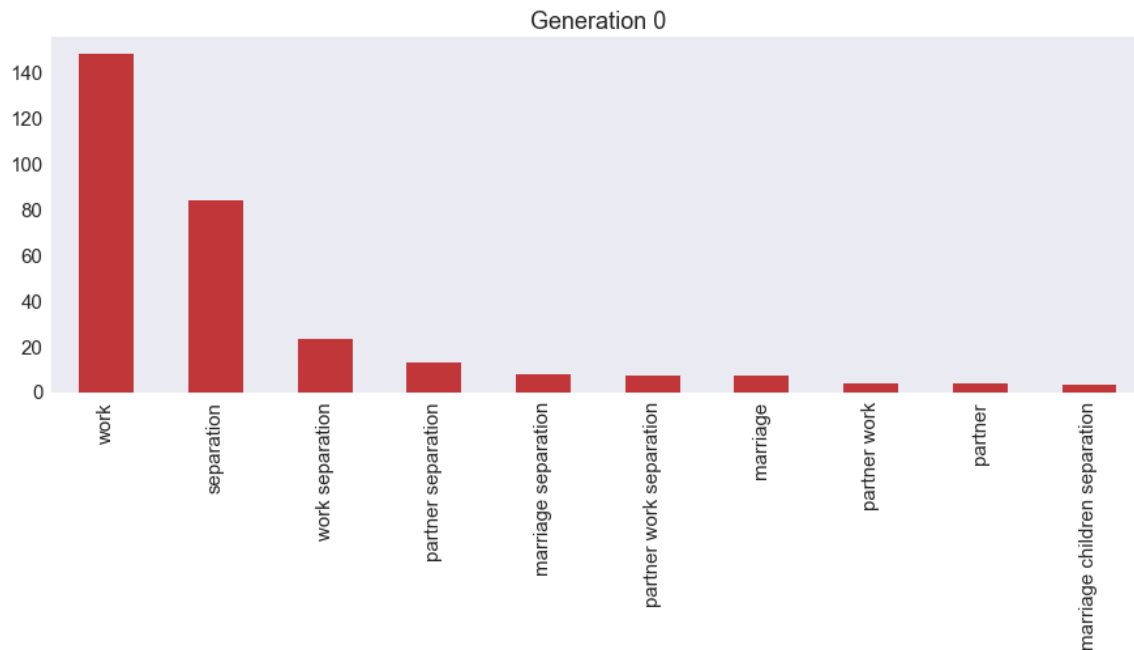
Задание 1.

Задание 1-1.

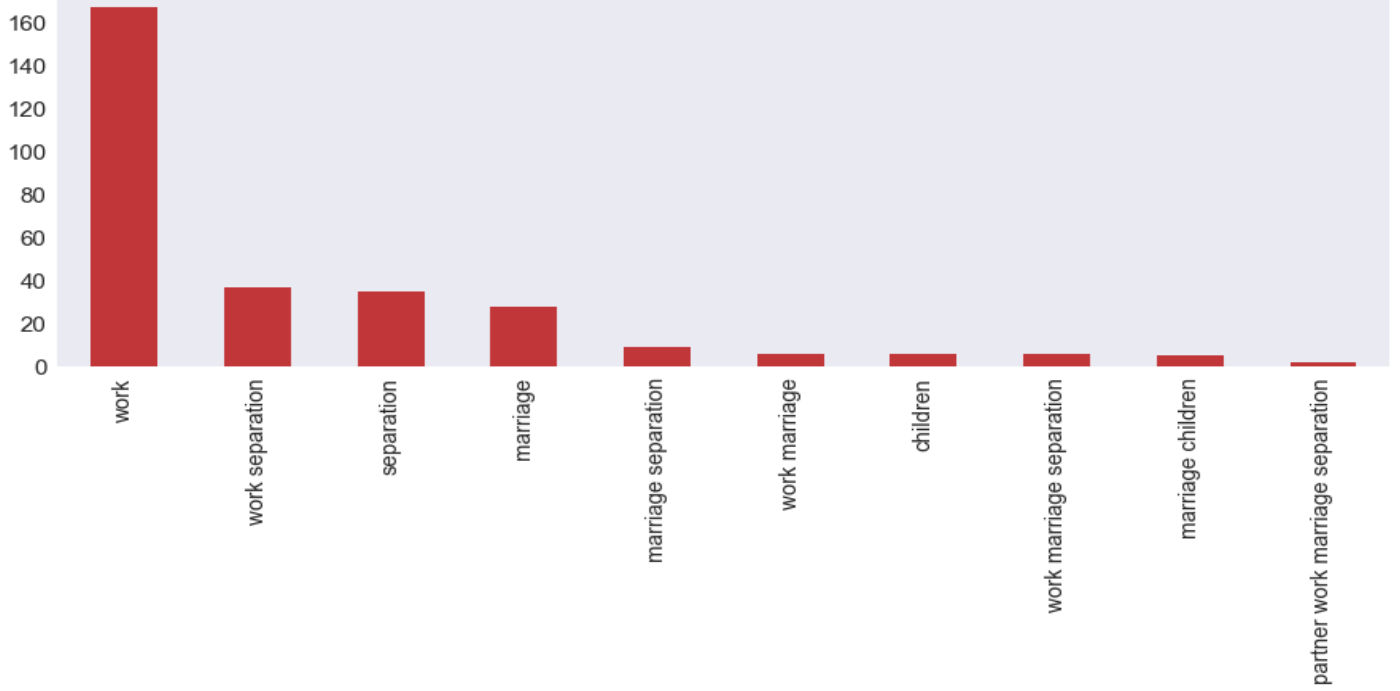
Построим с помощью python гистограммы для 10 наиболее популярных событий, которые являются первыми для каждого поколения:

```
In [5]: data[data.generation == 0].event1.value_counts()[:10].plot(kind='bar', label='author')
plt.title('Generation {}'.format(0))
plt.figure(figsize=(12,8))
```

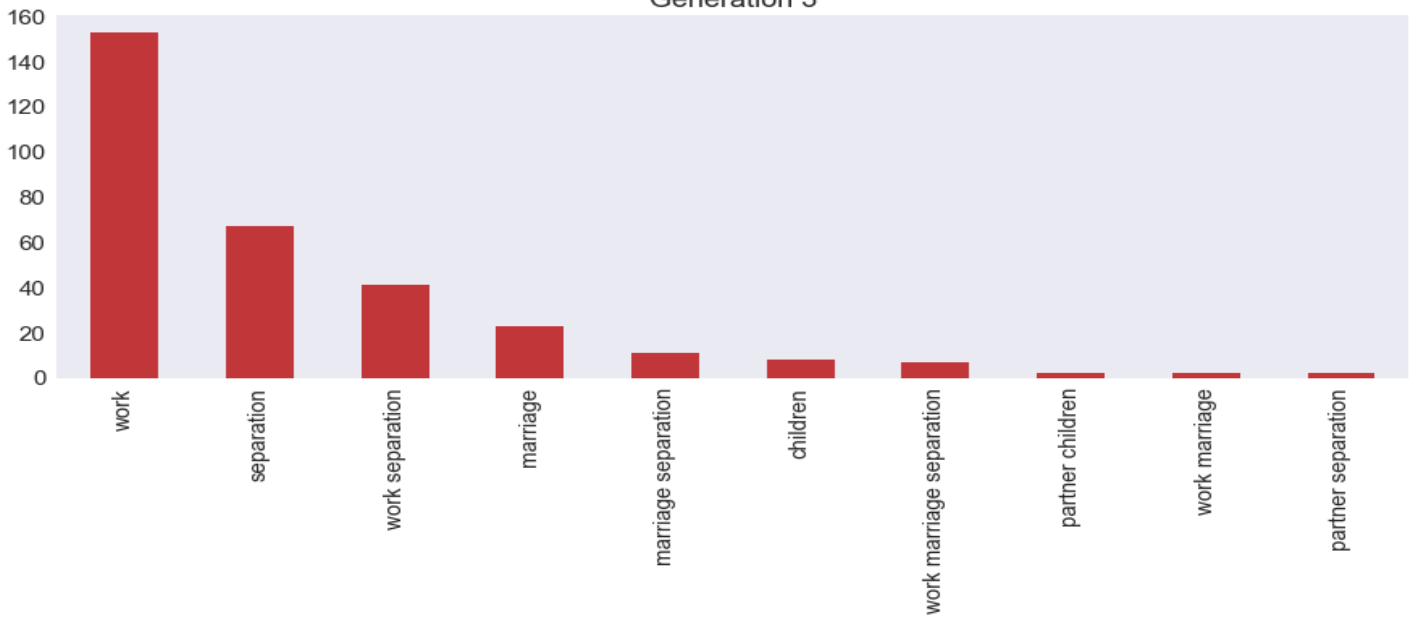
Out[5]: <matplotlib.figure.Figure at 0x2a898e8aef0>



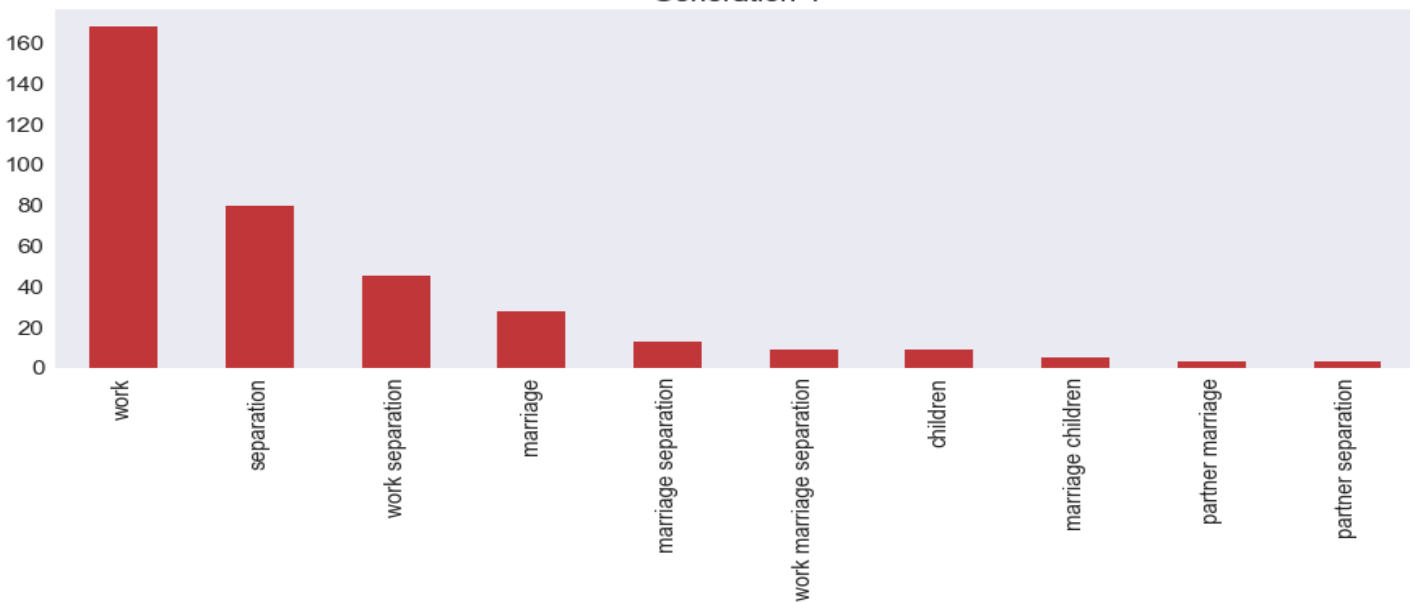
Generation 2

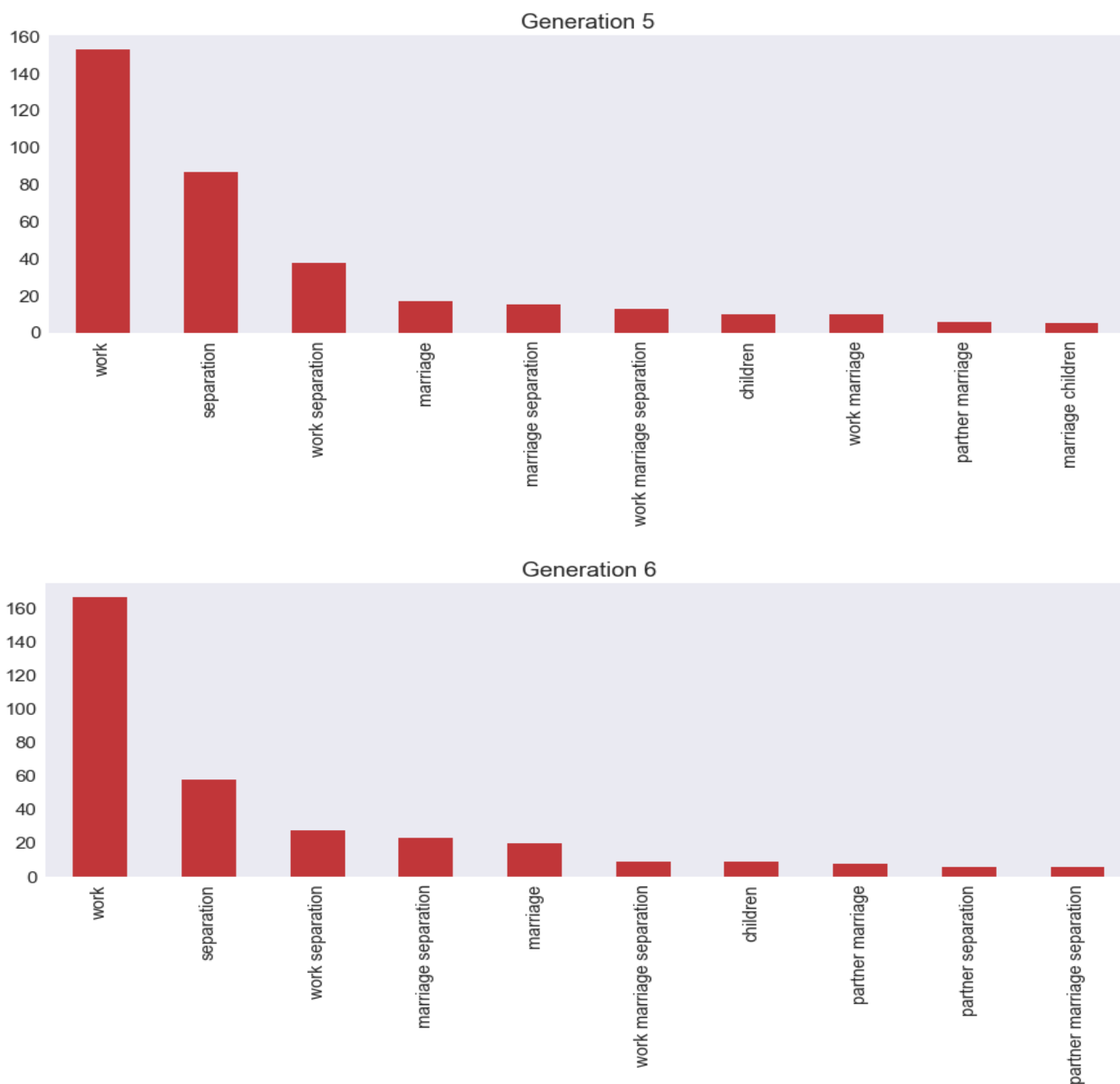


Generation 3



Generation 4

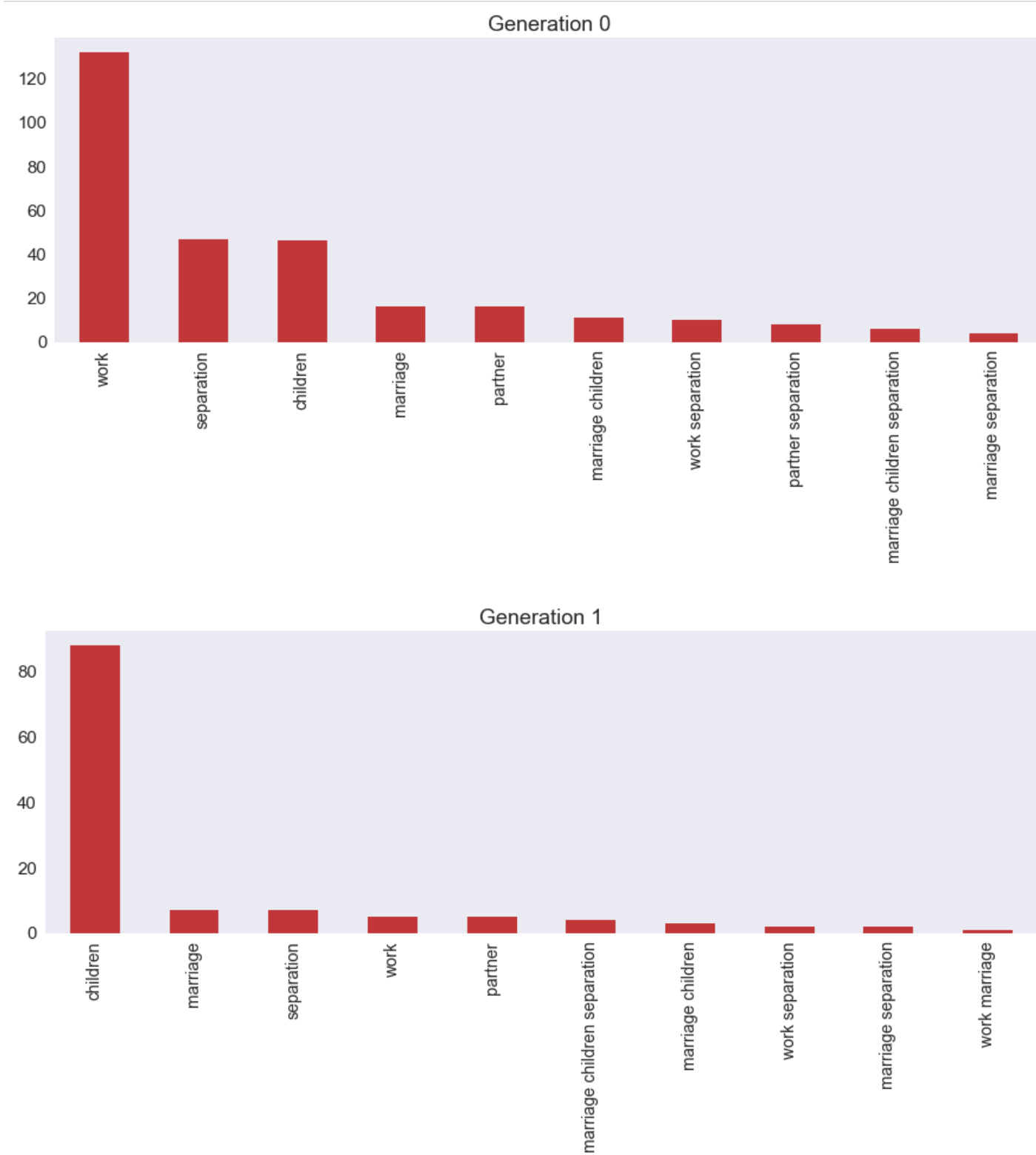




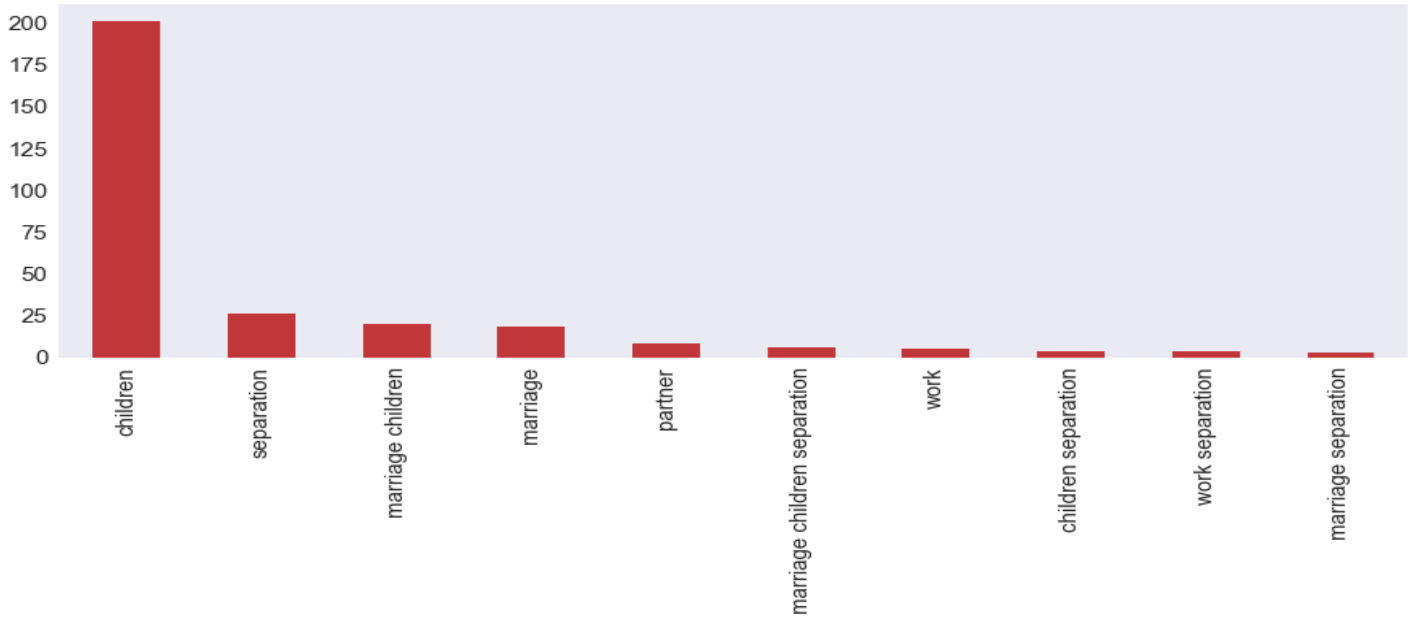
Легко заметить, что тройка событий, которые наиболее часто случаются первыми для всех поколений неизменна. Всегда для всех поколений с большим отрывом по частоте первое событие – *work*. Затем идет *separation* и *work separation*. Только для поколения 2 на втором месте было *work separation*, а на третьем *separation*.

Задание 1-2.

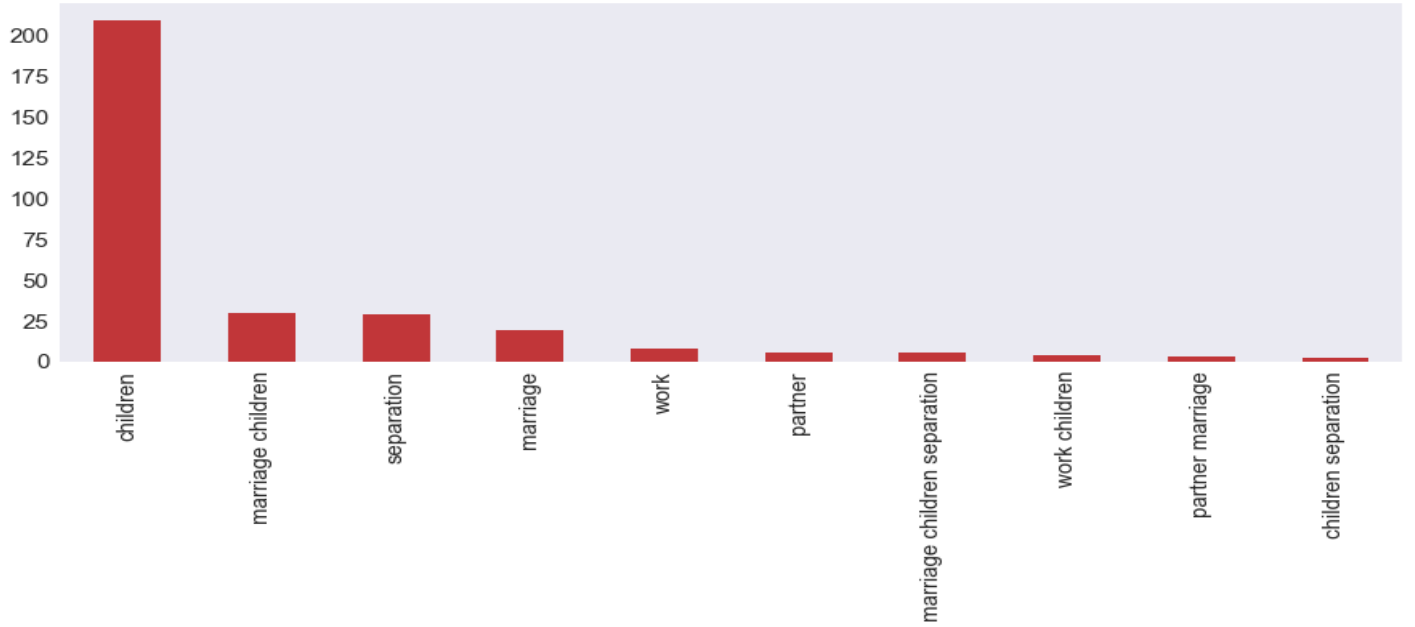
Построим гистограммы частот последних событий для каждого поколения:



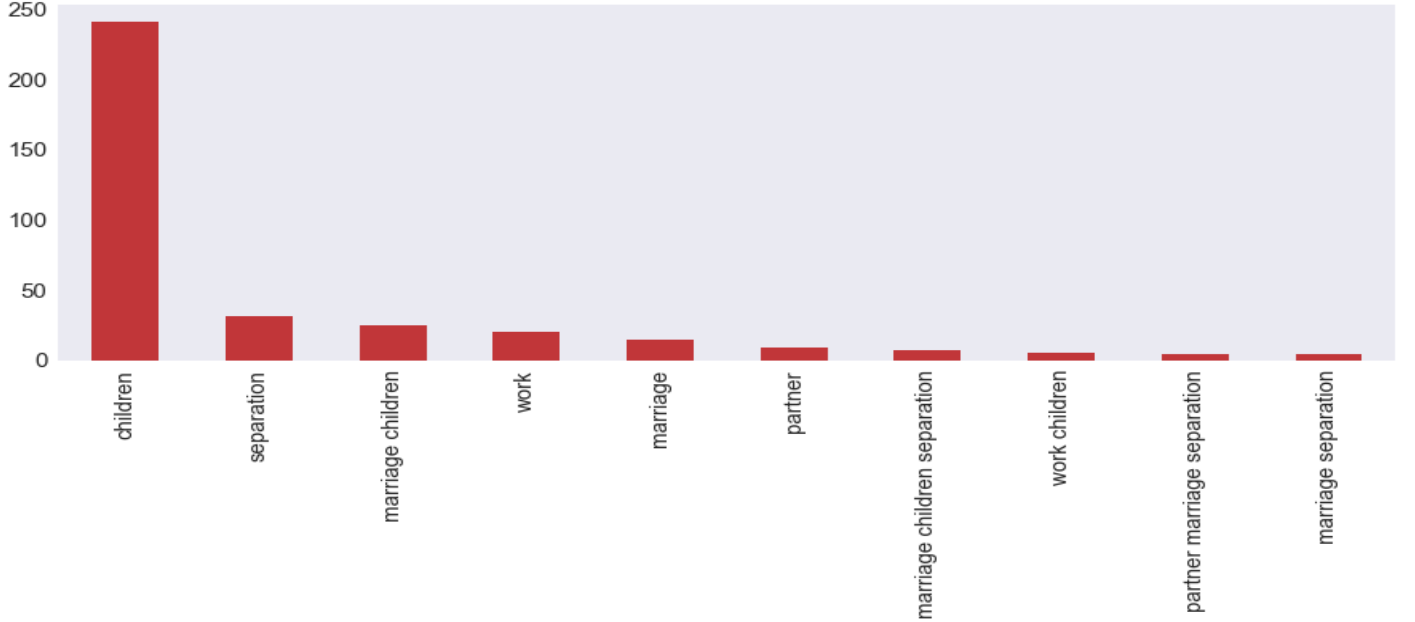
Generation 2

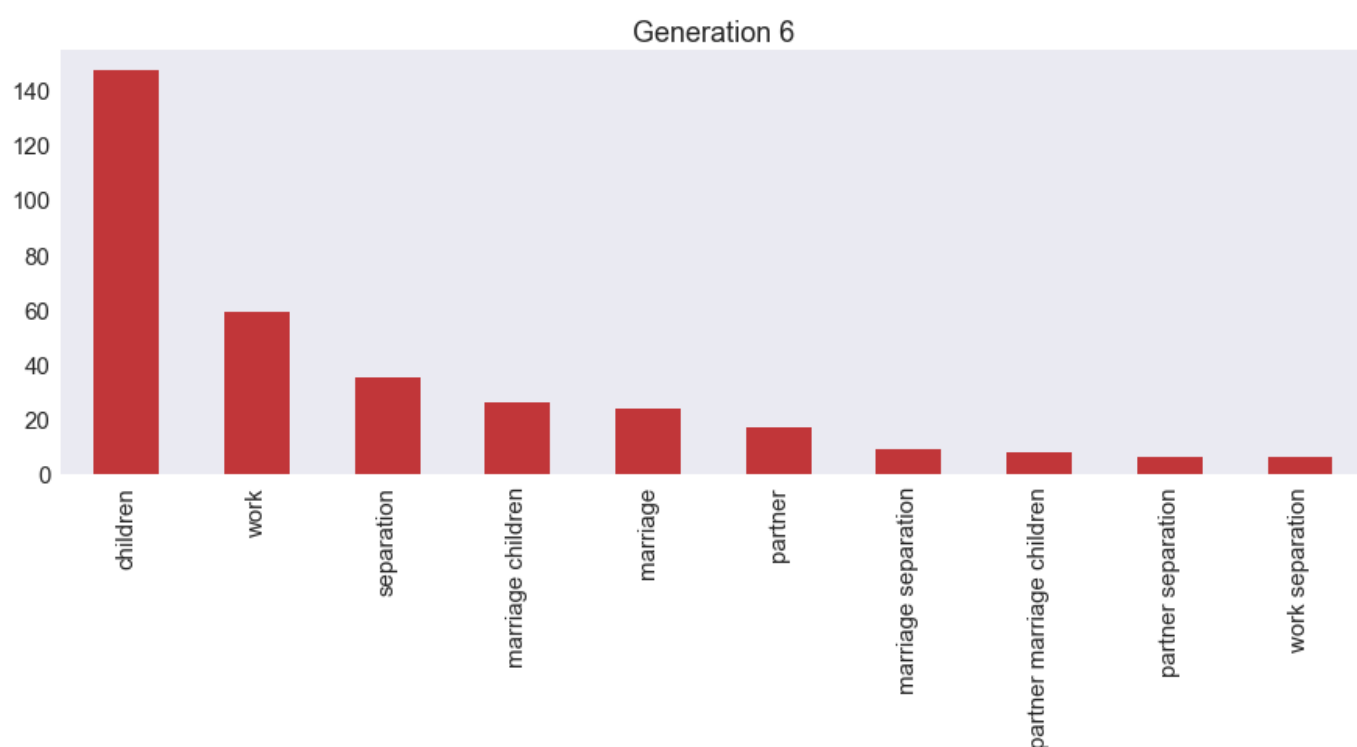
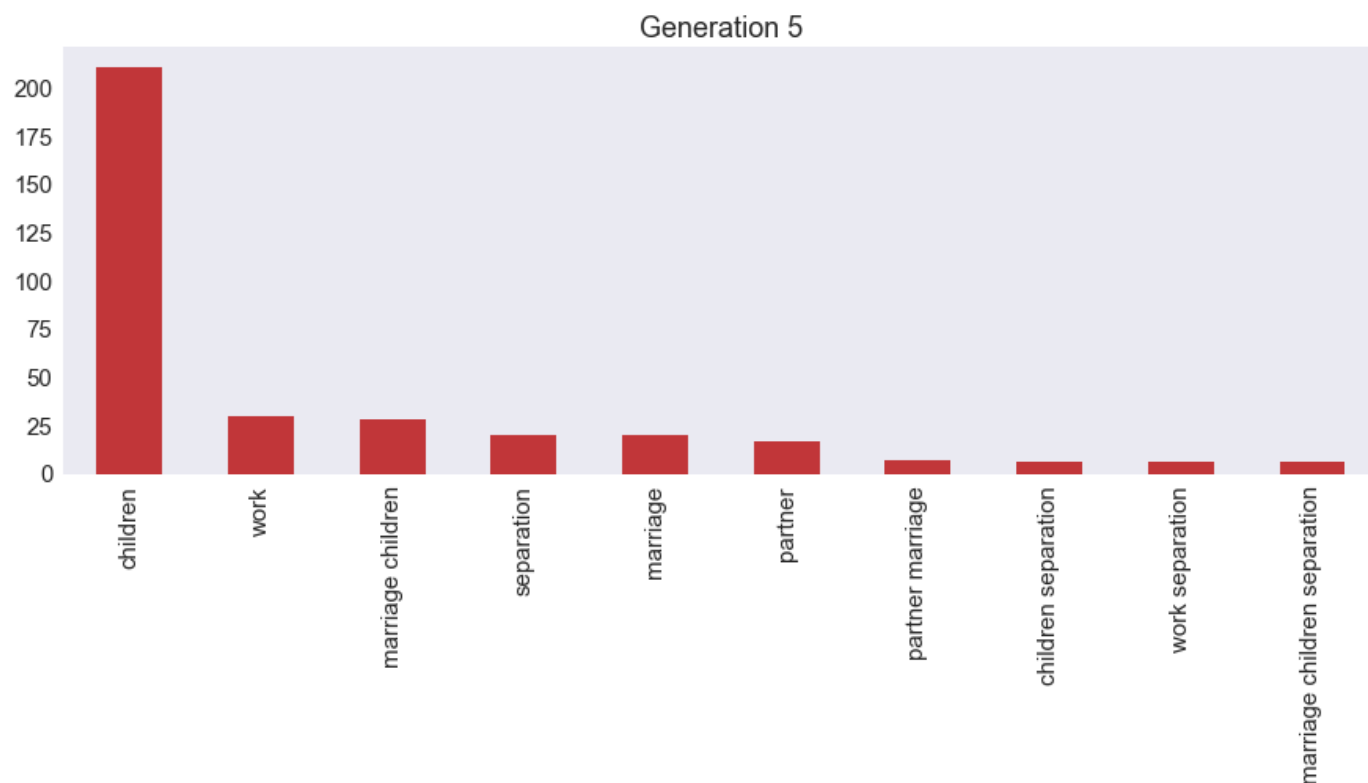


Generation 3



Generation 4



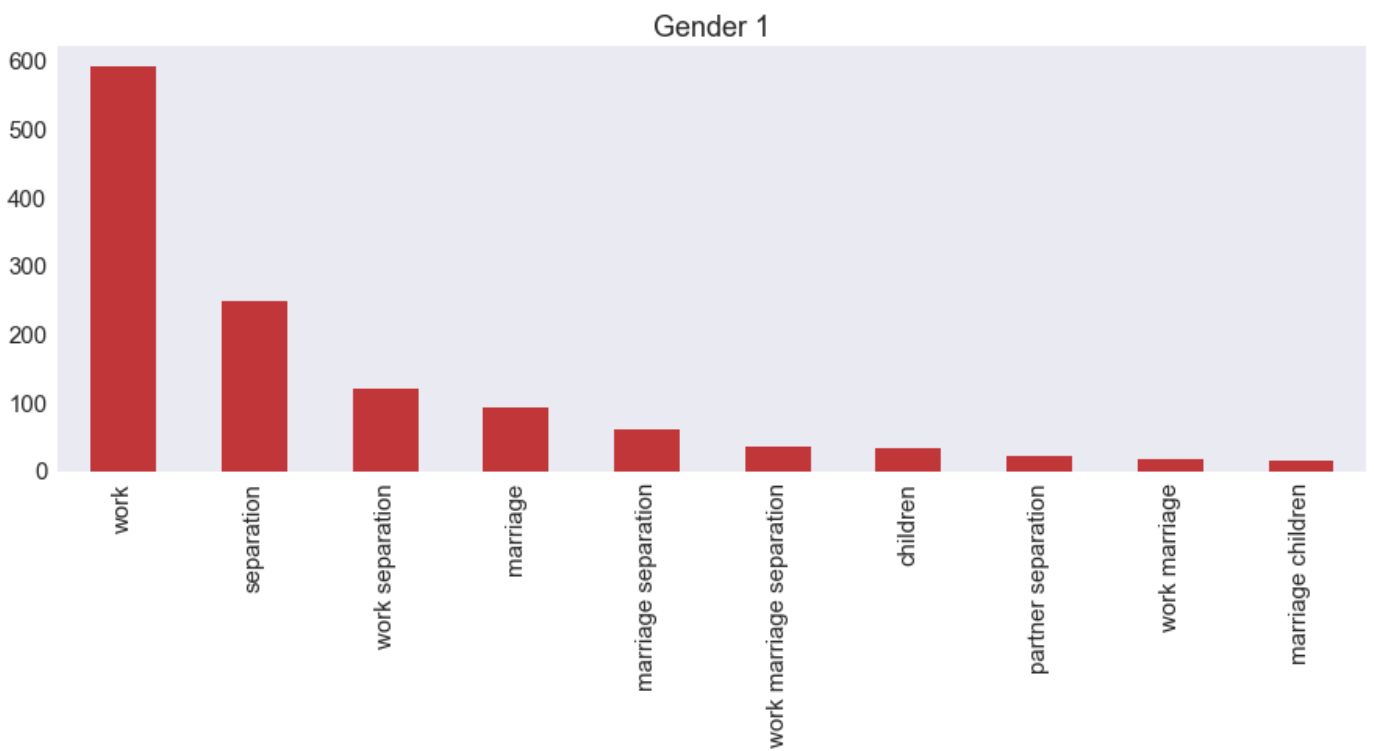
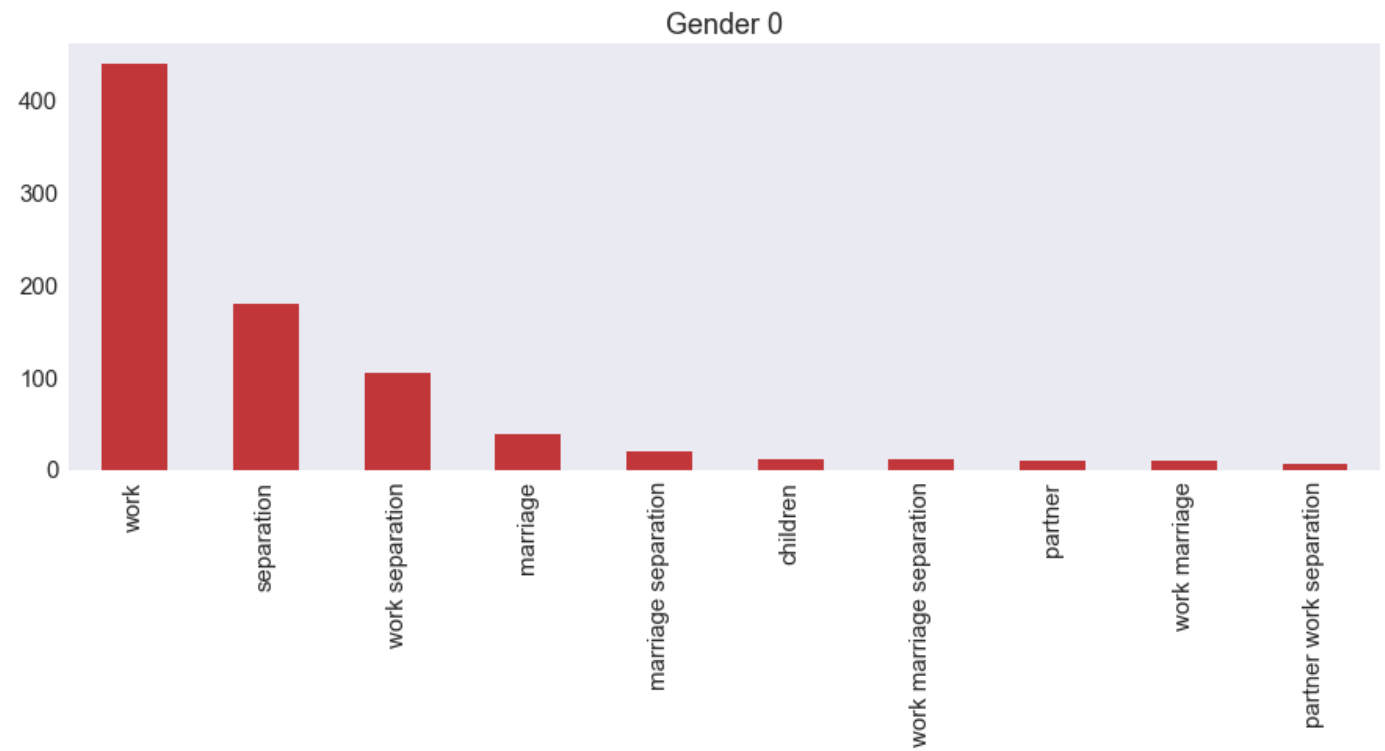


В подавляющем большинстве поколений последним событием становилось *children*. Только в поколении 0 последним событием стало *work*, оно же было самым частым первым событием в этом поколении.

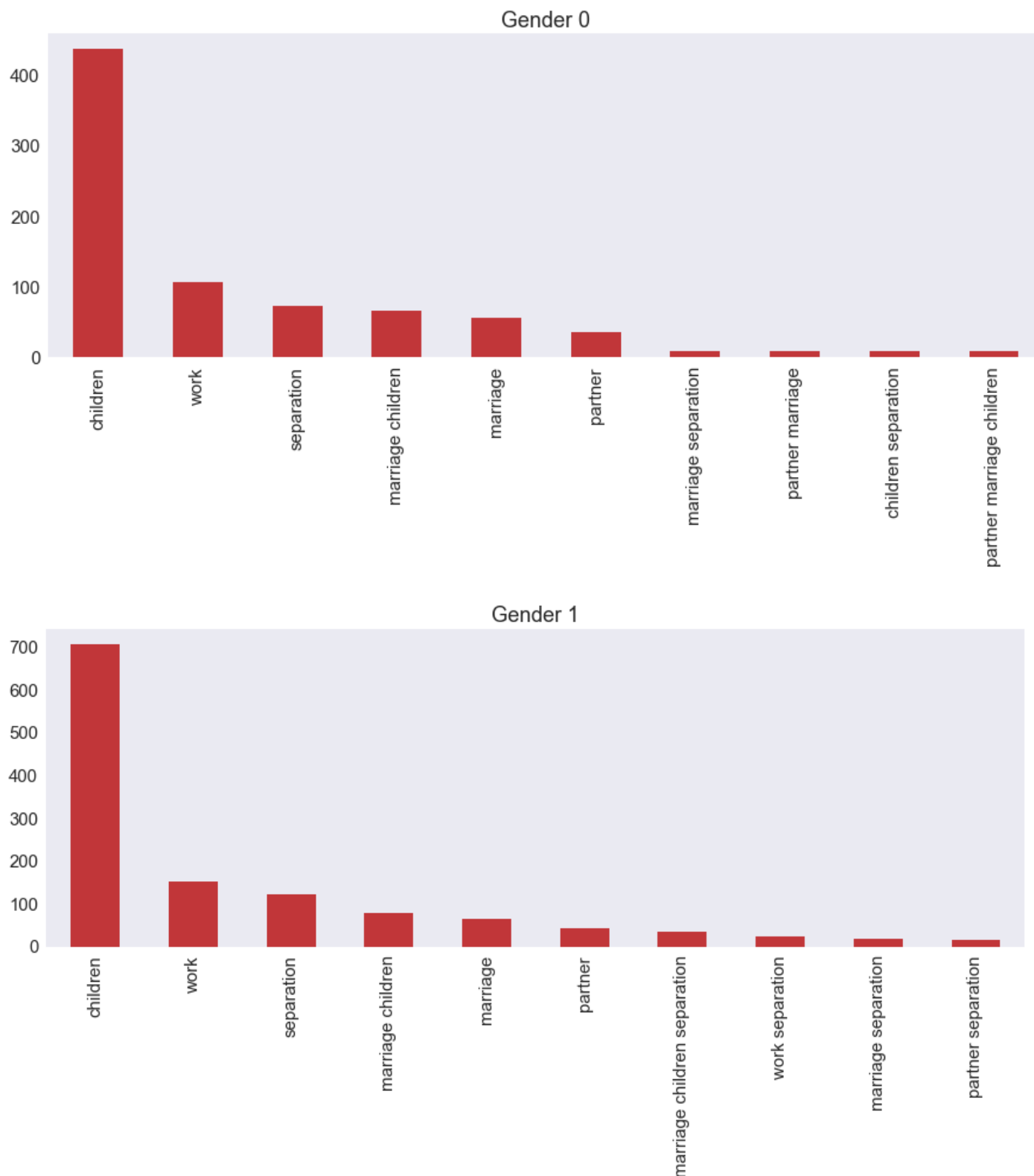
В целом, проглядывается некая связь данных с реальной жизнью, в начале люди пытаются устроить свою карьеру, а затем становятся важными другие ценности, и они начинают заводить собственные семьи.

Задание 1-3.

Какое событие чаще всего было первым для каждого *gender*:

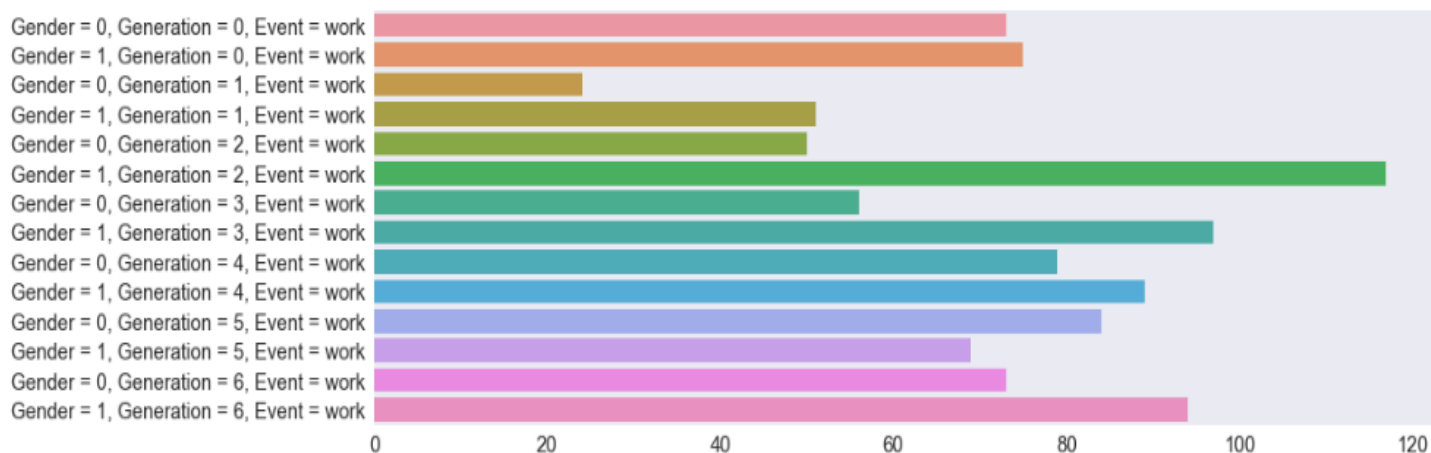


Какое событие чаще всего было последним для каждого *gender*:



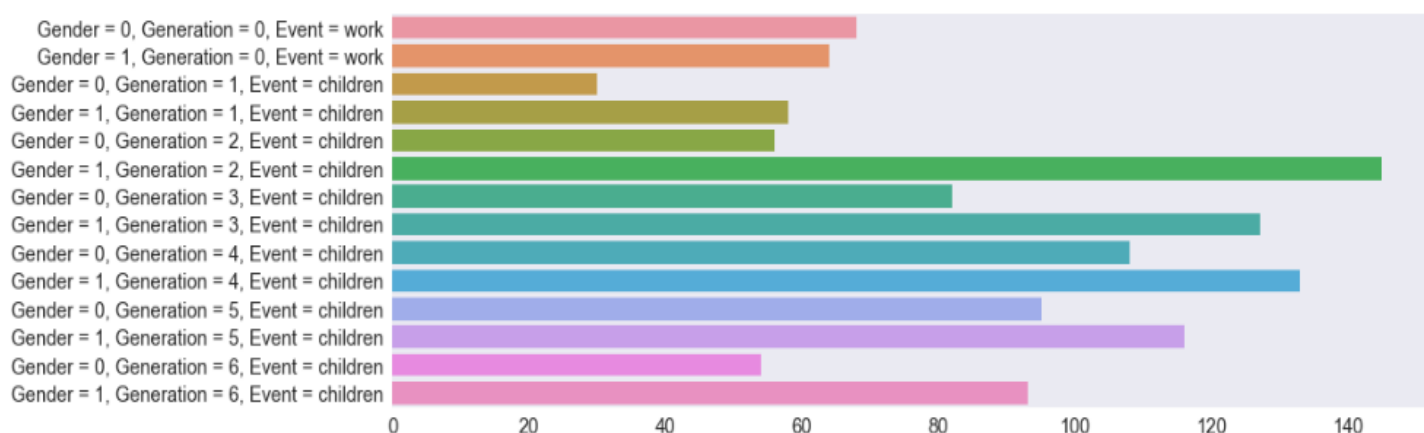
В обоих случаях события не сильно различаются в зависимости от *gender*, разница начинается только после 5 наиболее часто встречающихся событий.

Теперь рассмотрим комбинации признаков *generation-gender*. Так как таких комбинаций в наших данных 14 штук, то построим один общий график, на котором видно какое событие стало самым частым первым событием для каждой комбинации:



Для любой комбинации первым событием чаще всего становилось *work*.

Сделаем тоже самое для последнего события:



Из построенных графиков видно, что значение признака *gender* не оказывает никакой роли на то, какое событие будет самым частым для каждой пары *generation-gender*.

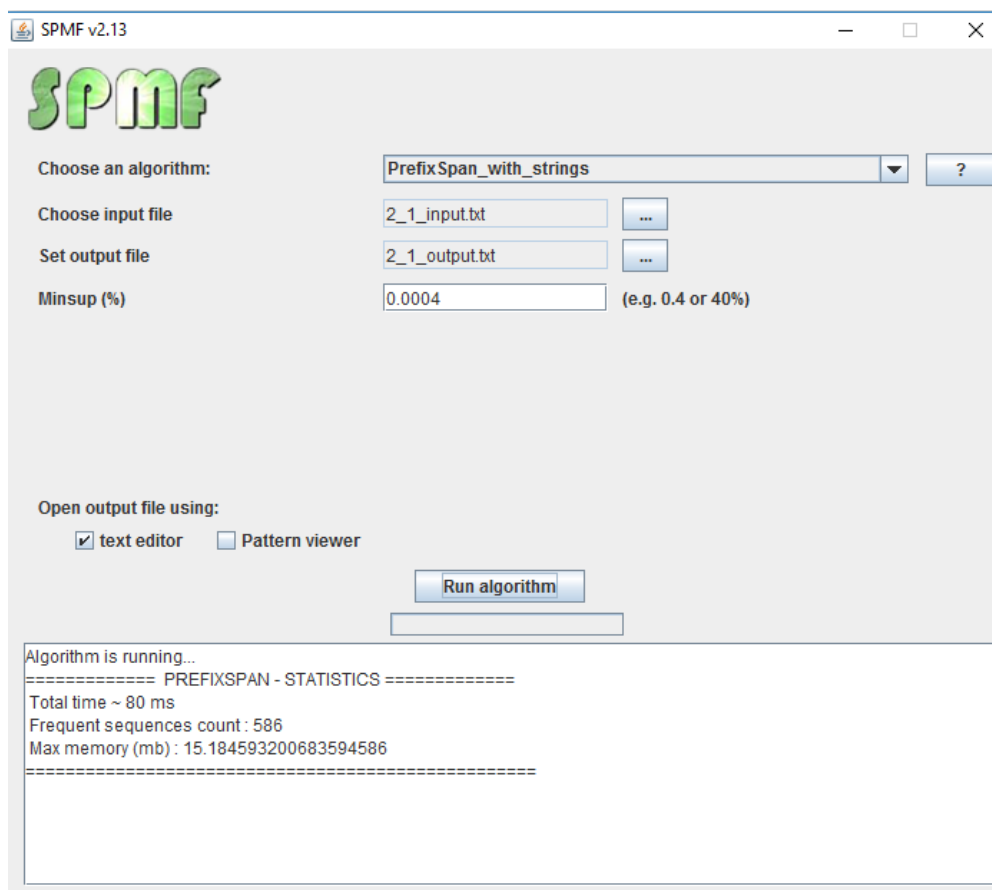
Задание 2.

Задание 2-1.

Для подсчета числа частых и частых замкнутых последовательностей при минимальной поддержке 1 будем использовать данные, в которых будут содержаться только события без признаков *generation*, *gender* и других.

$$\text{minsupp} = \frac{|(A \cup B)'|}{|F|} = \frac{1}{2336} = 0.0004$$

Поиск частых последовательностей с помощью *PrefixSpan*:

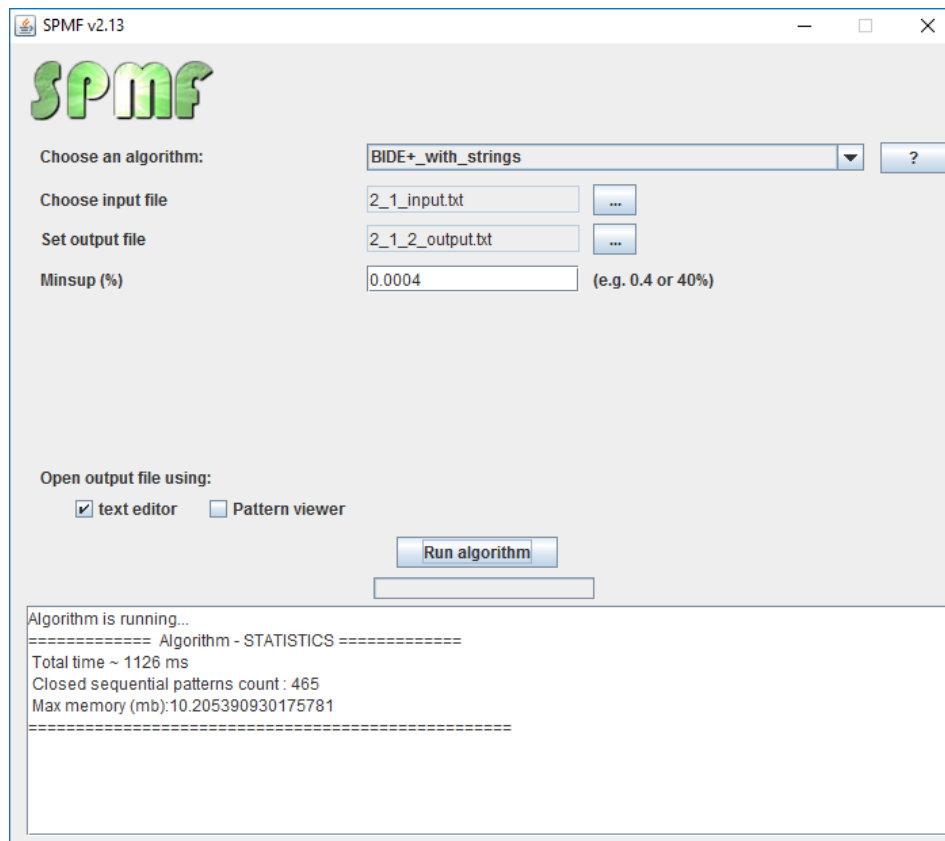


The screenshot shows the SPMF v2.13 application window. The 'Choose an algorithm:' dropdown is set to 'PrefixSpan_with_strings'. The 'Choose input file' field contains '2_1_input.txt' and the 'Set output file' field contains '2_1_output.txt'. The 'Minsup (%)' field is set to '0.0004'. The 'Open output file using:' section has 'text editor' selected. A 'Run algorithm' button is visible. Below the button, a text box displays the following output:

```
Algorithm is running...
===== PREFIXSPAN - STATISTICS =====
Total time ~ 80 ms
Frequent sequences count : 586
Max memory (mb) : 15.184593200683594586
=====
```

Число частых последовательностей равно **586**.

Поиск частых замкнутых последовательностей с помощью *BIDE+*:

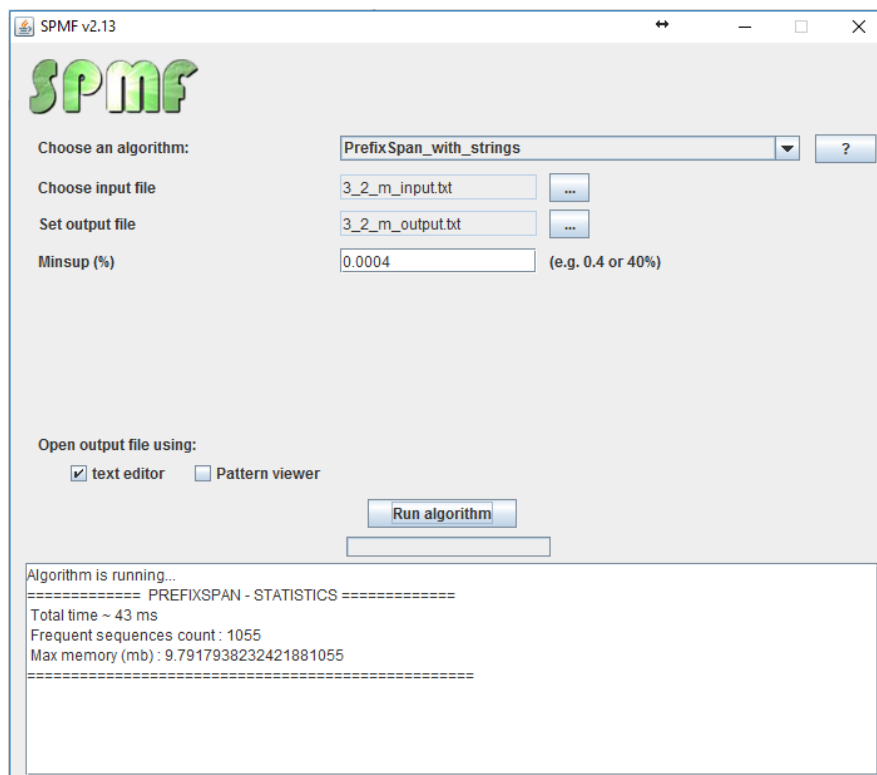


Число замкнутых частых последовательностей равно **465**.

Задание 2-2.

Разделим все наши объекты по признаку *gender* и найдем частые последовательности в них.

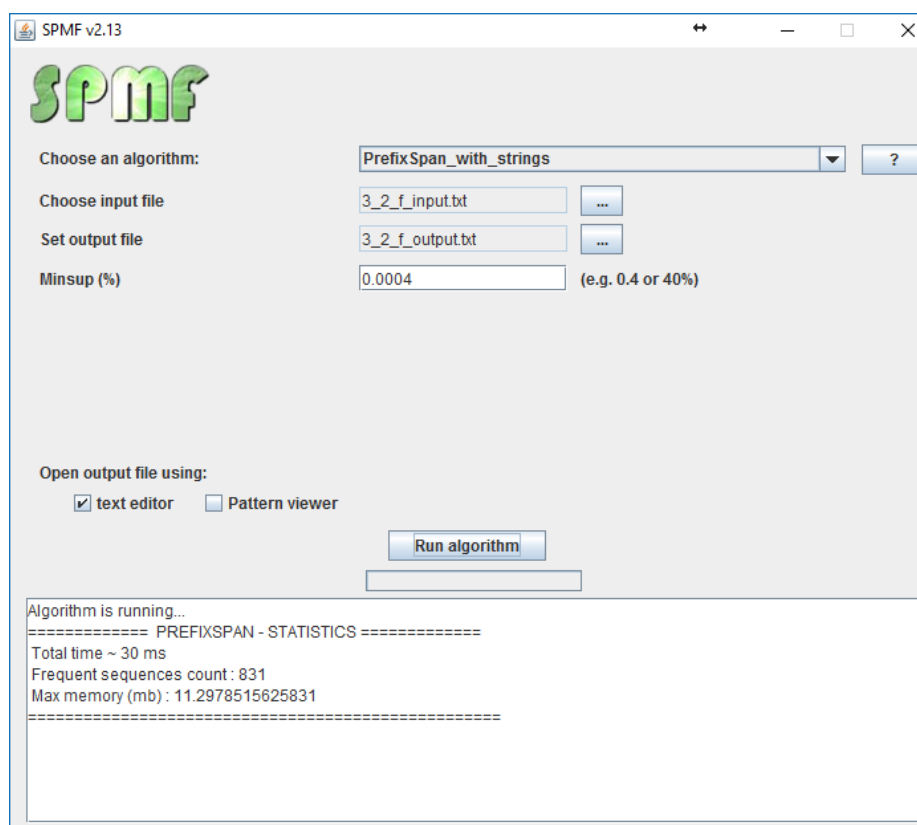
Для мужчин:



Самые популярные из них:

| ДЛИНА | ПОСЛЕДОВАТЕЛЬНОСТЬ | SUP |
|-------|--|-----|
| 2 | <i>marriage children</i> | 782 |
| 3 | <i>work marriage children</i> | 480 |
| 4 | <i>separation work marriage children</i> | 69 |
| 5 | <i>work separation partner children marriage</i> | 6 |

Для женщин:



Самые популярные:

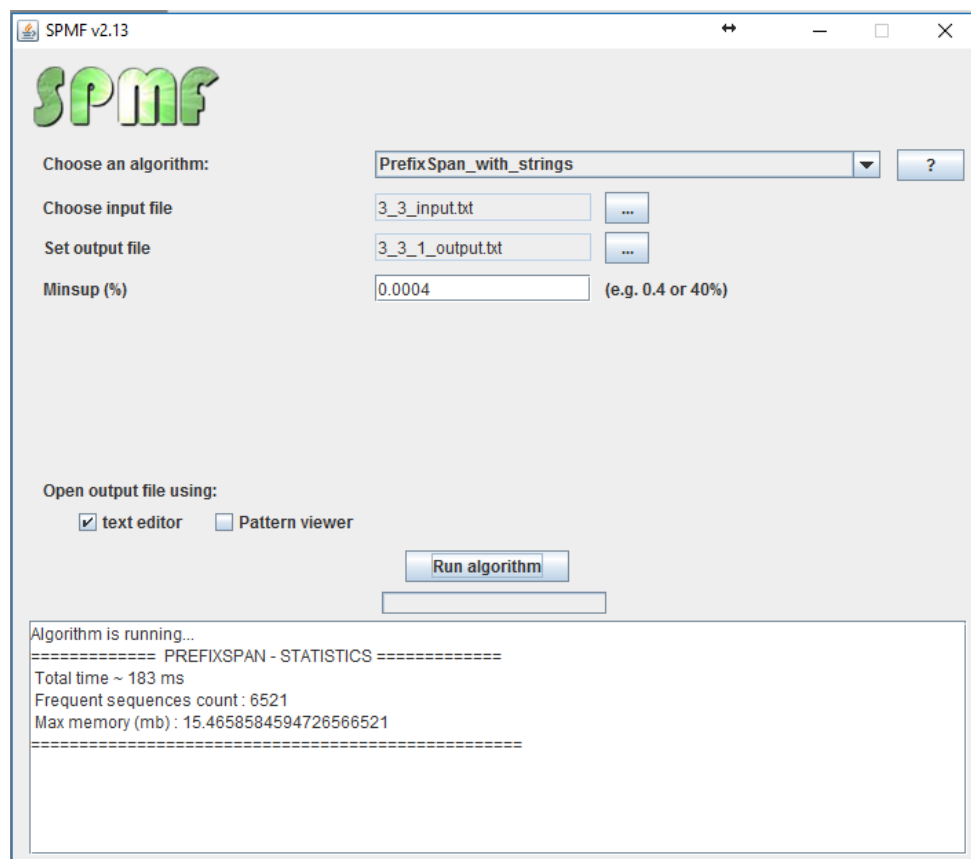
| ДЛИНА | ПОСЛЕДОВАТЕЛЬНОСТЬ | SUP |
|-------|--|-----|
| 2 | <i>Work; children</i> | 514 |
| 3 | <i>Work; marriage; children</i> | 362 |
| 4 | <i>Work; separation; marriage; children</i> | 95 |
| 5 | <i>Work; separation; partner; marriage; children</i> | 8 |

Задание 2-3.

Добавим во входной файл значения признаков *gender* и *generations*. Таким образом теперь файл, который мы будем использовать для поиска последовательностей, имеет вид:

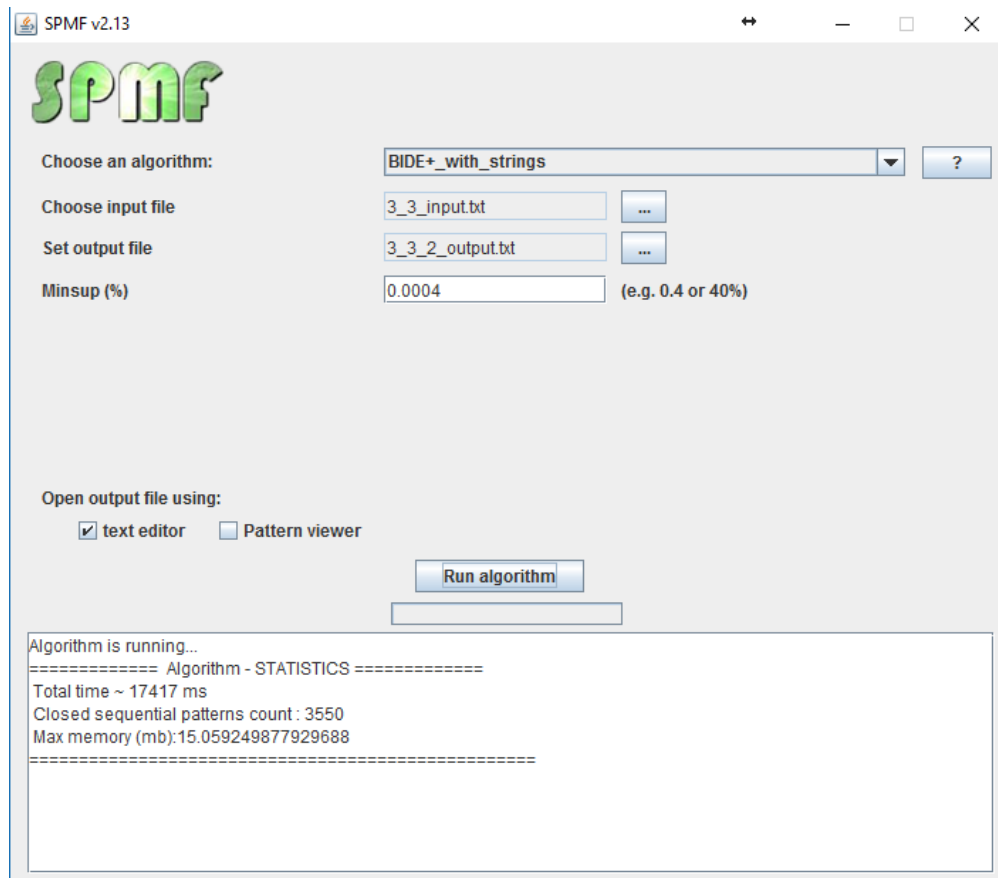
```
gender=1 -1 generation=6 -1 work separation -1 partner -1 marriage -1 children -1 -2
```

Найдем количество число частых последовательностей с помощью метода *PrefixSpan* на этих данных:



Число частых последовательностей равно **6 521**.

Найдем число частых замкнутых последовательностей при помощи *BIDE+*:



Число частых последовательностей равно **3 550**.

Аналогично заданию 2-2 найдем наиболее частые последовательности для комбинаций признаков *gender* и *generation*.

| GENDER- GENERATION | ПОСЛЕДОВАТЕЛЬНОСТЬ | SUP |
|-----------------------|-----------------------------|-----|
| 1-0 | <i>Separation; children</i> | 36 |
| 0-0 | <i>Separation; work</i> | 30 |
| 1-1 | <i>Marriage; children</i> | 60 |
| 0-1 | <i>Work; children</i> | 32 |
| 1-2 | <i>Marriage; children</i> | 162 |
| 0-2 | <i>Work; children</i> | 66 |
| 1-3 | <i>Work; children</i> | 141 |
| 0-3 | <i>Marriage; children</i> | 88 |
| 1-4 | <i>Marriage; children</i> | 154 |
| 0-4 | <i>Marriage; children</i> | 123 |
| 1-5 | <i>Work; children</i> | 128 |
| 0-5 | <i>Work; children</i> | 122 |
| 1-6 | <i>Marriage; children</i> | 114 |
| 0-6 | <i>Work; marriage</i> | 74 |

Задание 2-4.

Для поиска последовательных ассоциативных правил нам снова придется сформировать новый файл. Закодируем все события на числовые обозначения, в одном файле оставим только женщин, во втором только мужчин.

```
In [6]: def create_output_txt(data, filename):
        outfile=open(filename+'.txt', 'w')
        for seq in data.values:
            for val in seq:
                if len(val)>0:
                    outfile.write(val+' -1 ')
            outfile.write('-2\n')
        outfile.close()

In [7]: data_4 = data.drop(['generation', 'education', 'place', 'last_event'], axis=1)
        columns_names = [x for x in data_4 if x != 'gender']

In [8]: data_4 = data_4.fillna('')
        data_4[columns_names] = data_4[columns_names].astype(str)

In [9]: events = data_4[columns_names].values.flatten()
        events = np.unique([x for x in events if x != ''])

In [10]: events_d = {v:str(k) for k, v in enumerate(events)}

In [11]: data_4.event1 = data_4.event1.map(events_d)
        data_4.event2 = data_4.event2.map(events_d)
        data_4.event3 = data_4.event3.map(events_d)
        data_4.event4 = data_4.event4.map(events_d)
        data_4.event5 = data_4.event5.map(events_d)

In [12]: data_m = data_4[data_4.gender == 1].drop('gender', axis=1).fillna('').astype(str)
        data_f = data_4[data_4.gender == 0].drop('gender', axis=1).fillna('').astype(str)

In [15]: create_output_txt(data_m, '2_4_m_input.txt')

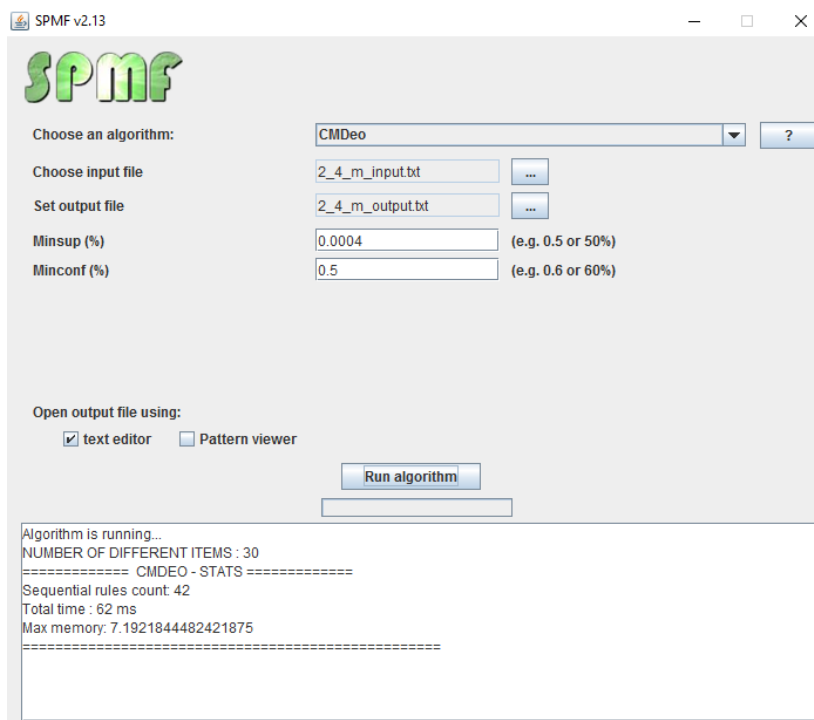
In [16]: create_output_txt(data_f, '2_4_f_input')
```

Формат, полученного файла:

```
30 -1 6 -1 2 -1 0 -1 -2
23 -1 22 -1 -2
```

Где 6 это айди события *partner*, а под 0, например, закодировано событие *children* и так далее.

Для поиска последовательных ассоциативных правил будем использовать метод *CMDeo* реализованный в SPMF.



Пример наиболее достоверных правил для *gender=0*:

1. 2,13,23 ==> 0 #SUP: 8 #CONF: 1.0 #LIFT: 1.826171875
2. 5 ==> 0 #SUP: 82 #CONF: 0.8723404255319149 #LIFT: 1.593043550531915
3. 2,30 ==> 0 #SUP: 60 #CONF: 0.8695652173913043 #LIFT: 1.5879755434782612

Перекодировав события обратно, получаем:

1. Marriage, partner separation, work ==> children
2. Marriage separation ==> children
3. Marriage, work separation ==> children

Пример наиболее достоверных правил для *gender=1*:

1. 29 ==> 0 #SUP: 35 #CONF: 0.9722222222222222 #LIFT: 1.5602329133256967
2. 5 ==> 0 #SUP: 190 #CONF: 0.8715596330275229 #LIFT: 1.3986884832434818
3. 16 ==> 2 #SUP: 1 #CONF: 1.0 #LIFT: 2.7363281249999996

Перекодировав получаем:

1. Work marriage separation ==> children
2. Marriage separation ==> children
3. Partner work children separation ==> marriage

Задание 3.

Поиск ассоциативных правил впервые начал применяться в такой ежедневной и обыденной ситуации как поход в супермаркет. Еще в 1993 году появился алгоритм для нахождения ассоциативных правил в покупках в магазине. Итак, формализуем данную задачу:

Имеется некоторый контекст $K = (F, T, I_{FT})$. Где F – множество транзакций(покупок). T – множество продуктов, которые продаются в рассматриваемом магазине. $f I t$ – это отношение показывающее, что в транзакции f был куплен товар t .

Для решения задачи поиска ассоциативных правил, частых множеств и частых (под)последовательностей в SMPF, данные должны иметь следующий формат:

Каждая строка — это объект (в нашем случае транзакции), в строке записаны признаки, которыми этот объект обладает (в нашем случае это будут товары). Пример такого формата данных следующий:

| Транзакция | Товары |
|------------|-----------|
| t1 | 1 2 4 5 |
| t2 | 2 3 5 |
| t3 | 1 2 4 5 |
| t4 | 1 2 3 5 |
| t5 | 1 2 3 4 5 |
| t6 | 2 3 4 |

Соответственно в файле, загружаемом в SMPF не нужно явно указывать в строке номер транзакции, так как id транзакции будет соответствовать номеру строки, в которой записаны продукты из этой транзакции. Также нужно учесть, что работу со строками поддерживают далеко не все алгоритмы, поэтому лучше сразу записывать в файле не сами названия товаров, а их id.

Получив такие данные, мы можем решить множество задач и найти различные зависимости в корзинах покупателей. Мы можем не только рассматривать корзины всех покупателей в целом, но и также выделять некоторые группы покупателей, если такая информация доступна, и искать их внутренние зависимости.

Как пример, полученных результатов может быть ассоциативное правило:

$$\text{Сыр} \Rightarrow \text{Масло} \quad SUP=3\% \quad CONF=75\%$$

Из него мы можем сделать вывод, что 75% транзакций, содержащих *Сыр*, также содержат *Масло*. А 3% транзакций от общего числа всех транзакций содержат оба товара.

Существует уже не мало примеров, когда после проведенного анализа покупок владельцам магазина удавалось увеличить продаж за счет правильной расстановки товара на прилавках, которая подтолкнет посетителя к покупке еще одного товара. Уже, наверное, всем известный пример сети Walmart, которая обнаружила зависимость

между продажами пива и памперсов, передвинув эти товары ближе друг к другу увеличили продажи и одного и второго товара.

Поиск частых графов или подграфов может понадобиться при анализе социальных сетей.

Социальная сеть - это социальная структура, состоящая из объектов (пользователей), представляющих собой взаимосвязанные узлы, которые отображают взаимоотношения между пользователями. Наиболее частым математическим представлением социальной сети является граф или матрица. Такое направление, как анализ социальных сетей (Social Network Analysis, SNA), применяет теорию графов, теорию информации и математической статистики к описанию и анализу социальных сетей. Так, одной из ключевых задач при этом становится идентификация некоторых групповых структур сети. Получается, что некоторое общество содержит внутри себя крайне плотно переплетенные между собой узлы, которые практически не связаны с остальной сетью.

Также поиск частотных графов может применяться в задаче прогнозирования токсичности и ряда других свойств химических соединений до их выхода на рынок.

В зависимости от применяемого алгоритма анализа графов предъявляются различные требования к данным. Наиболее часто подходящей структурой данных будет является матрица смежности. Некоторые алгоритмы могут основываться на списке ребер или хеш-таблицах.