In [ ]:

```
수집 데이터(국제, 생활 등) => 토픽모델링
네이버-영화 -> 영화평 수집(*)
```

In [1]:

```python
from os import listdir

def fileids(path, ext='txt'):
    return [path+file for file in listdir(path) if file.split(".")[-1] == ext]

def filecontent(file):
    with open(file, encoding="utf-8") as fp:
        content = fp.read()
    return content
```

In [2]:

```python
def ngram(term, n=2):
    return [term[i:i+n] for i in range(len(term) - n + 1)]
```

In [3]:

```python
import re
from string import punctuation
#corpus = filecontent(fileids('F:/daumnews (2)/daumnews')[-2])
#print(corpus)
#print(len(corpus))
pattern = dict()
#구두점
pattern1 = re.compile(r"[{0}]".format(re.escape(punctuation)))
#corpus = pattern1.sub(" ", corpus)
pattern["punc"] = pattern1
# print(len(corpus))
# corpus
#불용어
pattern2 = re.compile(r"[A-Za-z0-9]{7,}")
#corpus = pattern2.sub(" ", corpus)
pattern["stop"] = pattern2
#print(len(corpus))
#이메일
pattern3 = re.compile(r"₩w{2,}@₩w{3,}.(.₩w{2,})+")
pattern3 = re.compile(r"₩w{2,}@(.?₩w{2,})+")
#corpus = pattern3.sub("", corpus)
pattern["email"] = pattern3
#도메인
pattern4 = re.compile(r"(.?₩w{2,}){2,}")
#corpus = pattern4.sub("", corpus)
pattern["domain"] = pattern4
#한글 이외
pattern5 = re.compile(r"[^가-힣0-9]+")
#corpus = pattern5.sub(" ", corpus)
pattern["nonkorean"] = pattern5
#반복되는 공백문자
pattern6 = re.compile(r"₩s{2,}")
#corpus = pattern6.sub(" ", corpus)
pattern["whitespace"] = pattern6
```

In [19]:

```python
from nltk import word_tokenize
from konlpy.tag import Komoran

ma = Komoran()
```

In [54]:

```python
documents = list()

for i in range(0,102):
    content = filecontent(fileids('')[i])
     #print(content)
    for _ in ["email", "punc", "stop", "whitespace"]:
        content = pattern[_].sub(" ", content)
        for token in word_tokenize(content):
            for term in ma.pos(token):
                if len(term[0]) > 1 and term[1].startswith("N"):
    #                   print(term[0])
    #                   print(type(term))
                    documents.append([term[0]])
```

In [57]:

```python
import random

a = 0.1
b = 0.1
K = 2 #군집으로 만듬..

docTermTopicMat = list()
Vocaburary = list()
random.seed(0)

for d in documents:
    termTopic = list()
    for t in d:
        termTopic.append([t, random.randrange(K)])
        Vocaburary.append(t)
    docTermTopicMat.append(termTopic)
Vocaburary = list(set(Vocaburary))

M = len(docTermTopicMat)
N = len(Vocaburary)
```

In [59]:

```python
from collections import defaultdict

topicTermMatrix = defaultdict(lambda:defaultdict(int))
docTopicMatrix = defaultdict(lambda:defaultdict(int))
for i, termTopic in enumerate(docTermTopicMat):
    for row in termTopic:
        topicTermMatrix[row[1]][row[0]] += 1
        docTopicMatrix[i][row[1]] += 1
```

In [60]:

```
def topicLikelihood(k,l):
    return (topicTermMatrix[k][l] + b) / ₩
           (sum(topicTermMatrix[k].values()) + (b*N))
```

In [61]:

```
def docLikelihood(m, k):
    return (docTopicMatrix[m][k] + a)
```

In [62]:

```
def topicAssign(m, l):
    totalCount = sum([f for k in range(K) for t,f in topicTermMatrix[k].items() if t == l])
    probList = list()
    for k in range(K):
        probList.append(topicLikelihood(k, l) * docLikelihood(m, k))

    _sum = sum(probList) * random.random()
    for i,p in enumerate(probList):
        _sum -= p
        if _sum <= 0:
            k = i
            break
    #(1) X (2)
    return k
```

In [63]:

```
_iter = 10

for _ in range(_iter):
    for i, termTopic in enumerate(docTermTopicMat):
        for row in termTopic:
            topicTermMatrix[row[1]][row[0]] -= 1
            docTopicMatrix[i][row[1]] -= 1

            k = topicAssign(i, row[0])

            row[1] = k
            topicTermMatrix[row[1]][row[0]] += 1
            docTopicMatrix[i][row[1]] += 1
            #count(_iter)
#             row[1] <= Topic Assign
#             row[0], row[1]
#    break
```

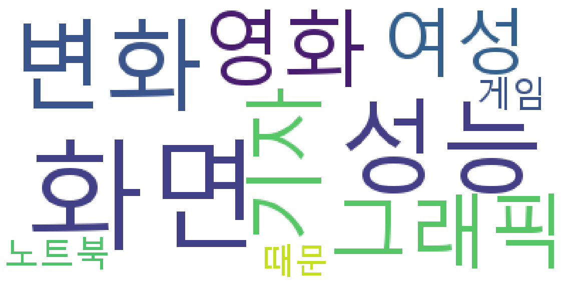
In [77]:

```
from wordcloud import WordCloud

font = 'C:/Windows/Fonts/malgun.ttf'
result = list()

for k,termList in  topicTermMatrix.items():
    #print(k, "번째 토픽")
    result.append(sorted(termList.items(), key=lambda x:x[1], reverse=True)[:10])
```

In [82]:

```
data = {x[0]:x[1] for x in result[0]}
wc = WordCloud(font, max_words=30, background_color="white")
wc.generate_from_frequencies(data)
wc.to_image()
```

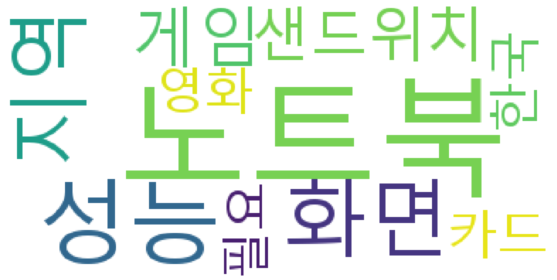
Out[82]:



In [83]:

```
data = {x[0]:x[1] for x in result[1]}
wc = WordCloud(font, max_words=30, background_color="white")
wc.generate_from_frequencies(data)
wc.to_image()
```

Out[83]:


가격
게임샌드위치
영화
노트북
성능
화면
카드