Produced by YE Qianying, GAO Ge, ZHAO Feier

# Your Nose Never Know

## Mysterious Network Behind The World of Perfume

# Contents


01 Background
02 Data Collection
03 Data Cleaning
04 Visualization
05 Results & Conclusions


# 01 Background


# About Perfume

*Perfume is closely linked to urbanites' daily life*

a. *Influences our* ***sense of smell****.*

b. *Influences our* ***mood state****.*

c. *For consideration of* ***social needs****, people would like to wear perfumes in both formal and informal situations.*

d. *Different fragrances show different* ***personalities****.*

e. ***Fashion items*** *for women and men.*

# Attitudes towards Perfume

**Chanel**

*"I wear nothing but a few drops of Chanel No.5."*


**Calvin Klein**

*"Be good, be bad, just be yourself."*

**Dior**

*"Gold is cold. Diamonds are dead. A Limousine is a car. Don't pretend.Feel what's real."*

**Burberry**

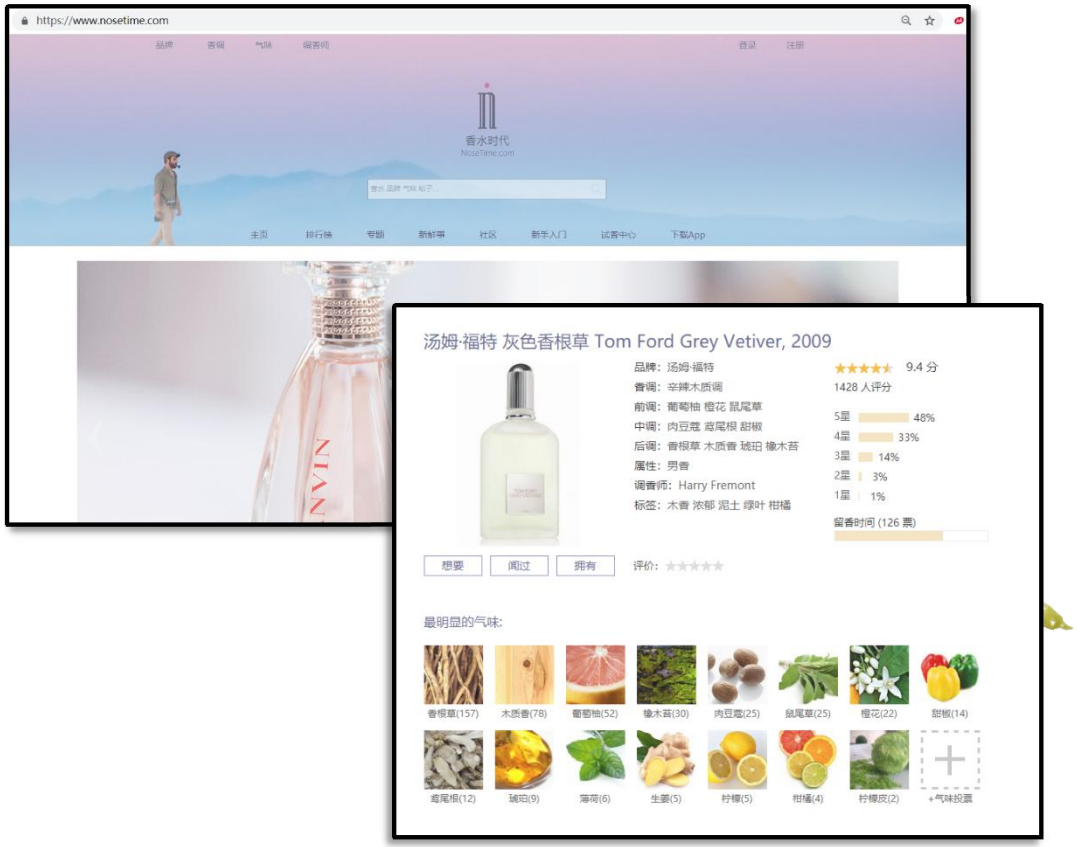
*"The good things in life never change."*


**Estee Lauder**

*"Is staying in touch the same as being in touch?"*

*Thousands of perfumes have been created by cosmetic manufacturers as* ***"name cards"*** *for consumers to choose, which also arouses our interests to explore more about* ***the world of perfume****...*

# 02 Data Collection




*https://www.nosetime.com*

# Data Source

## Nosetime 香水时刻

*We used python to get the information of all brands' products.*

*a. Got all information of* ***2040 brands****.*

*b. Got* ***2226+1441=3667 urls****.*

*c. Variables: ingredients,brand names, three notes (top notes, middle notes, base notes), years of perfumes, scores of perfumes, reply number of perfumes...*

*Here is what we used.*

In [4]:

```
from selenium import webdriver
import time
import re
from lxml import etree
from bs4 import BeautifulSoup
import bs4 as bs
from selenium.common.exceptions import NoSuchElementException
from selenium.webdriver.common.by import By
from selenium.webdriver.support.ui import WebDriverWait
from selenium.webdriver.support import expected_conditions as EC
import pandas as pd

URLs_brand = []
#browser = webdriver.Firefox()#Chrome('./chromedriver.exe')
whole = ['2-a','3-b','4-c','5-d','6-e','7-f','8-g','9-h','10-i','11-j','12-k','13-l','14-m','15-n','16-o','17-p',
         '18-q','19-r','20-s','21-t','22-u','23-v','24-w','25-x','26-y','27-z']

# For every page in the interval
for wh in whole:
    PATIENCE_TIME = 60
    driver = webdriver.Chrome()
    driver.get('https://www.nosetime.com/pinpai/'+str(wh)+'.html')
    driver.encoding = 'utf-8'
    driver.maximize_window()

    html = driver.page_source
    soup = BeautifulSoup(html,'lxml')
    url_containers = soup.find_all('div', class_='odorlist')

    for container in url_containers:
        #find all the date of the news
        try:
            for url_b in container.find_all('a'):
                url_brand = url_b.get('href')
                URLs_brand.append(url_brand)
        except:
            url_brand = "no_url"
            URLs_brand.append(url_brand)
    driver.close()
driver.quit()
```

In [39]:

```
#get the pure id of brand
Brand_id = []
for i in Cleaned_URls_brand:
    brand_id = re.sub(r'[^0-9]','',i)
    Brand_id.append(brand_id)
datapage1 = {
    'Brand_id': Brand_id,
}
brandid = pd.DataFrame.from_dict(datapage1, orient ='index').transpose()
print(brandid.info())
brandid.to_csv('Brand_id.csv',index = False, encoding = 'utf_8_sig')
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2040 entries, 0 to 2039
Data columns (total 1 columns):
Brand_id    2040 non-null object
dtypes: object(1)
memory usage: 16.0+ KB
None
```

*That is what we have got.*

**Got urls according to the brand names.**

*That is what we have got.*

```python
from selenium import webdriver
import time
import re
from lxml import etree
from bs4 import BeautifulSoup
import bs4 as bs
from selenium.common.exceptions import NoSuchElementException
from selenium.webdriver.common.by import By
from selenium.webdriver.support.ui import WebDriverWait
from selenium.webdriver.support import expected_conditions as EC
import pandas as pd

Names_perfume = []
Whole_info = []
Score = []
#browser = webdriver.Firefox()#Chrome('./chromedriver.exe')
for url in URLs:
    PATIENCE_TIME = 60 b
    driver = webdriver.Chrome()
    driver.get(url)
    driver.encoding = 'utf-8'
    driver.maximize_window()

    html = driver.page_source
    soup = BeautifulSoup(html,'lxml')
    perfume_containers = soup.find_all('div', class_='item')

    for container in perfume_containers:
        #find all the date of the news
        try:
            for name in container.find_all('h2'):
                Names_perfume.append(name.text)
        except:
            name = "no_date"
            Names_perfume.append(name)

        #find all the ingredients of the perfumes
        try:
            for whole_info in container.find_all('div',class_='info'):
                Whole_info.append(whole_info.text)

        except:
            whole_info = "no_whole_info"
            Whole_info.append(whole_info)

        #find all the scores of the perfumes
        try:
            for score in container.find_all('div',class_='score'):
                Score.append(score.text)
        except:
            score = "no_score"
            Score.append(score.text)
    driver.close()
driver.quit()
```

```
In [3]: print(Score)

[' 8.1 分(702 评价)', ' 8.8 分(1498 评价)', ' 9.2 分(1508 评价)', ' 8.8 分(1359 评价)', ' 8.9 分(1289 评价)', ' 8.9 分(1232 评价)', ' 9.3 分
(560 评价)', ' 8.7 分(732 评价)', ' 9.5 分(343 评价)', ' 9.2 分(461 评价)', ' 8.6 分(449 评价)', ' 8.6 分(98 评价)', ' 8.6 分(95 评价)', ' 9
分(244 评价)', ' 9 分(358 评价)', ' 8.6 分(506 评价)', ' 8.5 分(88 评价)', ' 9.3 分(376 评价)', ' 8.7 分(362 评价)', ' 8.9 分(416 评价)', '
8.8 分(587 评价)', ' 9.3 分(252 评价)', ' 9.3 分(144 评价)', ' 8.6 分(305 评价)', ' 9 分(199 评价)', ' 9 分(158 评价)', ' 7.9 分(143 评价)',
' 8.5 分(255 评价)', ' 8.3 分(216 评价)', ' 9.3 分(93 评价)', ' 8.5 分(290 评价)', ' 8.9 分(166 评价)', ' 9 分(295 评价)', ' 8.2 分(271 评
价)', ' 8.8 分(109 评价)', ' 8.8 分(108 评价)', ' 8.6 分(360 评价)', ' 8 分(91 评价)', ' 8.9 分(156 评价)', ' 8.1 分(153 评价)', ' 8.2 分(96
评价)', ' 8.6 分(94 评价)', ' 8.9 分(61 评价)', ' 8.8 分(45 评价)', ' 8.8 分(116 评价)', ' 8.6 分(68 评价)', ' 9.1 分(126 评价)', ' 8.5 分(8
3 评价)', ' 8.3 分(69 评价)', ' 8.1 分(61 评价)', ' 8.6 分(46 评价)', ' 8.7 分(180 评价)', ' 9.1 分(99 评价)', ' 8.1 分(58 评价)', ' 8.4 分
(42 评价)', ' 8.5 分(201 评价)', ' 8.4 分(180 评价)', ' 8.7 分(98 评价)', ' 7.8 分(82 评价)', ' 8.6 分(70 评价)', ' 8.3 分(65 评价)', ' 8.6
分(37 评价)', ' 8.6 分(34 评价)', ' 8.6 分(73 评价)', ' 8.2 分(53 评价)', ' 8.4 分(41 评价)', ' 8.4 分(34 评价)', ' 8.1 分(16 评价)', ' 7.8
分(16 评价)', ' 8.2 分(289 评价)']

In [31]: Score_number=[]
         for i in Score:
             i = i.split("分(")
             i = i[0]
             Score_number.append(i)

In [32]: print(Score_number)

[' 8.1 ', ' 8.8 ', ' 9.2 ', ' 8.8 ', ' 8.9 ', ' 8.9 ', ' 9.3 ', ' 8.7 ', ' 9.5 ', ' 9.2 ', ' 8.6 ', ' 8.6 ', ' 8.6 ', ' 9 ', ' 9 ', ' 8.6 ',
' 8.5 ', ' 9.3 ', ' 8.7 ', ' 8.9 ', ' 8.8 ', ' 9.3 ', ' 9.3 ', ' 8.6 ', ' 9 ', ' 9 ', ' 7.9 ', ' 8.5 ', ' 8.3 ', ' 9.3 ', ' 8.5 ', ' 8.9 ',
' 9 ', ' 8.2 ', ' 8.8 ', ' 8.8 ', ' 8.6 ', ' 8 ', ' 8.9 ', ' 8.1 ', ' 8.2 ', ' 8.6 ', ' 8.9 ', ' 8.8 ', ' 8.8 ', ' 8.6 ', ' 9.1 ', ' 8.5 ',
' 8.3 ', ' 8.1 ', ' 8.6 ', ' 8.7 ', ' 9.1 ', ' 8.1 ', ' 8.4 ', ' 8.5 ', ' 8.4 ', ' 8.7 ', ' 7.8 ', ' 8.6 ', ' 8.3 ', ' 8.6 ', ' 8.6 ', ' 8.6
', ' 8.2 ', ' 8.4 ', ' 8.4 ', ' 8.1 ', ' 7.8 ', ' 8.2 ']
```

```
In [5]: print(Whole[:10])

['前调：橘子，葡萄柚，黑加仑中调：晚香玉，依兰，茉莉，橙花后调：檀香木，麝香'，'气味：劳丹脂，肉豆蔻，生姜，檀香木，广藿香，薄荷，茉莉，葡萄柚，柠檬，香根
草，芬香，雪松，粉红胡椒'，'前调：橙子，橘子，橙花，香柠檬中调：含羞草，茉莉，玫瑰，依兰后调：零陵香豆，广藿香，红没药，香草，香根草，麝香'，'前调：柠檬，
雪松中调：粉红胡椒，凤梨兰，茉莉后调：柚木，瓷尾花，琥珀，广藿香，香根草，麝香'，'前调：酢，橙花油，依兰，香柠檬，柠檬中调：瓷尾花，瓷尾根，茉莉，铃兰，玫
瑰后调：琥珀，檀香木，广藿香，麝香，麝猫香，香草，橹木苔，香根草'，'前调：葡萄柚，木梨中调：茉莉，风信子后调：麝香，瓷尾花，雪松，琥珀'，'前调：橙花油，
香柠檬，绿叶中调：瓷尾花，水仙花，玫瑰，铃兰，依兰后调：香根草，檀香木，橹木苔，皮革'，'前调：蒲萝，瓷尾花，广藿香，粉红胡椒，风信子中调：茉莉，柠檬后
调：麝香，广藿香，香草，香根草'，'气味：香根草，檀香木，醛，烟草，紫罗兰，柏树，杜松，粉红胡椒，辛香料'，'前调：柠檬，橘子，橙子，橙花油，醛，香柠檬，青柠中
调：玫瑰，茉莉，依兰后调：雪松，麝香，香草，瓷尾根']

In [15]: Alone = []
         Main = []
         for i in Whole:
             if re.findall(r'^气味',i):
                 alone = i.replace(",","/n")
                 Alone.append(alone)
             else:
                 Main.append(i)

In [16]: print(Alone[:10])

['气味：劳丹脂/n肉豆蔻/n生姜/n檀香木/n广藿香/n薄荷/n茉莉/n葡萄柚/n柠檬/n香根草/n琴香/n雪松/n粉红胡椒'，'气味：香根草/n檀香木/n醛/n烟草/n紫罗
兰/n柏树/n杜松/n粉红胡椒/n辛香料'，'气味：劳丹脂/n肉豆蔻/n生姜/n檀香木/n广藿香/n香根草/n木质香/n薄荷/n茉莉/n柠檬/n葡萄柚/n琴香/n雪松/n粉红胡
椒/n琥珀'，'气味：香根草/n檀香木/n醛/n烟草/n紫罗兰/n柏树/n杜松/n粉红胡椒/n辛香料'，'气味：檀香木/n零陵香豆/n琥珀/n香根草/n麝香/n香柠檬/n雪
松/n香草/n柠檬/n粉红胡椒'，'气味：盖衣草/n鹏香/n香草/n野花/n青草/n玫瑰/n茉莉/n零陵香豆'，'气味：粉红胡椒/n茉莉/n广藿香/n香草/n麝香'，'气味：
瓷尾根/n玫瑰/n皮革'，'气味：蜂蜜/n小花兰/n鸡蛋花/n山楂'，'气味：绿叶/n皮革/n青草/n花香']

In [17]: print(Main[:10])

['前调：橘子，葡萄柚，黑加仑中调：晚香玉，依兰，茉莉，橙花后调：檀香木，麝香'，'前调：橙子，橘子，橙花，香柠檬中调：含羞草，茉莉，玫瑰，依兰后调：零陵香
豆，广藿香，红没药，香草，香根草，麝香'，'前调：柠檬，雪松中调：粉红胡椒，凤梨兰，茉莉后调：柚木，瓷尾花，琥珀，广藿香，香根草，麝香'，'前调：酢，橙花油，
依兰，香柠檬，柠檬中调：瓷尾花，瓷尾根，茉莉，铃兰，玫瑰后调：琥珀，檀香木，广藿香，麝香，麝猫香，香草，橹木苔，香根草'，'前调：葡萄柚，木梨中调：茉莉，风
信子后调：麝香，瓷尾花，雪松，琥珀'，'前调：橙花油，香柠檬，绿叶中调：瓷尾花，水仙花，玫瑰，铃兰，依兰后调：香根草，檀香木，橹木苔，皮革'，'前调：蒲萝，
瓷尾花，广藿香，粉红胡椒，风信子中调：茉莉，柠檬后调：麝香，广藿香，香草，香根草'，'前调：柠檬，橘子，橙子，橙花油，醛，香柠檬，青柠中调：玫瑰，茉莉，依兰
后调：雪松，麝香，香草，瓷尾根'，'前调：橙花油，依兰，香柠檬，柠檬，醛中调：瓷尾花，茉莉，瓷尾根，玫瑰，铃兰后调：香根草，麝香，檀香木，广藿香，橹木苔，琥
珀，香草，麝猫香'，'前调：罗勒，橙子，香柠檬，柠檬，青柠，苦橙叶中调：茉莉，玫瑰，绿叶，希蒂氢后调：广藿香']

In [18]: Top = []
         for i in Main:
             i = i.strip("前调：")
             i = i.split("中调")
             top = i[0]
             top = top.replace(",","/n")
             Top.append(top)

In [19]: print(Top[:10])

['橘子/n葡萄柚/n黑加仑'，'橙子/n橘子/n橙花/n香柠檬'，'柠檬/n雪松'，'醛/n橙花油/n依兰/n香柠檬/n柠檬'，'葡萄柚/n木梨'，'橙花油/n香柠檬/n绿叶'，
'蒲萝/n瓷尾花/n广藿香/n粉红胡椒/n风信子'，'柠檬/n橘子/n橙子/n橙花油/n醛/n香柠檬/n青柠'，'橙花油/n依兰/n香柠檬/n柠檬/n醛'，'罗勒/n橙子/n香柠
檬/n柠檬/n青柠/n苦橙叶']
```

**Some variables.**

*That is what we have got.*

| | Names_perfume | Whole_info | Score |
|---|---|---|---|
| 1 | 皇家香氛 Parfums Regence Eau De Regence | 前调：中调：后调： | 0 人评价 |
| 2 | 皇家香氛 Parfums Regence Lavandou | 气味：薰衣草 | 0 人评价 |
| 3 | 皇家香氛 Parfums Regence Lavendou | 前调：中调：后调： | 0 人评价 |
| 4 | 皇家香氛 Parfums Regence Vert Regence | 前调：中调：后调： | 0 人评价 |
| 5 | 皇家香氛 Parfums Regence Santal | 前调：中调：后调： | 0 人评价 |
| 6 | 皇家香氛 Parfums Regence Oakmoss | 气味：香柠檬, 薰衣草, 广藿香, 白松香, 劳丹脂, 橡木苔 | 0 人评价 |
| 7 | 皇家香氛 Parfums Regence Kolnisch Juchten | 前调：中调：后调： | 0 人评价 |
| 8 | 皇家香氛 Parfums Regence Eau De Regence | 气味：柑橘, 果香 | 0 人评价 |
| 9 | 皇家香氛 Parfums Regence Santal | 气味：檀香木 | 0 人评价 |
| 10 | 皇家香氛 Parfums Regence Kolnisch Juchten | 气味：乳香, 闭鞘姜, 皮革, 麝香 | 0 人评价 |
| 11 | 皇家香氛 Parfums Regence Oakmoss | 前调：中调：后调： | 0 人评价 |
| 12 | 布鲁诺·法兹拉瑞 Bruno Fazzolari Jimmy, 2013 | 前调：柠檬, 依兰中调：玫瑰, 老鹳草后调：紫罗兰叶, 天芥菜, 檀香木 | 2 人评价 |
| 13 | 布鲁诺·法兹拉瑞 Bruno Fazzolari Lampblack, 2013 | 前调：胡椒, 橙子中调：葡萄柚后调：莎草, 安息香脂, 香根草 | 6 人评价 |
| 14 | 布鲁诺·法兹拉瑞 Bruno Fazzolari Room 237, 2015 | 气味：小雏菊, 乙烯基, 当归, 红没药, 闭鞘姜, 绿叶, 花香, 麝香 | 3 人评价 |
| 15 | 布鲁诺·法兹拉瑞 Au Dela – Narcisse des Montagnes Bruno Fazzolari Au Delà Narcisse, 2014 | 气味：香柠檬, 水仙花, 茉莉, 橡木苔, 橙花, 琥珀 | 2 人评价 |
| 16 | 布鲁诺·法兹拉瑞 Bruno Fazzolari Au Dela, 2013 | 前调：芫荽中调：橙花油, 茉莉后调：橙花, 琥珀, 树脂, 橡木苔 | 2 人评价 |
| 17 | 布鲁诺·法兹拉瑞 Bruno Fazzolari Seyrig, 2015 | 前调：醛, 橘子中调：依兰, 玫瑰, 铃兰, 丁香花后调：橡木苔, 麝香 | 2 人评价 |
| 18 | 布鲁诺·法兹拉瑞 Bruno Fazzolari Feu Secret, 2017 | 气味：鸢尾根, 云杉, 尤加利, 姜黄, 粉红胡椒, 雪松, 桦木 | 1 人评价 |
| 19 | 布鲁诺·法兹拉瑞 Bruno Fazzolari Monserrat, 2013 | 前调：葡萄柚, 胡萝卜籽中调：杏, 绿叶, 茉莉后调：龙涎香, 麝香, 石膏 | 1 人评价 |
| 20 | 布鲁诺·法兹拉瑞 Bruno Fazzolari Five, 2013 | 前调：柠檬, 橙子中调：迷迭香, 苦橙叶后调：木质香 | 1 人评价 |
| 21 | 布鲁诺·法兹拉瑞 Bruno Fazzolari Vetiverissimo, 2018 | 气味：辛香料, 香根草, 雪松, 树脂, 木质香 | 0 人评价 |
| 22 | 布鲁诺·法兹拉瑞 Bruno Fazzolari Vetiverissimo, 2018 | 气味：辛香料, 香根草, 雪松, 树脂, 木质香 | 0 人评价 |
| 23 | 布鲁诺·法兹拉瑞 Bruno Fazzolari Cadavre Exquis, 2016 | 气味：血橙, 橡胶, 依兰, 金盏花, 干果, 八角, 黑巧克力, 柏树, 安息香脂, 香草, 麝猫香 | 0 人评价 |
| 24 | 布鲁诺·法兹拉瑞 Bruno Fazzolari Fontevraud, 2018 | 气味：香柠檬, 番石榴, 梨, 玫瑰, 橡木苔, 红没药, 广藿香, 琥珀 | 0 人评价 |
| 25 | 布鲁诺·法兹拉瑞 Bruno Fazzolari Unsettled, 2017 | 气味：香柠檬, 茶叶, 快乐鼠尾草, 菠萝, 檀香木, 劳丹脂, 香草, 海水 | 0 人评价 |
| 26 | 伊露米琳 白色栀子花瓣 Illuminum White Gardenia Petals, 2011 | 前调：百合, 香柠檬, 醋栗叶中调：茉莉, 铃兰, 栀子花, 依兰后调：琥珀, 木质香 | 7.1 分 66 人评价 |
| 27 | 伊露米琳 Illuminum Wild Tobacco, 2011 | 前调：公丁香, 快乐鼠尾草中调：雪松, 烟草后调：劳丹脂, 海狸香, 零陵香豆 | 6.5 分 11 人评价 |
| 28 | 伊露米琳 Illuminum Tribal Black Tea, 2011 | 前调：香柠檬, 小豆蔻, 柠檬中调：茶叶, 花香, 茉莉后调：琥珀, 木质香, 肉豆蔻, 雪松 | 7 人评价 |
| 29 | 伊露米琳 Illuminum Moroccan Tuberose | 前调：老鹳草, 玫瑰中调：依兰, 玫瑰后调：晚香玉, 雪松, 麝香 | 2 人评价 |
| 30 | 伊露米琳 Illuminum Ginger Pear, 2011 | 前调：柑橘, 梨, 香柠檬中调：茶叶, 花香, 生姜, 大黄, 小苗香后调：麝香, 檀香木 | 8 人评价 |
| 31 | 伊露米琳 Illuminum Piper Leather, 2011 | 前调：芫荽, 胡椒中调：皮革, 茉莉, 胡萝卜籽后调：乳香, 麝猫香, 麝香 | 6 人评价 |
| 32 | 伊露米琳 Illuminum Rose Oud, 2011 | 前调：茉莉, 铃兰, 罗勒, 芫荽中调：玫瑰, 老鹳草后调：海狸香, 沉香(乌木), 广藿香 | 5 人评价 |
| 33 | 伊露米琳 Illuminum Taif Rose | 前调：老鹳草, 玫瑰中调：塔伊夫玫瑰, 玫瑰后调：麝香, 檀香木 | 5 人评价 |
| 34 | 伊露米琳 Illuminum Scarlet Oud, 2011 | 前调：胡椒, 蜂蜜中调：沉香(乌木), 香根草, 树脂后调：麝猫香, 海狸香 | 3 人评价 |
| 35 | 伊露米琳 Illuminum Wild Berry Blossom, 2011 | 前调：醋栗叶, 仙客来中调：牡丹, 野花, 树莓花, 木兰后调：雪松, 麝香 | 2 人评价 |
| 36 | 伊露米琳 Illuminum Black OUD | 前调：广藿香, 柑橘, 公丁香中调：沉香(乌木)后调：麝猫香 | 2 人评价 |
| 37 | 伊露米琳 Illuminum Saffron Amber, 2011 | 前调：香柠檬, 桃子, 芫荽中调：依兰, 藏红花, 茉莉后调：琥珀, 麝香 | 2 人评价 |
| 38 | 伊露米琳 Illuminum White on White | 前调：肉桂, 公丁香, 小豆蔻中调：广藿香后调：安息香, 香豆素, 橡木苔 | 2 人评价 |
| 39 | 伊露米琳 Illuminum Hindi OUD | 前调：桂花, 鸡蛋花, 栀子花中调：茉莉, 沉香(乌木), 黑檀红木后调：麝香 | 2 人评价 |
| 40 | 伊露米琳 Illuminum Skin Petals, 2011 | 前调：苹果花, 森林水果, 黑加仑中调：铃兰, 紫罗兰, 茉莉后调：琥珀, 麝香 | 1 人评价 |
| 41 | 伊露米琳 Illuminum White Saffron | 前调：老鹳草, 茉莉中调：藏红花, 玫瑰, 香根草后调：愈创木, 檀香木, 麝香 | 1 人评价 |
| 42 | 伊露米琳 Illuminum Cashmere Musk, 2011 | 前调：椰子, 绿叶, 依兰, 仙客来中调：风信子, 雪松, 檀香木后调：开司米木, 麝香, 木质香 | 1 人评价 |

**Original data.**

# 03 Data Cleaning


*In the process of data cleaning, we chose the variables from the whole information we get. For example, split the English name, Chinese name and year from the column of "Names_perfume", and split the three notes from the information of whole fragrances...*

# Step 1

**Extracted the variables we need from the raw data and eliminated some unwanted information.**

*# For missing values, we filled in with "None" or "0".*

```
import jieba
f1= open("cn_names.txt","a",encoding="utf-8",newline="\n")
for i in Chinese_names:
    i.encode("utf-8")
    seg_list = jieba.cut(i,cut_all=False)
    seg_list = ",".join(seg_list)
    f1.write(seg_list)
    f1.write("\n")
```

```
import re
Brand_names = []
Chinese_names = []
for i in Names_perfume:
    i = i.split(" ")
    cn = i[1]
    ncn = re.sub(r'[a-zA-Z]','',cn)
    ncn1 = re.sub(u"\\(.*?)|\\{.*?}|\\[.*?]|\\【.*?】", "",ncn)
    #i = re.sub(r'[^\u4e00-\u9fa5]','',ncn1)

    if ncn1 != '':
        Chinese_names.append(ncn1)
```

```
Years = []
for i in Names_perfume:
    if not "," in i:
        Years.append("None")
    else:
        i = i.split(",")
        i = i[-1]
        i = re.sub(r'(\s+)','',i)
        Years.append(i)
print(Years[100:120])
```

```
['2015', 'None', 'None', 'None', 'None', 'None', '2016', 'None', '2013', 'None', 'None', 'None', 'None', 'None', '2013', '2010', '2010', 'None', 'None', '2010']
```

```
print(Chinese_names[100:200])
```

```
['爱神周一晚报', '爱恋', '金色露雅', '斯嘉丽天堂', '安妮浓缩版', '捕捉我之水', '之水', '诺亚天堂', '斯嘉丽', '爱神之水', '露雅之水2018', '趣味火烈鸟之水', '露露', '闪耀爱神', '露雅', '安妮', '趣味火烈鸟之水', '露露', '爱神之水', '诺亚之水', '爱神丘比特之水火烈鸟', '优雅小姐天堂', '优雅小姐之水', '伊甸园', '诺亚之水火烈鸟', '爱神丘比特', '爱神丘比特', '红衣女郎', '拜占庭', '红衣女郎', '水之男', '拜占庭', '诱惑之水', '胡须', '他', '原版光辉', '同名', '点水成金', '谜', '蓝海珍珠', '光辉', '渴望女士香水', '秘密玫瑰极致', '环球', '鸢尾瀑布', '马加萨', '神秘沉香', '沉思之水男士香水', '夏日偷闲', '绝对', '花之水', '流行', '摇滚缪斯', '阳光', '幸运之星', '拜占庭之旅', '幻彩柑橘', '渴望男士香水', '秘密', '极度简红', '勇敢', '沉思之水', '玫瑰', '游艇', '浓情时光', '游艇', '时光', '鼠尾草与海盐', '蓝色风铃草', '英国梨与小苍兰', '檀花', '杏桃花与蜂蜜', '红玫瑰', '黑莓与月桂叶', '青柠罗勒与柑橘', '牡丹与胭红鹿绒', '伯爵茶与黄瓜', '白茉莉与薄荷', '黑石榴', '浓古龙-乌木与佛手柑', '浓古龙-丝绒玫瑰与乌木', '英国橡树与榛果', '浓古龙-黑琥珀与姜百合', '含羞草与小豆蔻', '柚子', '罗勒与橙花', '英国橡树与红醋栗', '琥珀与薰衣草', '浓古龙-晚香玉与天使草', '菩提花', '伦敦雨-黑雪松与杜松', '合欢花', '龙舌兰可可', '豆蔻辛姜', '浓古龙-末药与冬加豆', '樱花', '浓古龙-梵香与香樟', '星花木兰', '桂花', '夏之青梅']
```

# Step 1

**Extracted the variables we need from the raw data and eliminated some unwanted information.**

*# For missing values, we filled in with "None" or "0".*

```
Years = []
for i in Names_perfume:
    if not "," in i:
        Years.append("None")
    else:
        i = i.split(",")
        i = i[-1]
        i = re.sub(r'(\s+)','',i)
        Years.append(i)
print(Years[100:120])
```

```
['2015', 'None', 'None', 'None', 'None', 'None', '2016', 'None', '2013', 'None', 'None', 'None', '2010']
```

```
Score_number=[]
for i in Score:
    if "分" in i:
        i1 = i.split("分")
        i1 = i1[0]
        Score_number.append(i1)
    elif "分(" in i:
        i2 = i.split("分(")
        i2 = i[0]
        Score_number.append(i2)
    else:
        Score_number.append("None")
print(Score_number[100:120])
```

```
['None', ' 6.9 ', 'None', 'None', ' 7.1 ', 'None', 'None', 'None', 'None', 'None', 'None', ' 7.6 ', ' 6.9 ', 'None', 'None', 'None', 'None', 'None', 'None', 'None']
```

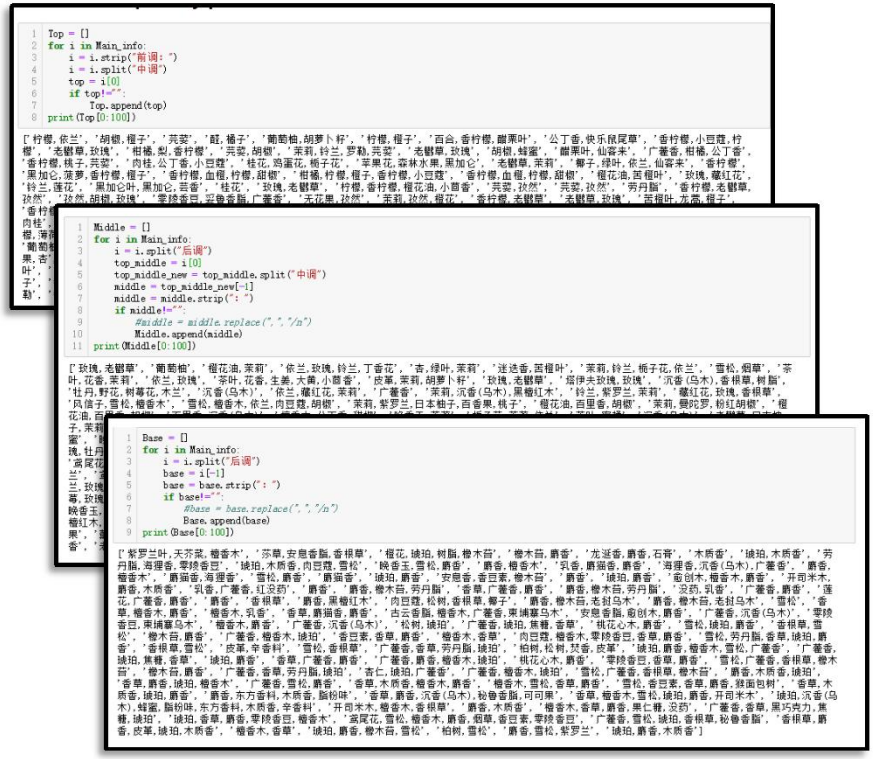
```
Reply_number = []
for i in Score:
    if "分" in i:
        i1 = i.strip("人评价")
        i1 = i1.split("分")
        i1 = i1[-1]
        Reply_number.append(i1)
    elif "分(" in i:
        i2 = i2.strip("评价)")
        i2 = i2.split("分(")
        i2 = i2[-1]
        Reply_number.append(i2)
    else:
        Reply_number.append("0")
print(Reply_number[100:120])
```

```
['0', ' 19 ', '0', '0', ' 17 ', '0', '0', '0', '0', '0', '0', ' 12 ', ' 11 ', '0', '0', '0', '0', '0', '0', '0']
```

# Step 1

**Extracted the variables we need from the raw data and eliminated some unwanted information.**

*# For missing values, we filled in with "None" or "0".*



```
Top = []
for i in Main_info:
    i = i.strip("前调：")
    i = i.split("中调")
    top = i[0]
    if top!="":
        Top.append(top)
print(Top[0:100])
```

```
['柠檬, 依兰', '胡椒, 橙子', '芫荽', '甜, 橘子', '葡萄柚, 胡萝卜籽', '柠檬, 橙子', '百合, 香柠檬, 酿栗叶', '公丁香, 快乐鼠尾草', '香柠檬, 小豆蔻, 柠檬', '老鹳草, 玫瑰', '柑橘, 梨, 香柠檬', '芫荽, 胡椒', '茉莉, 铃兰, 罗勒, 芫荽', '老鹳草, 玫瑰', '胡椒, 蜂蜜', '酿栗叶, 仙客来', '广藿香, 柑橘, 公丁香', '香柠檬, 桃子, 芫荽', '肉桂, 公丁香, 小豆蔻', '桂花, 鸡蛋花, 栀子花', '苹果花, 蓉林水果, 黑加仑', '老鹳草, 茉莉', '椰子, 绿叶, 依兰, 仙客来', '香柠檬', '黑加仑, 菠萝, 香柠檬, 橙子', '香柠檬, 血橙, 柠檬, 甜橘', '柑橘, 柠檬, 橙子, 香柠檬, 小豆蔻', '香柠檬, 血橙, 柠檬, 甜橘', '橙花油, 茧橙叶', '玫瑰, 檫红花', '铃兰, 蓬花', '黑加仑叶, 黑加仑, 苔香', '桂花', '玫瑰, 老鹳草', '柠檬, 香柠檬, 橙花油, 小茴香', '芫荽, 孜然', '芫荽, 孜然', '劳丹脂', '香柠檬, 老鹳草, 孜然', '孜然, 胡椒, 玫瑰', '零陵香豆, 妥鲁香脂, 广藿香', '无花果, 孜然', '茉莉, 孜然, 橙花', '香柠檬, 老鹳草', '老鹳草, 玫瑰', '茧橙叶, 龙蒿, 橙子',
```

```
Middle = []
for i in Main_info:
    i = i.split("后调")
    top_middle = i[0]
    top_middle_new = top_middle.split("中调")
    middle = top_middle_new[-1]
    middle = middle.strip(": ")
    if middle!="":
        #middle = middle.replace(",","/n")
        Middle.append(middle)
print(Middle[0:100])
```

```
['玫瑰, 老鹳草', '葡萄柚', '橙花油, 茉莉', '依兰, 玫瑰, 铃兰, 丁香花', '杏, 绿叶, 茉莉', '迷迭香, 茧橙叶', '茉莉, 铃兰, 栀子花, 依兰', '雪松, 烟草', '茶叶, 花香, 茉莉', '依兰, 玫瑰', '茶叶, 花香, 生姜, 大黄, 小茴香', '皮革, 茉莉, 胡萝卜籽', '玫瑰, 老鹳草', '熔伊夫玫瑰, 玫瑰', '沉香(乌木), 香根草, 树脂', '牡丹, 野花, 树莓花, 木兰', '沉香(乌木)', '依兰, 檫红花, 茉莉', '广藿香', '茉莉, 沉香(乌木), 黑檀红木', '铃兰, 紫罗兰, 茉莉', '檫红花, 玫瑰, 香根草', '风信子, 雪松, 檀香木', '雪松, 檀香木, 依兰, 向日葵, 胡椒', '茉莉, 紫罗兰, 日本柚子, 百香果, 桃子', '橙花油, 百里香, 胡椒', '茉莉, 曼陀罗, 粉红胡椒', '橙花油, 白
```

```
Base = []
for i in Main_info:
    i = i.split("后调")
    base = i[-1]
    base = base.strip(": ")
    if base!="":
        #base = base.replace(",","/n")
        Base.append(base)
print(Base[0:100])
```

```
['紫罗兰叶, 天芥菜, 檀香木', '苏草, 安息香脂, 香根草', '橙花, 琥珀, 树脂, 橙木苔', '檀木苔, 麝香', '龙涎香, 麝香, 石蒜', '木质香', '琥珀, 木质香', '劳丹脂, 海狸香, 零陵香豆', '琥珀, 木质香, 向日葵, 雪松', '琥珀, 木质香, 晚香玉, 雪松, 麝香', '麝香, 檀香木', '乳香, 麝猫香, 麝香', '海狸香, 沉香(乌木), 广藿香', '檀香木', '麝猫香, 海狸香', '雪松, 麝香', '麝猫香', '琥珀, 麝香', '安息香, 香豆素, 檀木苔', '麝香', '琥珀, 麝香', '愈创木, 檀香木, 麝香', '开司米木, 麝香, 木质香', '乳香, 广藿香, 红没药', '麝香', '麝香, 檀木苔, 劳丹脂', '香草, 广藿香, 麝香', '麝香, 檀木苔, 劳丹脂', '没药, 乳香', '广藿香, 麝香', '莲花, 广藿香, 麝香', '麝香', '香根草', '麝香, 黑檀红木', '向日葵, 松树, 香根草, 柳子', '麝香, 檀木苔, 老挝乌木', '麝香, 檀木苔, 老挝乌木', '雪松', '香草, 檀香木, 麝香', '檀香木, 乳香', '香草, 麝猫香, 麝香', '古云香脂, 檀香木, 广藿香, 果塘寨乌木', '安息香脂, 愈创木, 麝香', '广藿香, 沉香(乌木)', '零陵香豆, 果塘寨乌木', '檀香木, 麝香', '广藿香, 沉香(乌木)', '松树, 琥珀', '广藿香, 琥珀, 焦糖, 香草', '桃花心木, 麝香', '雪松, 琥珀, 麝香', '香根草, 雪松', '檀木苔, 麝香', '广藿香, 檀香木, 琥珀', '香豆素, 香草, 麝香', '檀香木, 香草', '向日葵, 檀香木, 零陵香豆, 香草, 麝香', '雪松, 劳丹脂, 香草, 琥珀, 麝香', '香根草, 雪松', '皮革, 辛香料', '雪松, 香根草', '广藿香, 香草, 劳丹脂, 琥珀', '柏树, 松树, 焚香, 皮革', '琥珀, 麝香, 檀香木, 雪松, 广藿香', '琥珀, 焦糖, 香草', '琥珀, 麝香', '香草, 广藿香, 麝香', '广藿香, 麝香, 檀香木, 琥珀', '桃花心木, 麝香', '零陵香豆, 香草, 麝香', '雪松, 广藿香, 香根草, 檀木苔', '檀木苔, 麝香', '广藿香, 香草, 劳丹脂, 琥珀', '杏仁, 琥珀, 广藿香', '广藿香, 檀香木, 琥珀', '雪松, 广藿香, 香根草, 檀木苔', '麝香, 木质香, 琥珀', '香草, 麝香, 琥珀, 檀香木', '广藿香, 雪松, 麝香', '香草, 木质香, 檀香木, 麝香', '檀香木, 雪松, 香草, 麝香', '雪松, 香豆素, 香草, 麝香, 猩面包树', '香草, 木质香, 琥珀, 麝香', '麝香, 东方香料, 木质香, 脂粉味', '香草, 麝香, 沉香(乌木), 秘鲁香脂, 可可果', '香草, 檀香木, 雪松, 琥珀, 麝香, 开司米木', '琥珀, 沉香(乌木), 蜂蜜, 脂粉味, 东方香料, 木质香, 辛香料', '开司米木, 檀香木, 香根草', '麝香, 木质香', '檀香木, 香草, 麝香, 黑仁鳍, 没药', '广藿香, 香草, 黑巧克力', '黑檀, 琥珀', '琥珀, 香草, 麝香, 零陵香豆, 檀香木', '葛尾花, 雪松, 檀香木, 麝香, 烟草, 香豆素, 零陵香豆', '广藿香, 雪松, 琥珀, 香根草, 秘鲁香脂', '香根草, 麝香, 皮革, 琥珀, 木质香', '檀香木, 香草', '琥珀, 麝香, 檀木苔, 雪松', '柏树, 雪松', '麝香, 雪松, 紫罗兰', '琥珀, 麝香, 木质香']
```

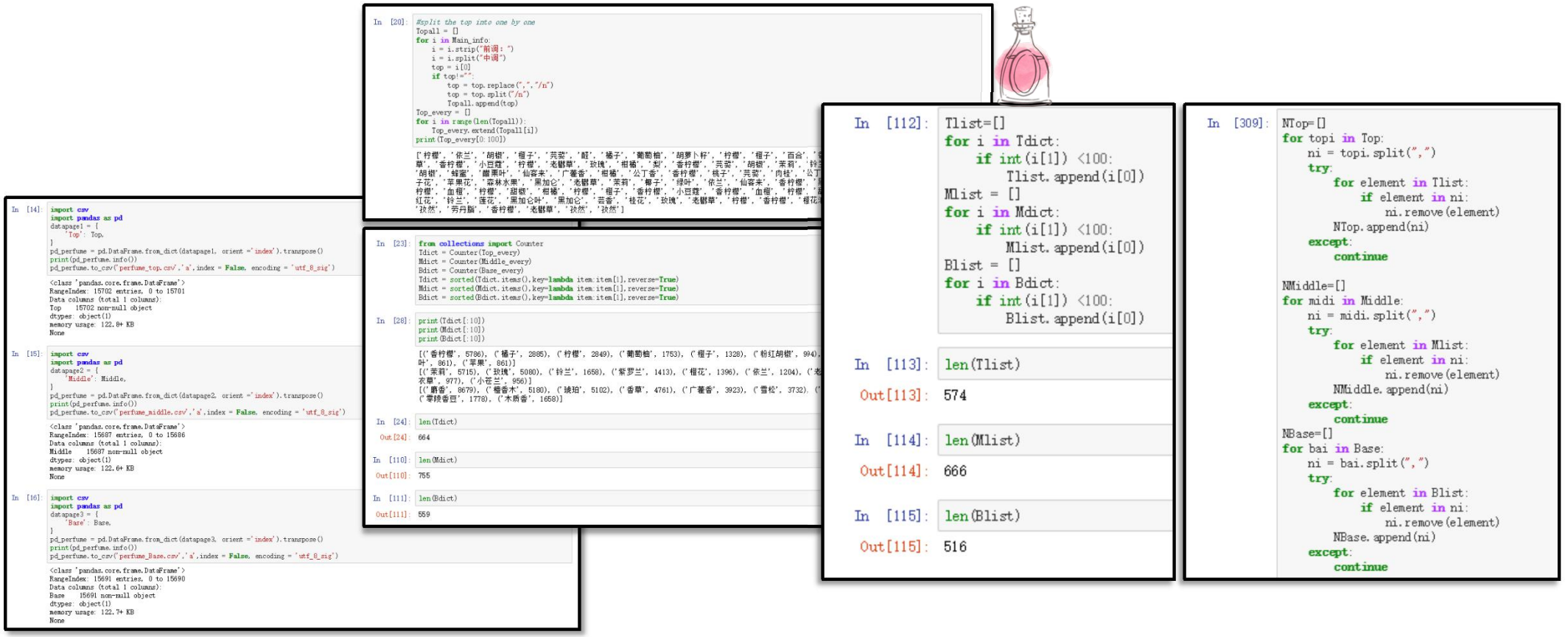
```
Alone_info = []
Main_info = []
for i in Whole_info:
    if re.findall(r"^气味",i):
        alone = i.replace("气味：","")
        #alone = alone.replace(",","/n")
        Alone_info.append(alone)
    else:
        Main_info.append(i)
print(Alone_info[100:120])
print(Main_info[100:110])
```

```
['顿加豆, 干草, 可可果, 蜂蜜, 烟草', '威士忌, 玫瑰, 甜椒, 雪松, 蜂蜡', '小麦, 绿叶, 葡萄柚, 绣线菊', '橘子, 鼠尾草, 蜂蜡, 烟草', '报春花, 玉米穗, 含羞草, 香草, 847', '睡莲, 绿叶, 依兰, 麝香, 香草', '劳丹脂, 愈创木', '杏仁, 红醋栗, 檀香木', '风信子, 老鹳草, 香根草', 'palo santo wood, siam wood, white champa flower', 'black pepper, 康乃馨, 肉桂, 老鹳草', '青草, 茉莉, 橙子, wood', '橙子, sampaguita, ylang ylang', '茉莉, 柠檬, 玫瑰, 檀香木', 'citrus, 玫瑰, cedar wood, fire tree', '草桂花, 皮革, rosewood', '苦艾, balsam, oud', '香柠檬, 茉莉, 广藿香, 香草', '小豆蔻, indonesian vetiver, woods', '香柠檬, 茉莉, 广藿香, 香草']
['前调：鼠尾草, 柏树 中调：红苹果, 纸莎草, 雪花莲, 木质香后调：香根草, 麝香, 皮革, 琥珀, 木质香', '前调：栀子花, 晚香玉, 无花果叶, 紫丁香, 紫罗兰, 大麻, 鸦片, 罗勒 中调：老鹳草, 猕猴桃, 菠萝, 葡萄柚, 公丁香, 晚香玉后调：檀香木, 香草', '前调：香瓜, 桃子, 柠檬 中调：菠萝, 黑加仑 后调：琥珀, 麝香, 橡木苔, 雪松', '前调：柚子, 粉红胡椒 中调：花香, 果仁糖, 树莓 后调：柏树, 雪松', '前调：柠檬, 黑加仑, 红醋栗 中调：玫瑰, 树莓, 石榴 后调：麝香, 雪松, 紫罗兰', '前调：荔枝, 日本柚子, 柠檬 中调：木兰, 小苍兰, 玫瑰 后调：琥珀, 麝香, 木质香', '前调：橙子, 桃子, 梨 中调：茉莉, 莲花, 忍冬 后调：雪松, 琥珀, 香草', '前调：橘子, 粉红胡椒, 芒果 中调：竹子, 玫瑰, 牡丹 后调：香根草, 檀香木, 香草', '前调：火龙果, 玫瑰, 梨, 红浆果 中调：牡丹, 小苍兰, 木兰 后调：麝香, 檀香木, 木质香, 香草, 树莓', '前调：橙子, 杏, 绿叶, 树莓, 香柠檬 中调：花香, 鸢尾花, 茉莉, 鸢尾根, 铃兰, 玫瑰 后调：零陵香豆, 檀香木, 琥珀, 雪松']
```

# Step 2

## Selected 100 “core nodes” by frequency counting.

*# To simplify the networks and easily find underlying patterns from the most important nodes.*



```
In [20]:
#split the top into one by one
Topall = []
for i in Main_info:
    i = i.strip("前调：")
    i = i.split("中调")
    top = i[0]
    if top!="":
        top = top.replace("，","/n")
        top = top.split("/n")
        Topall.append(top)
Top_every = []
for i in range(len(Topall)):
    Top_every.extend(Topall[i])
print(Top_every[0:100])
```

```
['柠檬', '依兰', '胡椒', '橙子', '芫荽', '醋', '橘子', '葡萄柚', '胡萝卜籽', '柠檬', '橙子', '百合', '
草', '香柠檬', '小豆蔻', '柠檬', '老鹳草', '玫瑰', '柑橘', '梨', '香柠檬', '芫荽', '胡椒', '茉莉', '铃
'胡椒', '蜡蜜', '雨果叶', '仙客来', '广藿香', '柑橘', '公丁香', '香柠檬', '桃子', '芫荽', '肉桂', '公丁
子花', '苹果花', '森林水果', '黑加仑', '老鹳草', '茉莉', '莓子', '绿叶', '依兰', '仙客来', '香柠檬', '
柠檬', '血橙', '柠檬', '甜橙', '柑橘', '柠檬', '橙子', '香柠檬', '小豆蔻', '香柠檬', '血橙', '柠檬', '
红花', '铃兰', '莲花', '黑加仑叶', '黑加仑', '芸香', '桂花', '玫瑰', '老鹳草', '柠檬', '香柠檬', '橙花
'孜然', '劳丹脂', '香柠檬', '老鹳草', '孜然', '孜然']
```

```
In [23]:
from collections import Counter
Tdict = Counter(Top_every)
Mdict = Counter(Middle_every)
Bdict = Counter(Base_every)
Tdict = sorted(Tdict.items(),key=lambda item:item[1],reverse=True)
Mdict = sorted(Mdict.items(),key=lambda item:item[1],reverse=True)
Bdict = sorted(Bdict.items(),key=lambda item:item[1],reverse=True)

In [28]:
print(Tdict[:10])
print(Mdict[:10])
print(Bdict[:10])

[('香柠檬', 5786), ('橘子', 2885), ('柠檬', 2849), ('葡萄柚', 1753), ('橙子', 1328), ('粉红胡椒', 994),
叶', 861), ('苹果', 861)]
[('茉莉', 5715), ('玫瑰', 5080), ('铃兰', 1658), ('紫罗兰', 1413), ('橙花', 1396), ('依兰', 1204), ('老
衣草', 977), ('小花兰', 956)]
[('雨香', 8679), ('檀香木', 5180), ('琥珀', 5102), ('香草', 4761), ('广藿香', 3923), ('雪松', 3732),
('零陵香豆', 1778), ('木质香', 1658)]

In [24]: len(Tdict)
Out[24]: 664

In [110]: len(Mdict)
Out[110]: 755

In [111]: len(Bdict)
Out[111]: 559
```

```
In [14]:
import csv
import pandas as pd
datapage1 = {
    'Top': Top,
}
pd_perfume = pd.DataFrame.from_dict(datapage1, orient='index').transpose()
print(pd_perfume.info())
pd_perfume.to_csv('perfume_top.csv','a',index = False, encoding = 'utf_8_sig')

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 15702 entries, 0 to 15701
Data columns (total 1 columns):
Top    15702 non-null object
dtypes: object(1)
memory usage: 122.8+ KB
None

In [15]:
import csv
import pandas as pd
datapage2 = {
    'Middle': Middle,
}
pd_perfume = pd.DataFrame.from_dict(datapage2, orient='index').transpose()
print(pd_perfume.info())
pd_perfume.to_csv('perfume_middle.csv','a',index = False, encoding = 'utf_8_sig')

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 15687 entries, 0 to 15686
Data columns (total 1 columns):
Middle    15687 non-null object
dtypes: object(1)
memory usage: 122.6+ KB
None

In [16]:
import csv
import pandas as pd
datapage3 = {
    'Base': Base,
}
pd_perfume = pd.DataFrame.from_dict(datapage3, orient='index').transpose()
print(pd_perfume.info())
pd_perfume.to_csv('perfume_Base.csv','a',index = False, encoding = 'utf_8_sig')

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 15691 entries, 0 to 15690
Data columns (total 1 columns):
Base    15691 non-null object
dtypes: object(1)
memory usage: 122.7+ KB
None
```

```
In [112]:
Tlist=[]
for i in Tdict:
    if int(i[1]) <100:
        Tlist.append(i[0])
Mlist = []
for i in Mdict:
    if int(i[1]) <100:
        Mlist.append(i[0])
Blist = []
for i in Bdict:
    if int(i[1]) <100:
        Blist.append(i[0])

In [113]: len(Tlist)
Out[113]: 574

In [114]: len(Mlist)
Out[114]: 666

In [115]: len(Blist)
Out[115]: 516
```

```
In [309]:
NTop=[]
for topi in Top:
    ni = topi.split(",")
    try:
        for element in Tlist:
            if element in ni:
                ni.remove(element)
        NTop.append(ni)
    except:
        continue

NMiddle=[]
for midi in Middle:
    ni = midi.split(",")
    try:
        for element in Mlist:
            if element in ni:
                ni.remove(element)
        NMiddle.append(ni)
    except:
        continue
NBase=[]
for bai in Base:
    ni = bai.split(",")
    try:
        for element in Blist:
            if element in ni:
                ni.remove(element)
        NBase.append(ni)
    except:
        continue
```

# Step 3

## Extracted the variables.

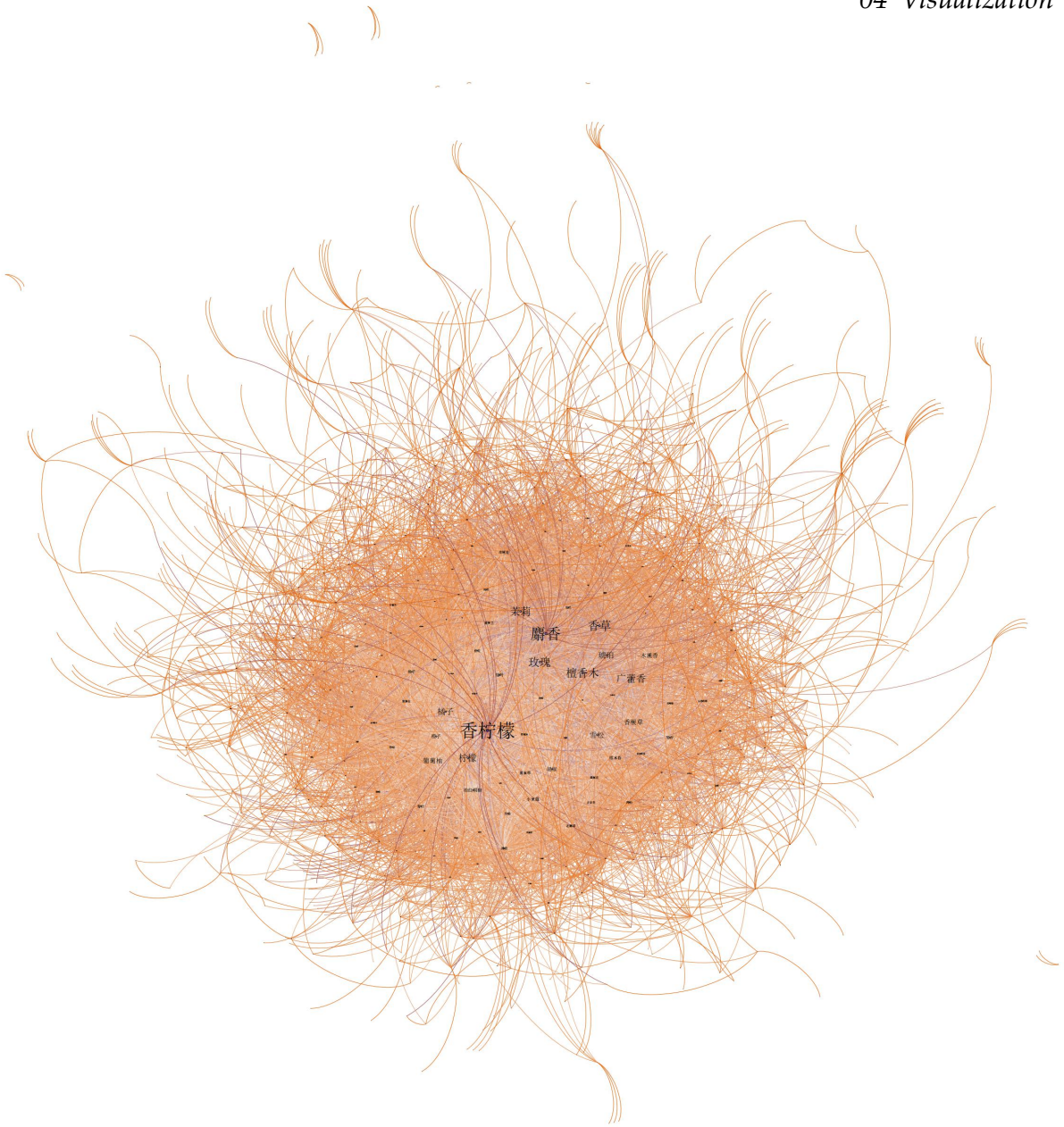
*By using the same method as above, we extracted the variables according to the year, score, and number of evaluations.*

*Among them, the year is divided into three periods: before 1900, 1900-2010, and 2019. The score is 9.0 and above. The reply number is more than 1,000.*

# 04 Visualization


*After data collection and analysis, we generated three clusters of the ingredients used in top notes, middle notes and base notes of perfumes. Then we drew several networks based on different variables.*

**We got 700+ nodes.**



# Notes Commonly Used TOP100

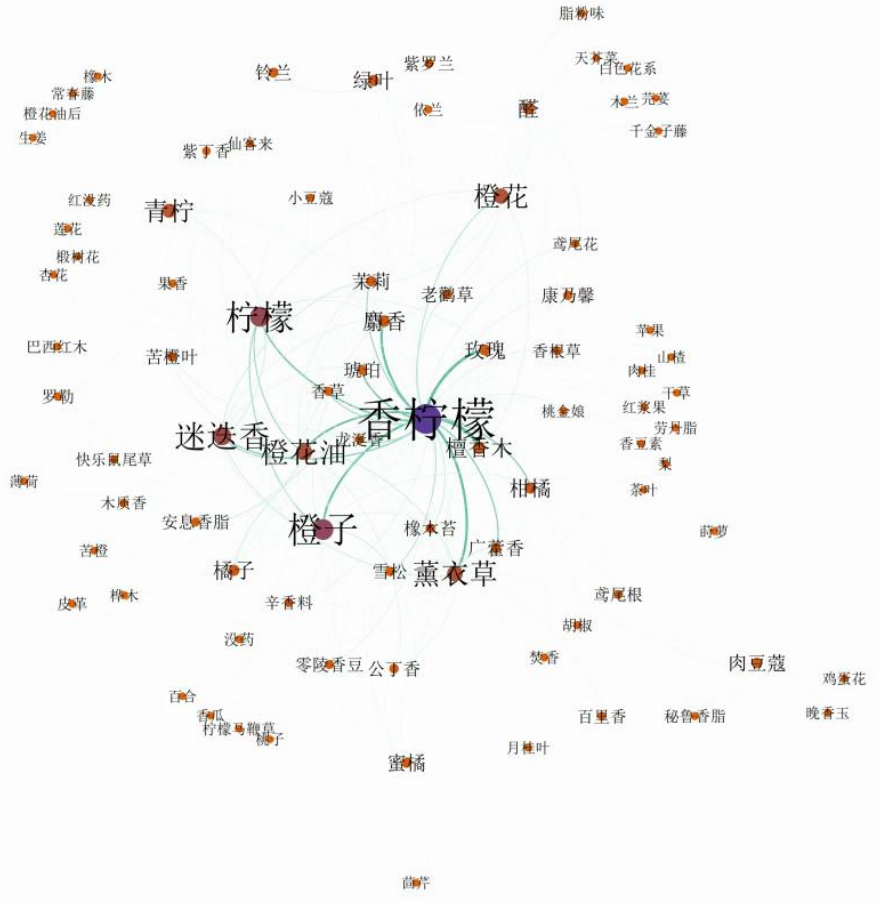
# Notes Used Before 1900

## All Notes Before 1900

**87** Nodes
**220** Edges

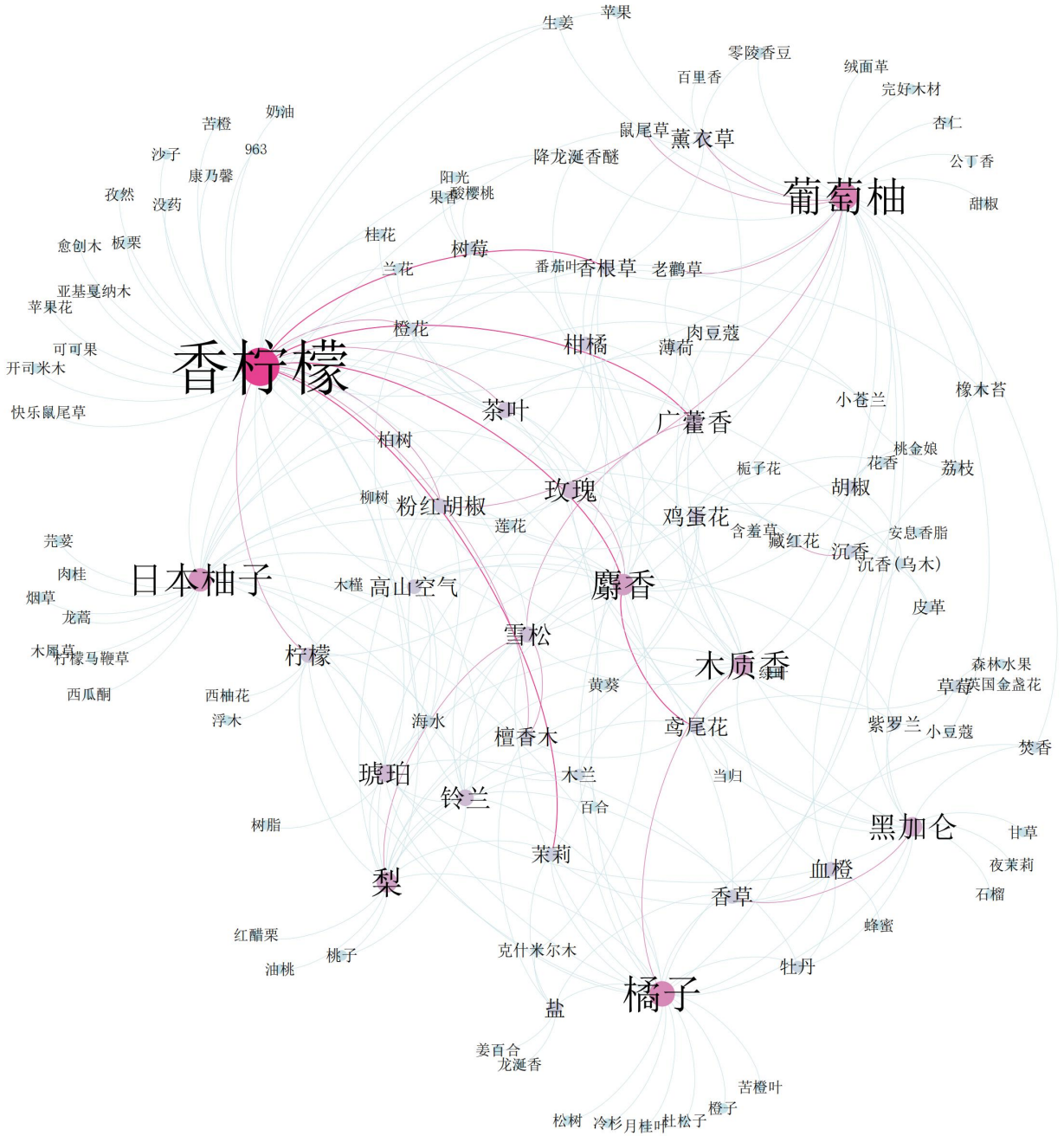
**The Biggest Nodes**
香柠檬 柠檬 橙子

**The Most Weighted Edges**
香柠檬-橙子
香柠檬-檀香木
香柠檬-玫瑰



# Notes Used After 1900

# Notes Used in 2019




**119 Nodes**
**253 Edges**

## The Biggest Nodes

香柠檬 葡萄柚

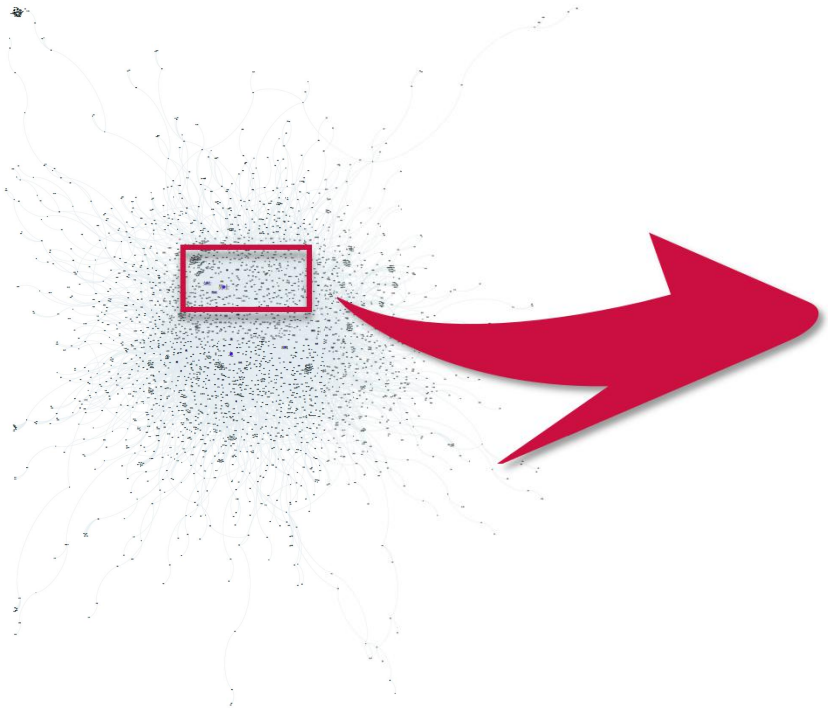
## The Most Weighted Edges

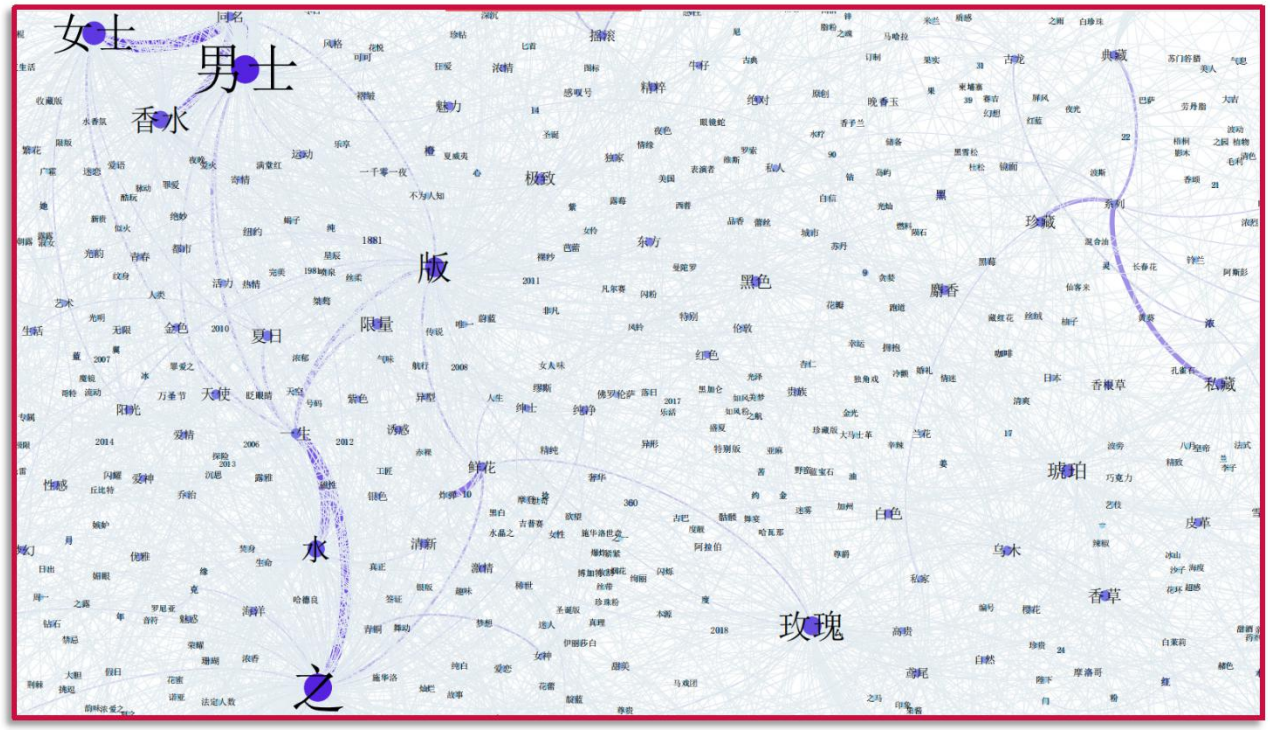
香柠檬-葡萄柚
香柠檬-香根草
麝香-鸢尾花

# Notes Most Commented

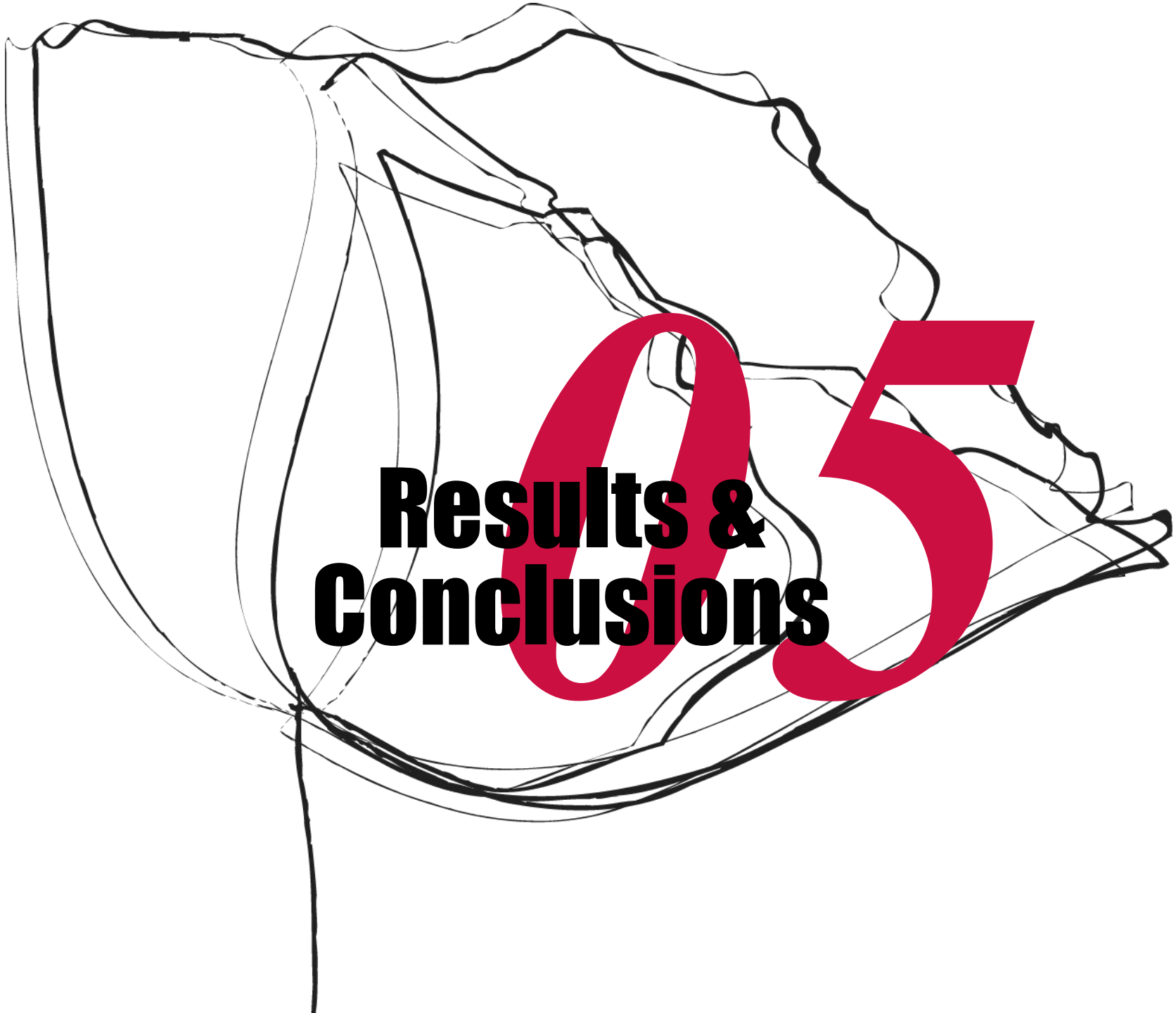
# Notes Scored above 9.0

# Brand Names in Chinese






**3586 Nodes**
**6782 Edges**



# 05 Results & Conclusions

*Although we got over 30 pictures about connected networks of perfumes, we still wonder that how and why any two nodes could be connected.*

*Another interesting thing is, some ingredients have been commonly used from ancient times to the present. Why are they so popular?*

*To answer these questions, we decided to* ***introduce a formula*** *to evaluate the performances of different ingredients.*

```
In [70]:
import csv
Escore = {}
wdict = {}
with open('all_score.csv', 'r',encoding="utf-8") as f:
    for line in f:
        line = line.strip('"')
        line = line.strip("\ufeff")
        line = line.strip("\n")
        line = line.split(",")
        newline = []

        for i in line:
            if i!="":
                newline.append(i)
        wordcount = len(newline)-1
        everyscore = float(line[-1])/int(wordcount)

        for i in newline[:-2]:
            i = i.strip('"')
            if i in wdict:
                count = wdict[i]
            else:
                count=0
            count = count + 1
            wdict[i]=count

            if i in Escore.keys():
                wholescore = float(Escore[i]) + everyscore
                Escore[i] = wholescore
            else:
                Escore[i] = everyscore
```

```
In [66]:
for k,v in Escore.items():
    if wdict.get(k):
        Escore[k]=round(float(Escore[k])/wdict[k],2)
print(Escore)
```

```
{'草莓': 0.91, '奶黄': 1.71, '橙子': 0.81, '苦橙': 0.93, '橘子': 0.81, '橙花': 0.82, '茉莉': 0.8, '桂花': 0.86, '牡丹': 0.89, '报春花': 1.34, '玫瑰': 0.83, '麝香': 0.79, '含羞草': 0.86, '康乃馨': 0.53, '晚香玉': 0.79, '醋栗叶': 0.7, '兰花': 0.73, '小苍兰': 0.82, '紫丁香': 0.79, '鸡蛋花': 0.88, '提亚蔷花': 0.86, '辛香料': 0.87, '琥珀': 0.77, '香草': 0.83, '快乐鼠尾草': 0.74, '焚香': 0.79, '蜂蜜': 0.79, '木质香': 1.0, '红没药': 0.66, '榄香脂': 0.77, '劳丹脂': 0.72, '零陵香豆': 0.73, '安息香脂': 0.73, '黑巧克力': 1.0, '鸢尾花': 0.82, '芫荽': 0.65, '龙涎香': 0.82, '檀香木': 0.74, '广藿香': 0.78, '橡木苔': 0.64, '848': 2.4, '矢车菊': 2.0, '榛子': 1.06, '阿米香树花': 0.96, '鸢尾根': 0.61, '椰子': 0.8, '黄葵': 0.93, '阿魏': 0.51, '香柠檬': 0.75, '巴西红木': 0.6, '艾蒿': 0.66, '薄荷': 0.87, '绿叶': 0.85, '桃子': 0.68, '依兰': 0.67, '雪松': 0.79, '皮革': 0.77, '马黛茶': 0.94, '藏红花': 0.84, '肉桂': 0.7, '粉红胡椒': 0.87, '朗姆酒': 0.85, '莎草': 0.74, '沉香(乌木)': 0.93, '杜松子': 0.78, '紫罗兰叶': 0.81, '橙花油': 0.87, '铃兰': 0.71, '月桂叶': 0.67, '甜椒': 0.82, '没药': 0.79, '愈创木': 0.78, '葛缕子': 0.64, '白松香': 0.74, '罗勒': 0.7, '肉豆蔻': 0.7, '香根草': 0.78, '松树': 0.68, '老鹳草': 0.7, '安息香': 0.68, '葡萄柚': 0.85, '柠檬': 0.77, '仙客来': 0.68, '洋甘菊': 0.83, '胡椒': 0.8, '冷杉': 0.68, '黑加仑': 0.79, '杏仁': 0.9, '木兰': 0.81, '杜松': 0.77, '纸莎草': 0.78, '牛奶': 1.22, '百里香': 0.79, '薰衣草': 0.77, '咖啡': 0.88, '苹果': 0.81, '鼠尾草': 0.72, '梨': 0.89, '小豆蔻': 0.74, '生姜': 0.85, '苦橙叶': 0.88, '桦木': 0.75, '果香': 0.92, '树脂': 0.81, '焦糖': 0.91, '孜然': 0.74, '柏树': 0.77, '桃金娘': 0.78, '迷迭香': 0.71, '龙蒿': 0.57, '竹子': 0.8, '红浆果': 0.88, '八角': 0.85, '开司米木': 0.82, '柚子': 0.82, '树莓': 0.77, '可可果': 0.95, '百合': 0.86, '栀子花': 0.81, '巴西坚果': 3.13, '芭蕉叶': 0.61, '香蕉': 0.71, '白色花系': 1.19, '红醋栗': 0.97, '天芥菜': 0.75, '柠檬花': 0.84, '柑橘': 1.07, '紫罗兰': 0.78, '水仙花': 0.7, '忍冬': 0.82, '葫芦巴': 0.78, '闭鞘姜': 0.39, '乳香': 0.76, '风信子': 0.68, '金雀花': 0.98, '青草': 0.97, '木犀草': 0.72, '茴芹': 0.71, '干邑白兰地': 0.91, '醛': 0.61, '莲花': 0.87, '橄榄花': 0.79, '克里曼丁红橘': 0.83, '蜜橘': 0.78, '绒面革': 0.76, '干草': 1.05, '山楂': 0.75, '公丁香': 0.65, '无花果叶': 0.89, '蓝莓': 0.85, '黑莓': 0.84, '金属': 1.06, '剑兰': 1.33, '菊花': 0.91, '桃花心木': 0.74, '棉花': 1.18, '杏': 0.76, '香瓜': 0.71, '睡莲': 0.85, '香豌豆': 1.0, '洋槐': 0.73, '百香果': 0.88, '番石榴': 0.9, '芒果': 0.84, '西瓜': 0.85, '英国金盏花': 0.58, '野玫瑰果': 0.81, '西番莲': 0.93, '荔枝': 0.85, '红苹果': 0.95, '樱花': 1.0, '猕猴桃': 0.81,
```

$$\textbf{Score} = \frac{\sum (\textit{perfume's score / the number of ingredents it contains})}{\textit{the frequency of node of whole perfumes}}$$

| | | |
|---|---|---|
| 1 | 巴西坚果 | 3.13 |
| 2 | 柳树 | 2.87 |
| 3 | 848 | 2.4 |
| 4 | 印蒿 | 2.33 |
| 5 | 矢车菊 | 2 |
| 6 | 苦艾酒 | 1.9 |
| 7 | 玉米穗 | 1.82 |
| 8 | 泥炭 | 1.76 |
| 9 | 顿加豆 | 1.74 |
| 10 | 奶黄 | 1.71 |
| 11 | 烈酒 | 1.7 |
| 12 | 麦芽 | 1.7 |
| 13 | 红色水果 | 1.68 |
| 14 | 云莓 | 1.68 |
| 15 | 乌木 | 1.66 |
| 16 | 番荔枝 | 1.64 |
| 17 | 面包 | 1.6 |
| 18 | 榛子可可酱 | 1.58 |
| 19 | 蛋白糖饼 | 1.51 |
| 20 | 鸦片 | 1.51 |
| 21 | 鱼子酱 | 1.47 |
| 22 | 树莓花 | 1.43 |
| 23 | 法式焦糖布丁 | 1.41 |
| 24 | 厚壳桂 | 1.39 |
| 25 | 油桃花 | 1.39 |
| 26 | 沙漠玫瑰 | 1.38 |
| 27 | 杜松子酒 | 1.37 |
| 28 | 牛肝菌 | 1.35 |
| 29 | 小麦 | 1.35 |
| 30 | 报春花 | 1.34 |

# High Socre But Less Used

*Here are some possible reasons:*

a. *Raw materials are* ***expensive****.*

b. ***Complex*** *production process. (e.g., flavor components can easily volatilize )*

c. *Some materials have strong smells, which leads to that they can* ***hardly find their "good partners"****.*

| | | |
|---|---|---|
| 370 | 橘子花 | 0.75 |
| 371 | 马天尼 | 0.75 |
| 372 | 蔓越莓 | 0.75 |
| 373 | 蔓长春花 | 0.75 |
| 374 | 木瓜 | 0.75 |
| 375 | 山楂 | 0.75 |
| 376 | 天芥菜 | 0.75 |
| 377 | 铁杉花 | 0.75 |
| 378 | 香柠檬 | 0.75 |
| 379 | 白松香 | 0.74 |
| 380 | 核桃 | 0.74 |
| 381 | 苦艾 | 0.74 |
| 382 | 快乐鼠尾草 | 0.74 |
| 383 | 李子 | 0.74 |
| 384 | 莎草 | 0.74 |
| 385 | 檀香木 | 0.74 |
| 386 | 桃花心木 | 0.74 |
| 387 | 小豆蔻 | 0.74 |
| 388 | 杏花 | 0.74 |
| 389 | 吲哚 | 0.74 |
| 390 | 柚木 | 0.74 |
| 391 | 芸香 | 0.74 |
| 392 | 孜然 | 0.74 |
| 393 | 安息香脂 | 0.73 |
| 394 | 菠萝 | 0.73 |
| 395 | 兰花 | 0.73 |
| 396 | 零陵香豆 | 0.73 |
| 397 | 玫瑰草 | 0.73 |
| 398 | 牛膝草 | 0.73 |

# Commonly Used But Relatively Low Score

*It might be due to:*

a. *Most of them are* ***common materials***.

b. *Light smell, usually serve as* ***a contrast or foil***. **“甘作绿叶当陪衬”**

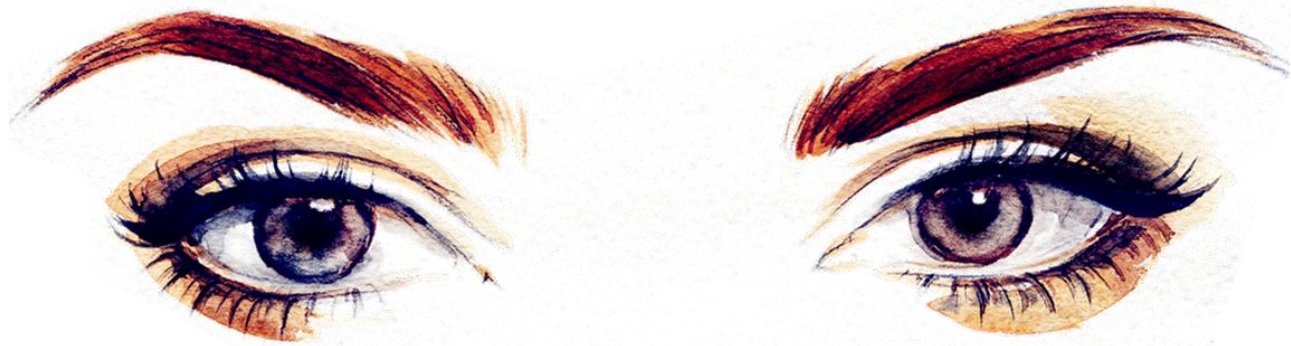

# Limitations

*a. Did not make full use of all information we scraped from Nosetime. (Why we failed to create more complicated networks)*

*b. The connections between brands could not be showed. (Time limitation)*

*c. The features of nodes was not represented well.*

*d. Most of our visualized outputs were static images.*



# Thanks for your attention