

## Домашнее задание 4.

Осина Анна  
Пляскин Павел  
ИАД-2

### Оглавление

Задание 1. Поиск частых множеств. ....	2
Задание 1-а. ....	3
Задание 1-б. ....	4
Задание 1-в. ....	5
Задание 1-г. ....	6
Задание 2. Поиск ассоциативных правил. ....	8
Задание 2-а. ....	8
Задание 2-б. ....	9
Задание 2-в. ....	10
Задание 3. ....	12
Анализ сайтов новостной тематики. ....	12
Задание 3-а. ....	12
Задание 3-б. ....	12
Задание 3-в. ....	13
Задание 3-г. ....	14
Анализ сайтов финансовой тематики. ....	14
Задание 3-а. ....	14
Задание 3-в. ....	15
Задание 3-г. ....	15
Анализ сайтов образовательной тематики. ....	16
Задание 3-а. ....	16
Задание 3-б. ....	16
Задание 3-в. ....	17
Задание 3-г. ....	17

## Задание 1. Поиск частых множеств.

Мы имеем контекст  $K = (F, T, I_{FT})$ , где  $F$  – множество компаний (объектов), 2000 значений.  $T$  – множество словосочетаний (признаков), 3000 значений.  $f I t$  – фирма  $f$  (принадлежащая множеству  $F$ ) купила словосочетание  $t$  (из множества  $T$ ).

Исходные данные представлены в виде:

*Id\_словосочетания      Id\_компании      количество\_пар*

Из этого понятно, какая компания использует какое словосочетание в своей рекламе. При этом словосочетания могут использоваться несколькими компаниями одновременно.

Теперь для работы с SMPF нам необходимо получить данные, в которой каждая строка — это компания, и в этой строке записаны id словосочетаний использующиеся этой компанией.

Выполним эту обработку в Python:

```
In [2]: data = pd.read_csv('task1-2/phrase-company.txt', sep=' ', header=None, names=['phrase', 'company', 'count'])
```

```
In [3]: data['phrase'] = data['phrase'].apply(str)
```

```
In [4]: data.head(3)
```

```
Out[4]:
```

	phrase	company	count
0	0	23	1
1	0	96	1
2	0	188	1

```
In [5]: data = data.groupby('company')['phrase'].apply(lambda x: ' '.join(x))
```

```
In [6]: data.head(3)
```

```
Out[6]: company
0      332 345 355 663 665 674 681 691 696 719 730 96...
1      281 285 286 287 740 741 766 767 830 831 832 83...
2       22 33 129 218 249 250 251 287 646 654 655 657 ...
Name: phrase, dtype: object
```

```
In [7]: data.to_csv('task1-2/input.txt', index=None)
```

```
In [8]: data.shape
```

```
Out[8]: (2000,)
```

```
In [9]: print('minsup =', 35/2000)
```

```
minsup = 0.0175
```

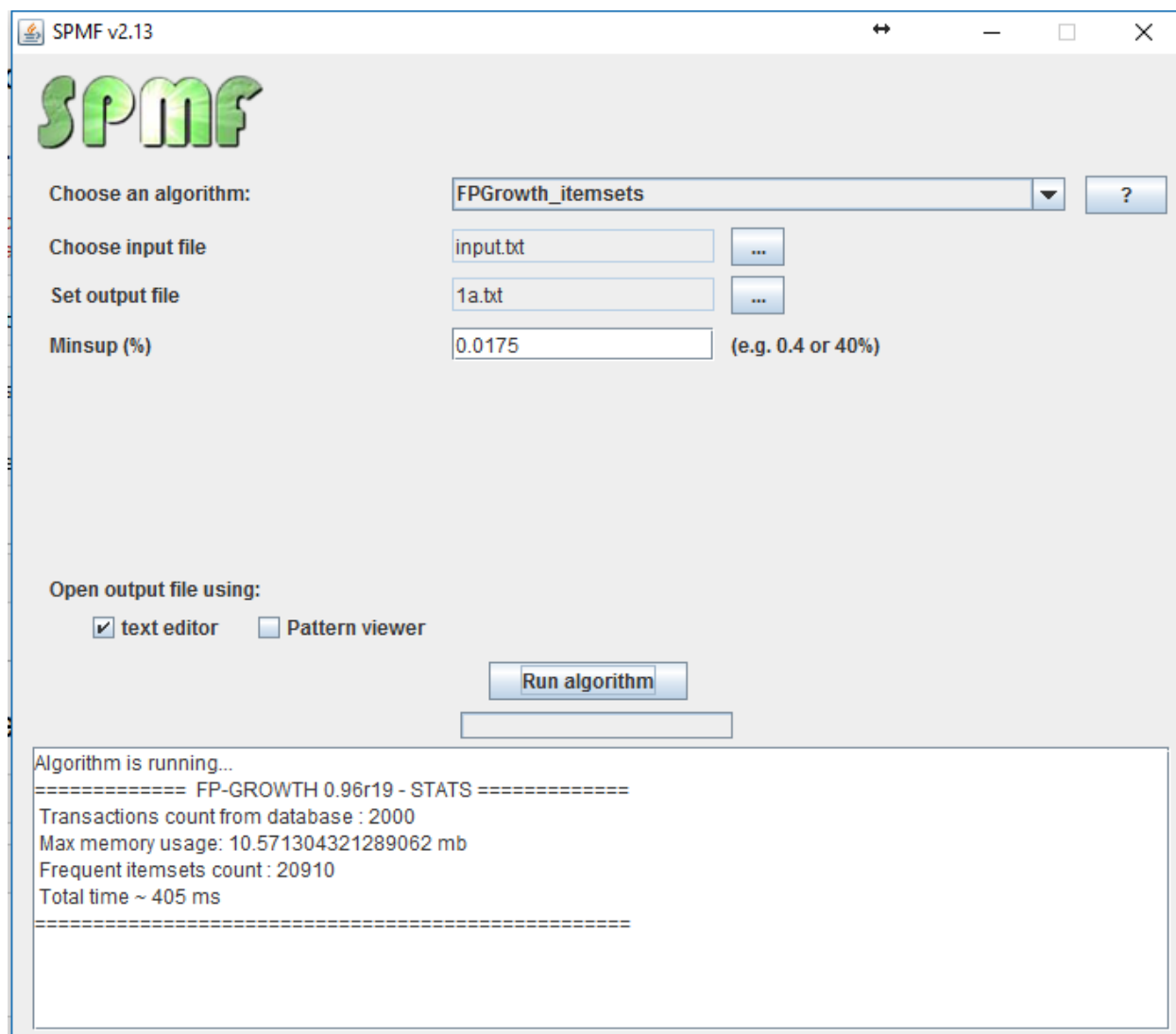
Мы получили данные в нужном нам формате, а также сразу вычислили необходимое значения минимальной поддержки требуемое в задании 1 и 2, по формуле

$$\text{minsupp} = \frac{|(A \cup B)'|}{|F|} = \frac{35}{2000} = 0.0175$$

## Задание 1-а.

Найдем частые множества для минимальной поддержки  $\text{minsupp}=35$ .

Для этого будем использовать SMPF с алгоритмом *FPGrowth\_itemsets*:

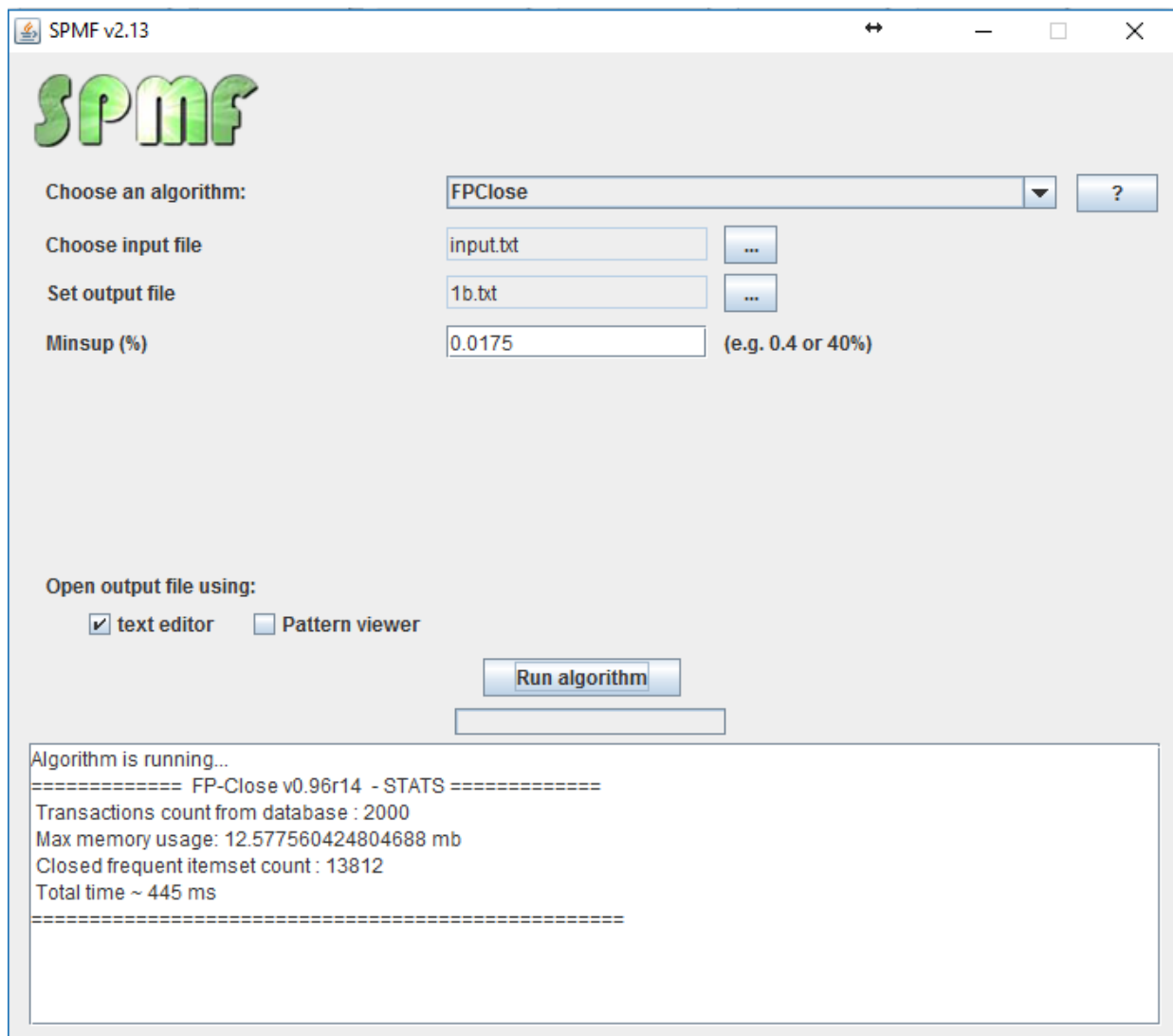


Количество найденных множеств, удовлетворяющие условиям = **20 910**.

## Задание 1-б.

Найдем частые замкнутые множества для минимальной поддержки  $\text{minsupp}=35$ .

Для этого будем использовать SMPF с алгоритмом *FPClose*:

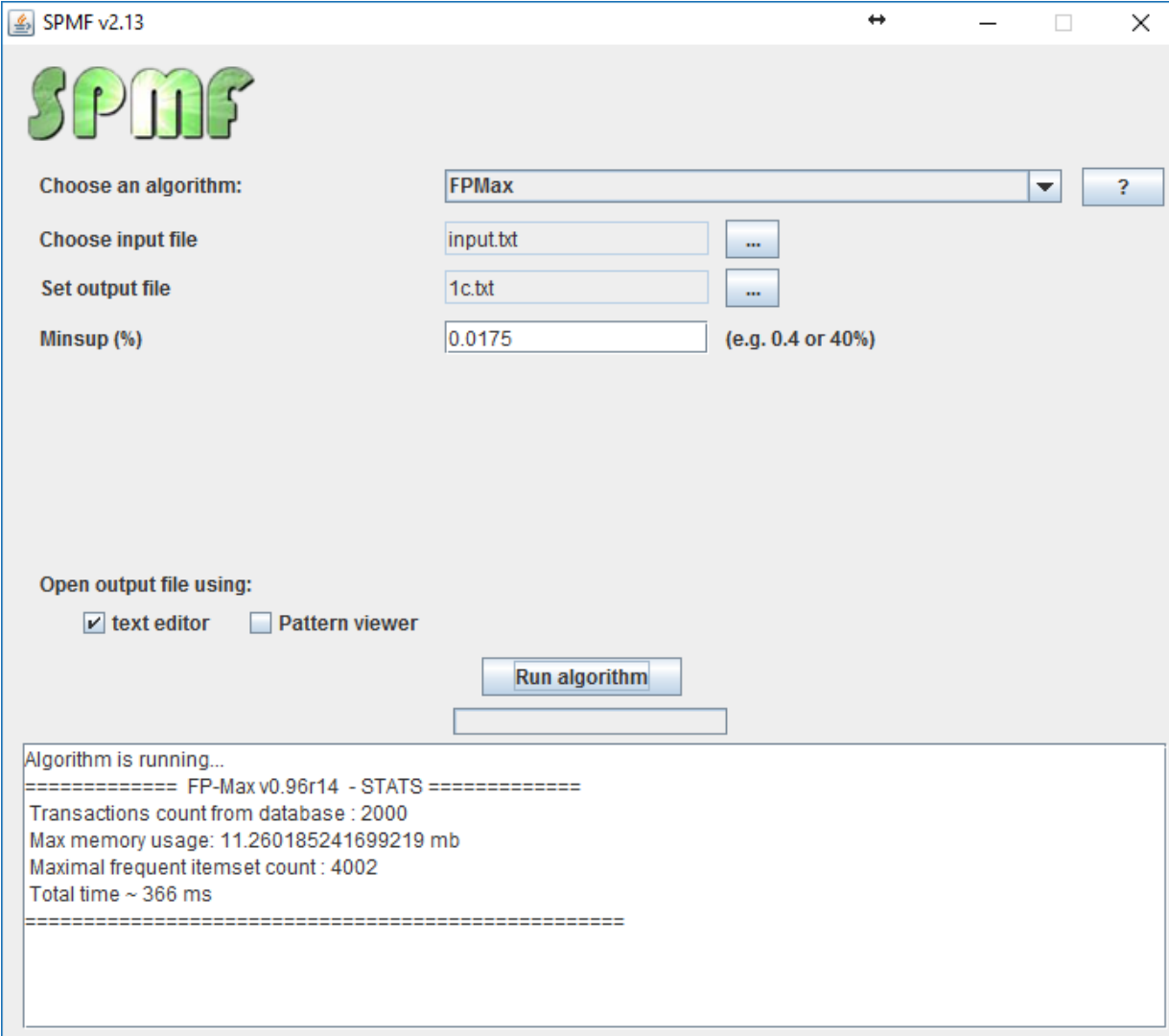


Количество найденных множеств, удовлетворяющие условиям = **13 812**.

## Задание 1-в.

Найдем частые максимальные множества для минимальной поддержки  $\text{minsupp}=35$ .

Для этого будем использовать SMPF с алгоритмом *FPMax*:



The screenshot shows the SMPF v2.13 application window. The title bar reads "SMPF v2.13". The main interface includes a logo, a dropdown menu for "Choose an algorithm:" set to "FPMax", and buttons for "input.txt" and "1c.txt" for file selection. The "Minsup (%)" field is set to "0.0175" with a note "(e.g. 0.4 or 40%)". Below these are checkboxes for "text editor" (checked) and "Pattern viewer". A "Run algorithm" button is present. At the bottom, a text box displays the following output:

```
Algorithm is running...
===== FP-Max v0.96r14 - STATS =====
Transactions count from database : 2000
Max memory usage: 11.260185241699219 mb
Maximal frequent itemset count : 4002
Total time ~ 366 ms
=====
```

Количество найденных множеств, удовлетворяющие условиям = **4 002**.

## Задание 1-г.

Посмотрим на примеры найденных множеств размером около 10 словосочетаний:

Из пункта А:

```
In [61]: a1 = pd.read_csv('task1-2/1a.txt', sep='#SUP:', names=['phrases', 'SUP'])
a1['count_phrases'] = a1['phrases'].apply(lambda x: len(x.split()))
a1 = a1[a1['count_phrases'] == a1.count_phrases.max()]
a1
```

```
C:\ProgramData\Anaconda3\lib\site-packages\ipykernel\__main__.py:1: ParserWarning: Falling back to the 'python' engine because the 'c' engine does not support regex separators (separators > 1 char and different from '\s+' are interpreted as regex); you can avoid this warning by specifying engine='python'.
  if __name__ == '__main__':
```

```
Out[61]:
```

	phrases	SUP	count_phrases
3313	989 998 1001 1011 1016 1019 1021 1888 1906	35	9
3321	989 995 998 1001 1011 1016 1019 1021 1906	36	9
3512	989 995 1001 1011 1016 1019 1021 1888 1906	37	9
3681	989 995 998 1001 1011 1016 1021 1888 1906	36	9
5403	120 1074 1075 1170 1233 1471 2156 2159 2166	35	9
17062	969 989 995 998 1011 1021 1882 1888 1906	36	9
17236	969 989 995 998 1001 1011 1021 1888 1906	35	9
17255	969 989 995 998 1001 1011 1021 1882 1906	35	9
18089	969 989 995 1001 1011 1021 1882 1888 1906	37	9

```
In [59]: for p in a1.loc[3313]['phrases'].split():
print (''.join(phrases_list[int(p)]))
```

```
casino gambling
casino game
casino game online
casino internet
casino line
casino net
casino online
gambling internet
gambling online
```

Из пункта Б:

```
In [62]: b1 = pd.read_csv('task1-2/1b.txt', sep='#SUP:', names=['phrases', 'SUP']);
b1['count_phrases'] = b1['phrases'].apply(lambda x: len(x.split()))
b1 = b1[b1['count_phrases'] == b1.count_phrases.max()]
b1
```

```
C:\ProgramData\Anaconda3\lib\site-packages\ipykernel\__main__.py:1: ParserWarning: Falling back to the 'python' engine because the 'c' engine does not support regex separators (separators > 1 char and different from '\s+' are interpreted as regex); you can avoid this warning by specifying engine='python'.
  if __name__ == '__main__':
```

```
Out[62]:
```

	phrases	SUP	count_phrases
2121	1021 1906 1888 1011 989 1001 998 1019 1016	35	9
2124	1021 1906 995 1011 989 1001 998 1019 1016	36	9
2149	1021 1906 995 1888 1011 989 1001 1019 1016	37	9
2252	1021 1906 995 1888 1011 989 1001 998 1016	36	9
3141	2166 2159 2156 120 1233 1075 1170 1074 1471	35	9
10810	1021 1906 995 969 1888 1011 989 1001 998	35	9
10822	1021 1906 995 1882 969 1011 989 1001 998	35	9
10927	1021 1906 995 1882 969 1888 1011 989 998	36	9
11335	1021 1906 995 1882 969 1888 1011 989 1001	37	9

```
In [70]: for p in b1.loc[3141]['phrases'].split():
print (''.join(phrases_list[int(p)]))
```

```
hosting web
hosting site web
hosting services web
affordable hosting web
cost hosting low web
cheap hosting web
company hosting web
cheap hosting site web
discount hosting web
```

## Из пункта В:

```
In [67]: c1 = pd.read_csv('task1-2/1c.txt', sep='#SUP:', names=['phrases', 'SUP']);
c1['count_phrases'] = c1['phrases'].apply(lambda x: len(x.split()))
c1 = c1[c1['count_phrases'] == c1.count_phrases.max()]
c1
```

```
C:\ProgramData\Anaconda3\lib\site-packages\ipykernel\_main_.py:1: ParserWarning: Falling back to the 'python' engine because the 'c' engine does not support regex separators (separators > 1 char and different from '\s+' are interpreted as regex); you can avoid this warning by specifying engine='python'.
if __name__ == '__main__':
```

```
Out[67]:
```

	phrases	SUP	count_phrases
1061	1021 1906 1888 1011 989 1001 998 1019 1016	35	9
1062	1021 1906 995 1011 989 1001 998 1019 1016	36	9
1067	1021 1906 995 1888 1011 989 1001 1019 1016	37	9
1102	1021 1906 995 1888 1011 989 1001 998 1016	36	9
1386	2166 2159 2156 120 1233 1075 1170 1074 1471	35	9
3505	1021 1906 995 969 1888 1011 989 1001 998	35	9
3508	1021 1906 995 1882 969 1011 989 1001 998	35	9
3512	1021 1906 995 1882 969 1888 1011 989 998	36	9
3553	1021 1906 995 1882 969 1888 1011 989 1001	37	9

```
In [69]: for p in c1.loc[1061]['phrases'].split():
print (''.join(phrases_list[int(p)]))
```

```
casino online
gambling online
gambling internet
casino internet
casino gambling
casino game online
casino game
casino net
casino line
```

Нетрудно заметить, что словосочетания в большинстве множеств говорят о сфере онлайн казино. Также могут встречаться словосочетания, касающиеся некоего дешевого хостинга.

Также если просмотреть множества длиной 8 или 7, то эта тенденция сохраняется. Большинство множеств состоят из словосочетаний, относящихся к онлайн казино, появляется чуть больше словосочетаний, относящихся к хостингу. Также среди множеств длиной 7, встречаются словосочетания про домашний бизнес.

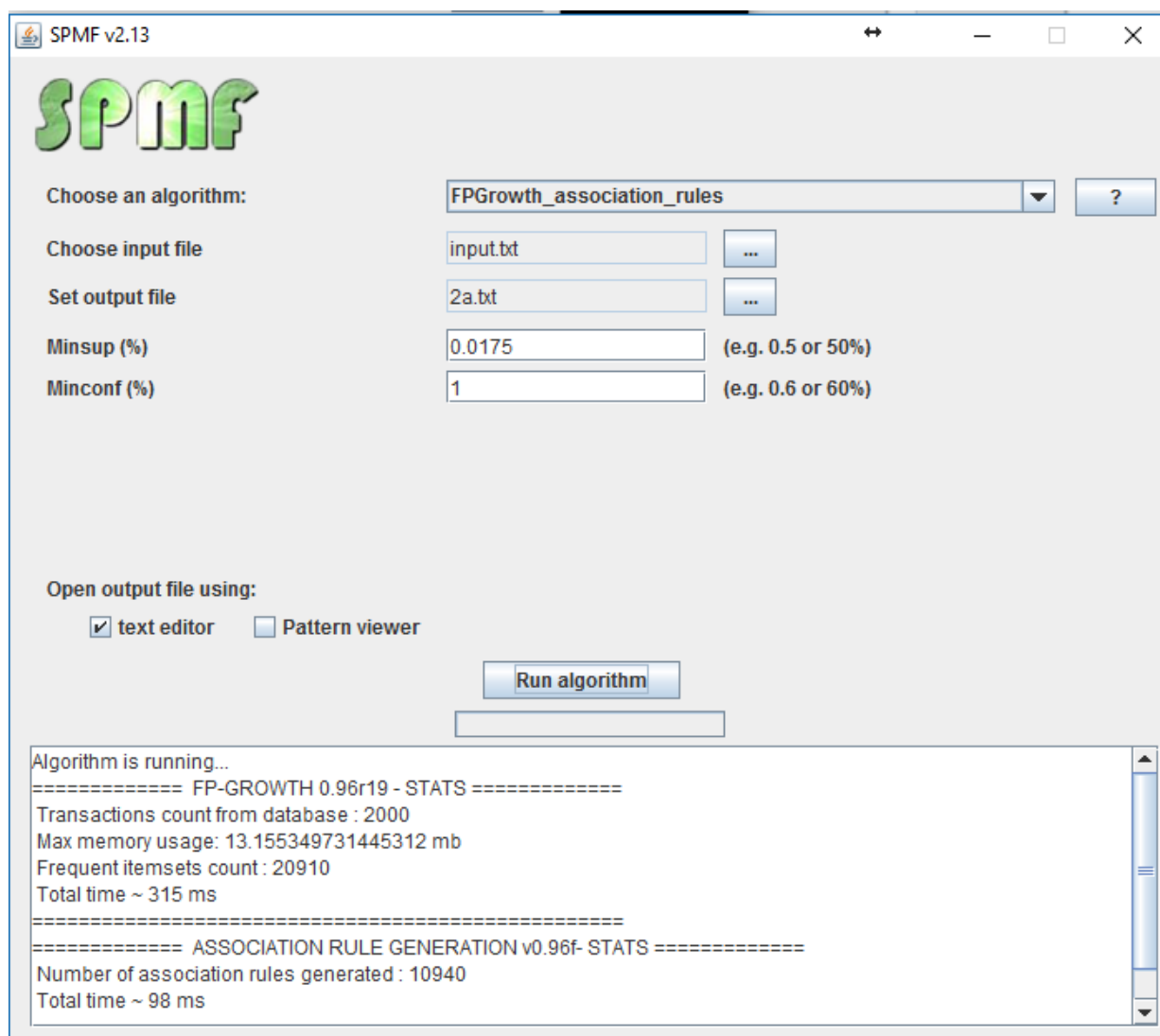
Стоит заметить, что при любой длине множеств наибольшее значение поддержки имеют множества, словосочетания в которых относятся к сфере онлайн казино.

## Задание 2. Поиск ассоциативных правил.

### Задание 2-а.

Найдем ассоциативные правила для минимальной поддержки  $\text{minsupp} = 35$  и  $\text{minconf} = 1$ .

Для этого будем использовать SMPF с алгоритмом *FPGrowth\_association\_rules*:



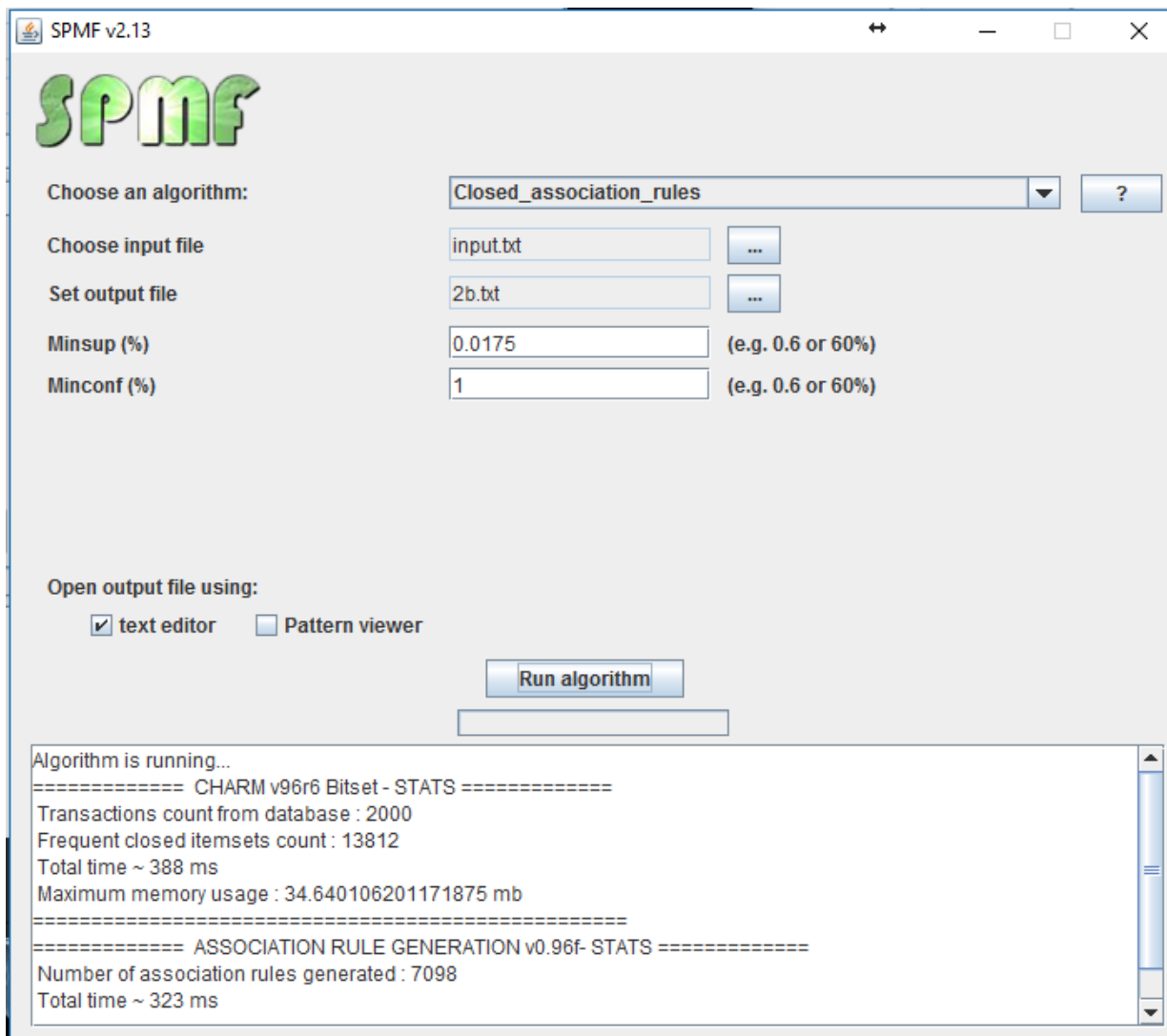
Количество найденных правил, удовлетворяющие условиям = **10 940**.



## Задание 2-б.

Найдем замкнутые ассоциативные правила для минимальной поддержки  $\text{minsupp} = 35$  и  $\text{minconf} = 1$ .

Для этого будем использовать SMPF с алгоритмом *Closed\_association\_rules*:

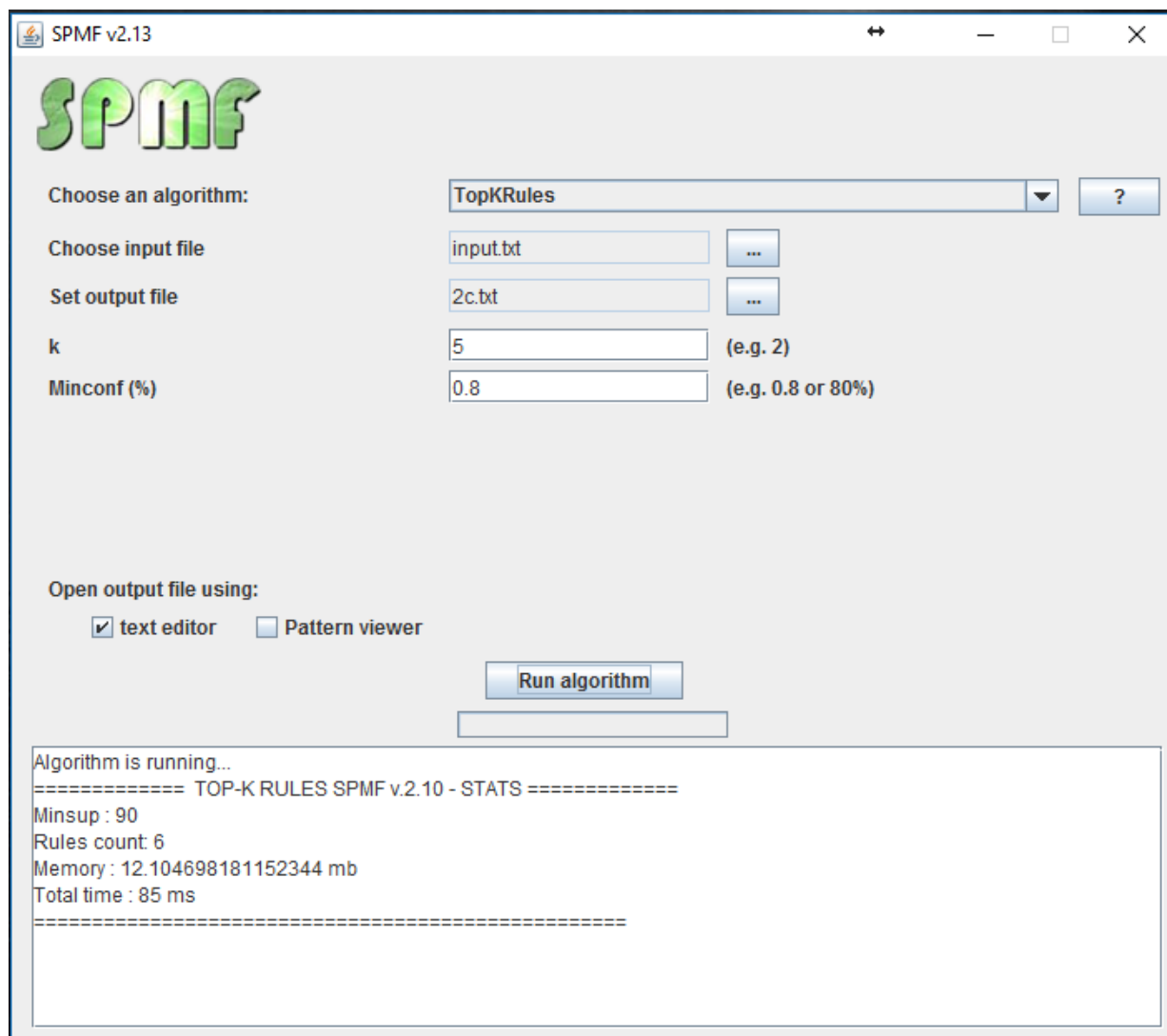


Количество найденных правил, удовлетворяющие условиям = **7 098**.

## Задание 2-в.

Найдем 5 самых частых правил для  $\text{minconf} = 0,8$ .

Для этого будем использовать SMPF с алгоритмом `Closed_association_rules`:



Количество найденных правил, удовлетворяющие условиям = **6**.

Найденные правила:

- 663 674 ==> 345 #SUP: 90 #CONF: 0.8256880733944955
- 345 674 ==> 663 #SUP: 90 #CONF: 0.8490566037735849
- 2536 ==> 2336 #SUP: 91 #CONF: 0.8666666666666667
- 355 ==> 345 #SUP: 102 #CONF: 0.8292682926829268
- 355 ==> 674 #SUP: 105 #CONF: 0.8536585365853658
- 2159 ==> 2166 #SUP: 109 #CONF: 0.8074074074074075

Обработаем их в Python и посмотрим, что именно это за словосочетания:

```
In [27]: phrases = pd.read_csv('task1-2\phrases.txt', header = None, names=['phrase']) #фразы
phrases_list = phrases.values.flatten()
```

```
In [28]: print (' "{0}" "{1}" ==> "{2}"'.format(phrases_list[663], phrases_list[674], phrases_list[345]))
print (' "{0}" "{1}" ==> "{2}"'.format(phrases_list[345], phrases_list[674], phrases_list[663]))
print (' "{0}" ==> "{1}"'.format(phrases_list[2536], phrases_list[2336]))
print (' "{0}" ==> "{1}"'.format(phrases_list[355], phrases_list[345]))
print (' "{0}" ==> "{1}"'.format(phrases_list[355], phrases_list[674]))
print (' "{0}" ==> "{1}"'.format(phrases_list[2159], phrases_list[2166]))
```

```
"business home" "business home opportunity" ==> "based business home"
"based business home" "business home opportunity" ==> "business home"
"marketing online" ==> "internet marketing"
"based business home opportunity" ==> "based business home"
"based business home opportunity" ==> "business home opportunity"
"hosting site web" ==> "hosting web"
```

Ассоциативные правила указывают на очень похожие словосочетания, некоторые из них буквально используют синонимичные слова. Также найденные ассоциации являются крайне близкими по смыслу к исходным словосочетаниям.

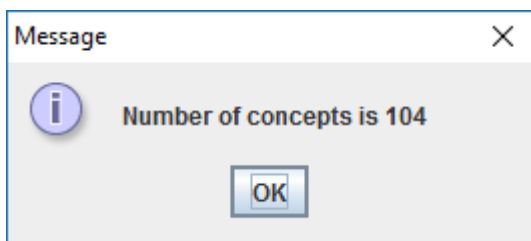
Если рассмотреть это, например, со стороны поисковых запросов, которые будет использовать какой-то пользователь, то с достоверностью больше 0.8 он попробует переформулирует запрос в том виде, в каком указывает нам ассоциативное правило.

### Задание 3.

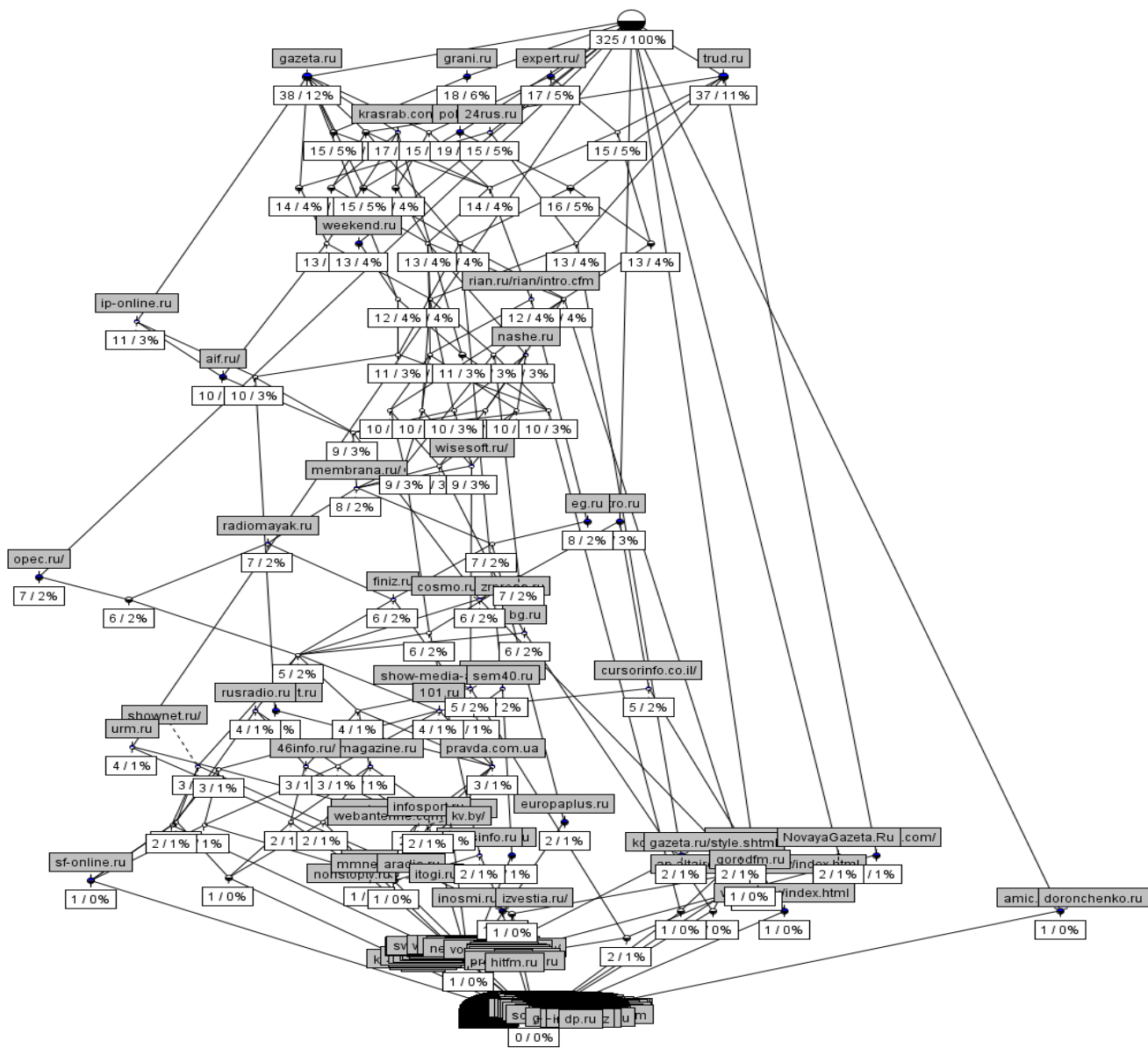
## Анализ сайтов новостной тематики

### Задание 3-а.

Сократив количество объектов до 325 и количество сайтов (признаков) до 35, мы получим 104 понятия.



### Задание 3-б.

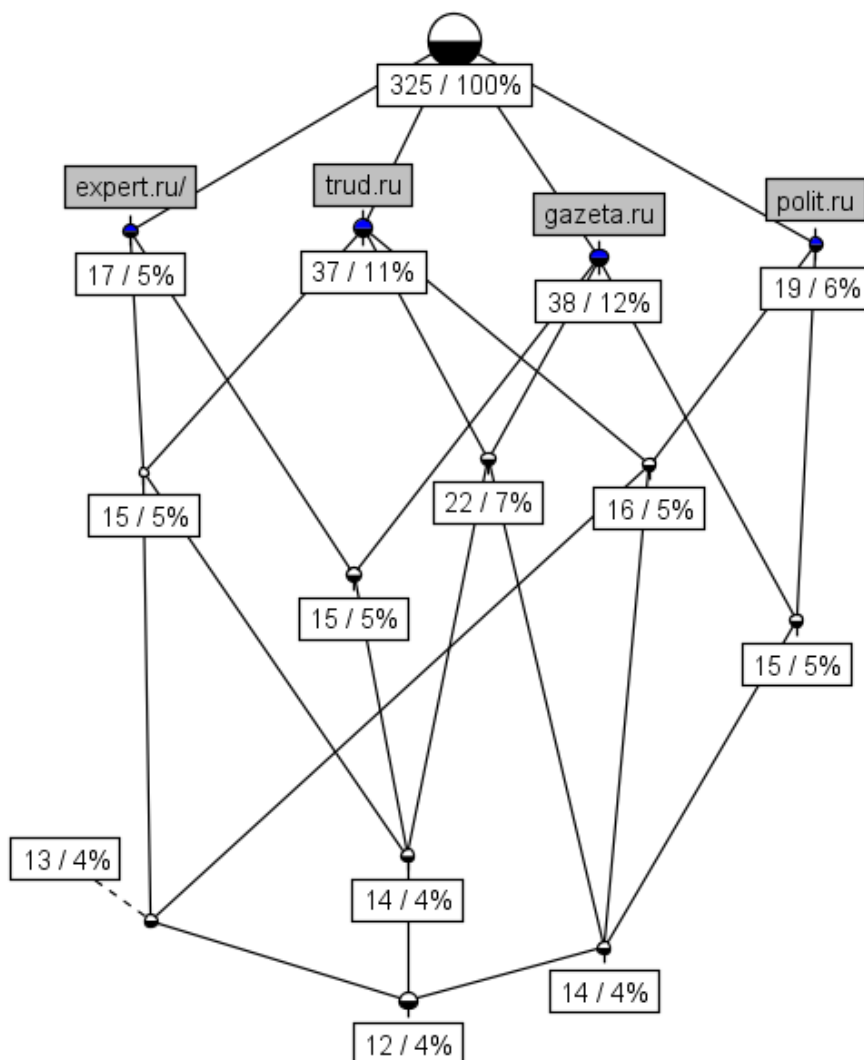


### Задание 3-в.

Построим решетку понятий. Однако сделаем это не для всех сайтов, а только для тех, которые оказались более или менее популярными, так как со всеми решетка получается очень сложно читаемой. Например, оставим сайты:

- expert.ru/
- trud.ru
- gazeta.ru
- polit.ru

Решетка выглядит следующим образом:



1. 15, *expert.ru trud.ru* – 15 человек (5% от общего числа объектов) посетили сайты ВШЭ, expert.ru и trud.ru
2. 22, *trud.ru gazeta.ru* – 22 человека (7% от общего числа объектов) посетили сайты ВШЭ, trud.ru и gazeta.ru
3. 14, *expert.ru trud.ru expert.ru trud.ru* – 14 человек (4% от общего числа объектов) посетили сайты ВШЭ, expert.ru, trud.ru expert.ru и trud.ru
4. 12, *expert.ru trud.ru expert.ru trud.ru polit.ru* – 12 человек (4% от общего числа объектов) посетили сайт ВШЭ и все сайты, которые присутствуют на данной решетке (expert.ru, trud.ru expert.ru, trud.ru, polit.ru)

### Задание 3-г.

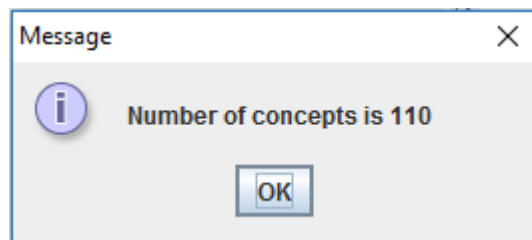
Пример импликаций:

- <14> trud.ru gazeta.ru expert.ru/ ==> grani.ru;      Поддержка = 14
- <13> expert.ru/ polit.ru ==> trud.ru;      Поддержка = 13

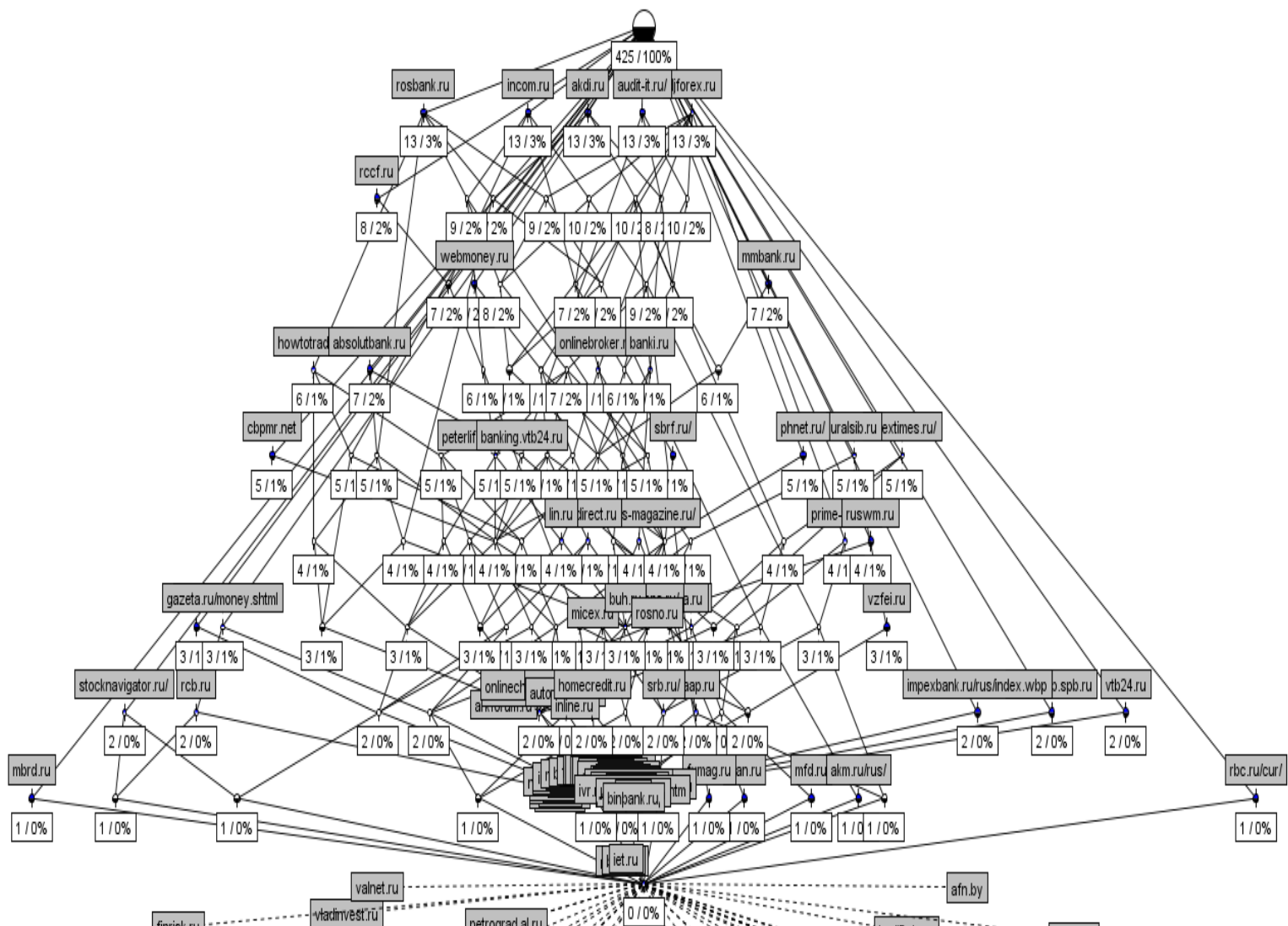
### Анализ сайтов финансовой тематики

#### Задание 3-а.

Сократив количество объектов до 425 и количество сайтов (признаков) до 136, мы получим 110 понятий.



#### Задание 3-б.

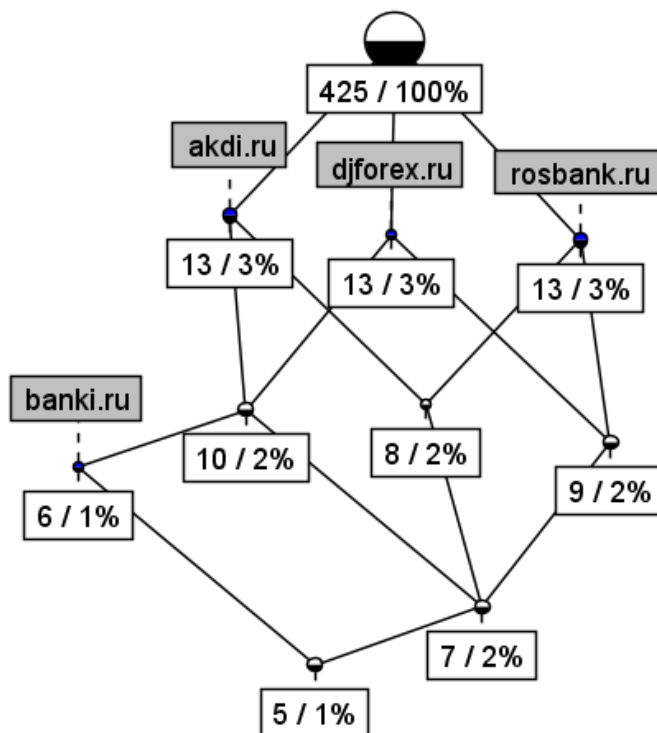


### Задание 3-в.

Построим решетку понятий. Снова сделаем это не для всех сайтов, а только для некоторых, выбранных случайно.

Например, оставим сайты:

- akdi.ru
- djforex.ru
- rosbank.ru
- banki.ru



1. 10, *akdi.ru djforex.ru* – 10 человек, посетивших сайт ВШЭ, *akdi.ru* и *djforex.ru*
2. 7, *akdi.ru djforex.ru rosbank.ru* – 7 человек, посетивших сайт ВШЭ, *akdi.ru*, *djforex.ru* и *rosbank.ru*
3. 6, *banki.ru akdi.ru djforex.ru* – 6 человек посетивших ВШЭ, *akdi.ru*, *djforex.ru* и *banki.ru*. При этом стоит заметить, что все посетители, посещавшие *banki.ru* посещали и *akdi.ru* и *djforex.ru*.
4. 5, *banki.ru akdi.ru djforex.ru rosbank.ru* – 5 человек посетили ВШЭ и все сайты, рассматриваемые на этой решетке.

### Задание 3-г.

Пример импликаций:

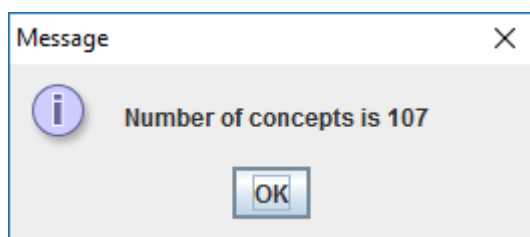
- <6> *rosbank.ru rccf.ru* ==> *incom.ru*; Поддержка = 6
- <5> *djforex.ru rosbank.ru howtotrade.ru* ==> *incom.ru audit-it.ru*/; Поддержка=5



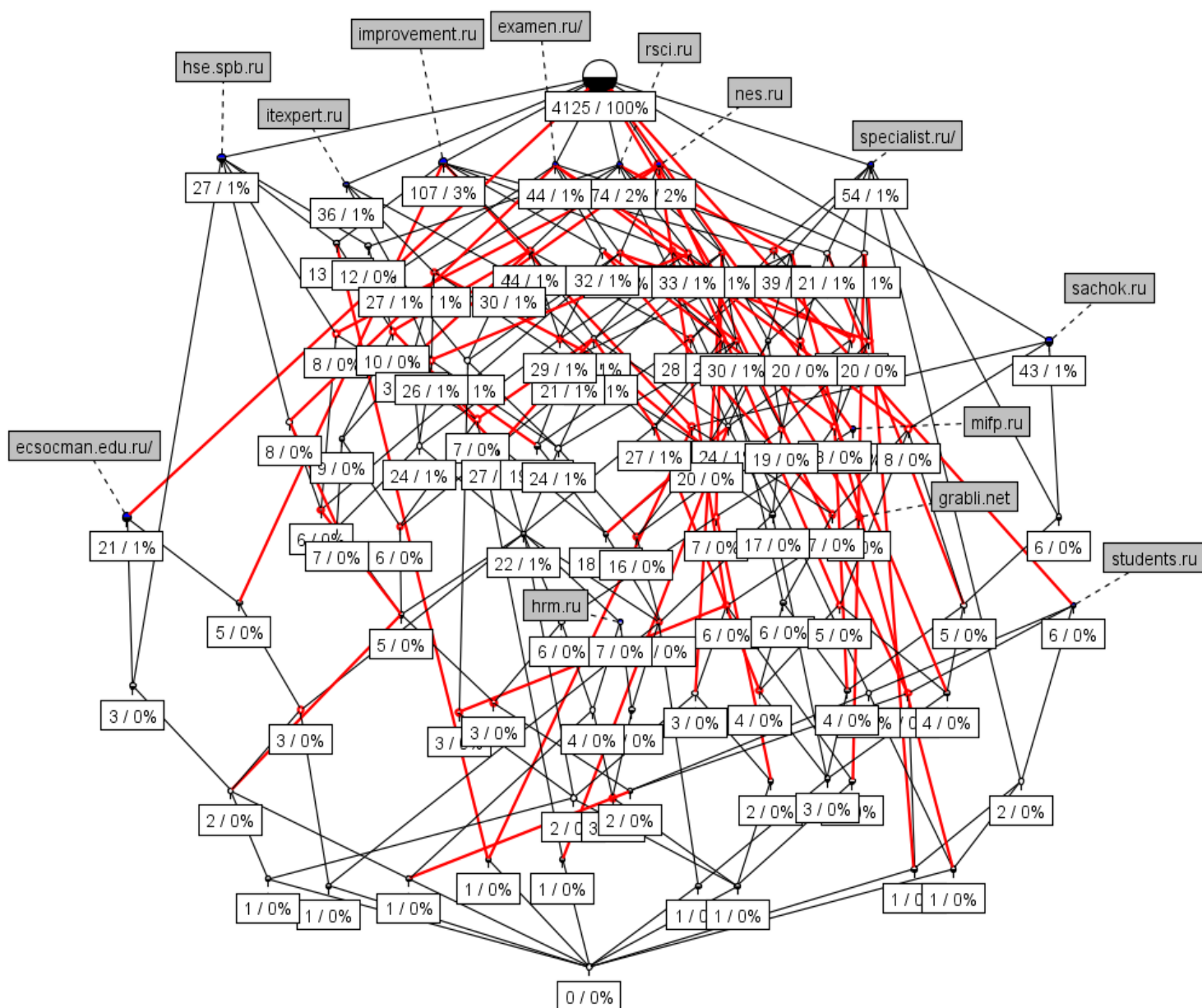
## Анализ сайтов образовательной тематики

### Задание 3-а.

Сократив количество сайтов (признаков) до 13, мы получим 107 понятий.



### Задание 3-б.



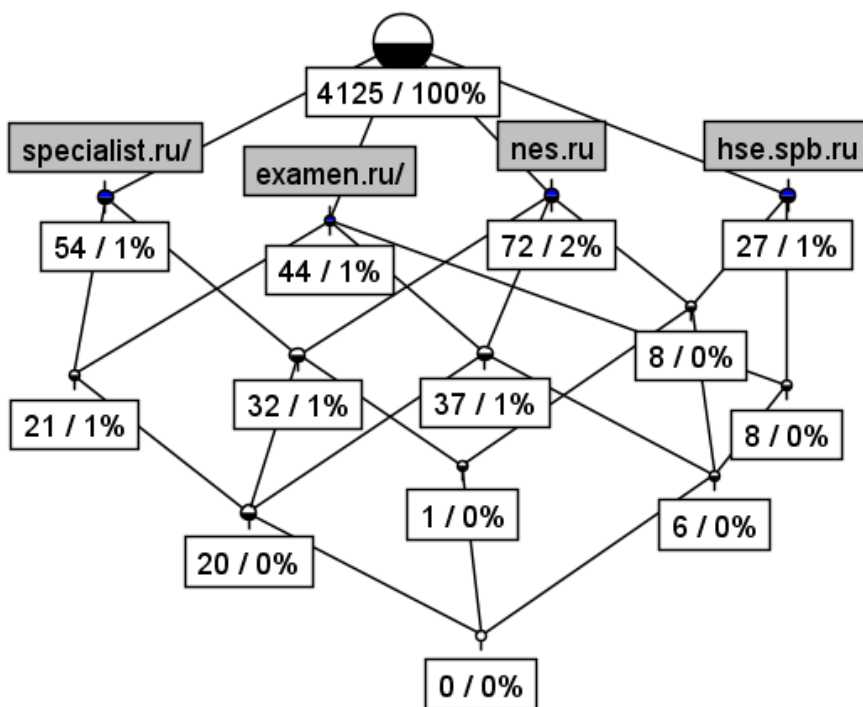


### Задание 3-в.

Построим решетку понятий. Сделаем это аналогично предыдущим контекстам.

Оставим сайты:

- specialist.ru
- nes.ru
- hse.spb.ru
- examen.ru



1. 32, *specialist.ru nes.ru* – 32 человека, посетили сайты ВШЭ, specialist.ru и nes.ru.

2. 20, *specialist.ru nes.ru examen.ru* – 20 человек, посетивших сайты ВШЭ, specialist.ru, nes.ru и examen.ru

3. 0, *specialist.ru nes.ru examen.ru hse.spb.ru* – ни один человек не посетил все сайты из рассматриваемой решетки.

### Задание 3-г.

Примеры импликаций:

- $\langle 31 \rangle$  improvement.ru itexpert.ru  $\implies$  rsci.ru;      Поддержка = 31
- $\langle 7 \rangle$  mifp.ru rsci.ru  $\implies$  nes.ru;      Поддержка = 7