In [ ]:

```
'''
네이버 영화평 수집 및 전처리
'''
```

In [1]:

```
import requests

userAgent = {"user-agent":"Mozilla/5.0 (Macintosh; Intel Mac OS X 10_13_6) AppleWebKit/537.36 (K
HTML, like Gecko) Chrome/69.0.3497.100 Safari/537.36"}
getDownload = lambda url: requests.get(url, headers=userAgent)
```

In [2]:

```
import re

getCleanText = lambda item: re.sub(r"[\s]{2,}", " ", item.strip().replace(",", " "))
```

In [3]:

```
from bs4 import BeautifulSoup

url = "https://movie.naver.com/movie/point/af/list.nhn?target=after&page="

pageNo = 1
maxPageNo = 1001
lastReviewNo = 0

saveFilePath = "0630.txt"

fp = open(saveFilePath, "w", encoding="utf-8")
fp.close()

html = getDownload(url+str(pageNo))

while html.status_code == 200 and pageNo < maxPageNo:
    dom = BeautifulSoup(html.text, "html.parser")
    pointList = dom.select(".list_netizen tbody > tr")

    fp = open(saveFilePath, "a", encoding="utf-8")

    for pointInfo in pointList:
        pointItem = [point for point in pointInfo.select("td")]
        #print(pointItem[:5])
        pointID = getCleanText(pointItem[0].text)
        pointScore = getCleanText(pointItem[2].text)
        pointMovie = getCleanText(pointItem[3].select_one("a").text)
        #print(pointMovie[:5])
        pointComment = getCleanText(pointItem[3].contents[4])
        #print(pointComment)
        pointRegData = getCleanText(pointItem[4].contents[-1])

        if lastReviewNo == pointID:
            continue
        else:
            fp.write("{0},{1},{2},{3},{4}\n".format(pointID, pointScore, pointMovie, pointComme
nt, pointRegData))

    fp.close()

    lastReviewNo = pointID

    pageNo += 1
    html = getDownload(url+str(pageNo))

    if pageNo % 100 == 0:
        print("{0} pages were crawled.".format(pageNo))
```

```
100 pages were crawled.
200 pages were crawled.
300 pages were crawled.
400 pages were crawled.
500 pages were crawled.
600 pages were crawled.
700 pages were crawled.
800 pages were crawled.
900 pages were crawled.
1000 pages were crawled.
```

In [6]:

```
import pandas as pd
reviews = pd.read_csv("0630.txt", header=None ₩
                      , names = ["no", "score", "movie", "review", "regdate"])
reviews = reviews[reviews["review"].notnull()]
reviews = reviews.drop_duplicates()
```

In [7]:

```
from konlpy.tag import Komoran

ma = Komoran()
documents = list()

for review in reviews["review"]:
    documents.append([pos[0] for pos in ma.pos(review) if len(pos[0]) > 1 and pos[1] in ["NNG",
"NNP"]])
```

토픽 모델링(LDA-클러스터링)

In [8]:

```
from random import randrange

a = 0.1
b = 0.1
K = 5

docTermTopicMat = list()
Vocabulary = list()

for doc in documents:
    termTopic = list()

    for term in doc:
        termTopic.append([term, randrange(K)])
        Vocabulary.append(term)

    docTermTopicMat.append(termTopic)

Vocabulary = list(set(Vocabulary))

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

In [9]:

```
from collections import defaultdict

topicTermMatrix = defaultdict(lambda:defaultdict(int))
docTopicMatrix = defaultdict(lambda:defaultdict(int))
topicCount = defaultdict(int)

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

In [10]:

```
from random import random, choices

def topicLikelihood(k,l):
    return (topicTermMatrix[k][l] + b / topicCount[k] + (b*N))

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

def topicAssign(m, l):
    probList = list()
    for k in range(K):
        probList.append(topicLikelihood(k,l) * docLikelihood(m,k))

    return choices(range(K), probList, k=1)[0]
```

In [11]:

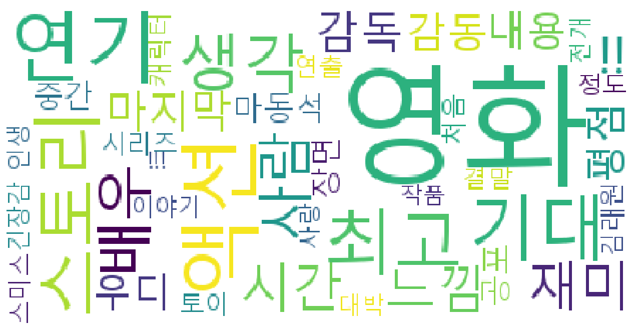
```
from tqdm import tqdm

_iter = 1000

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

        k = topicAssign(i, row[0])

        row[1] = k
        topicTermMatrix[row[1]][row[0]] += 1
        docTopicMatrix[i][row[1]] += 1
        topicCount[row[1]] += 1
```

```
100%|██████████████████████████████████████████████████████████████████████████
█| 1000/1000 [01:45<00:00,  9.44it/s]
```

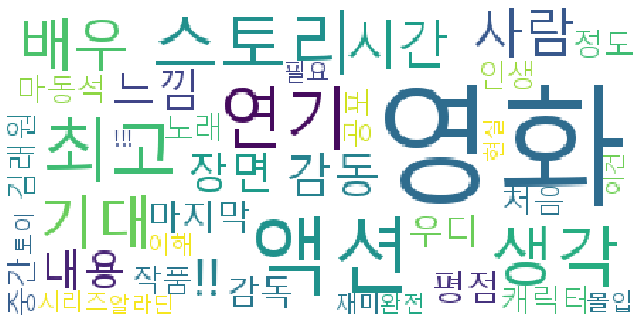
In [14]:

```
from wordcloud import WordCloud
import matplotlib.pyplot as plt

wc = WordCloud(font_path='C:/Windows/Fonts/malgun.ttf', background_color="white")

for k,name in zip(sorted(topicTermMatrix.keys()), ["알라딘", "토이스토리", "공포", "드라마", "액
션"]):
    print(str(k+1)+"번째 토픽 =>", name)
    print([term for term, freq in sorted(topicTermMatrix[k].items(), key=lambda x:x[1], reverse=
True)[:10]])
    wc.generate_from_frequencies({term:freq for term, freq in sorted(topicTermMatrix[k].items(),
key=lambda x:x[1], reverse=True)[:40]})
    plt.imshow(wc.to_array())
    plt.rcParams["figure.figsize"] = (8,6)
    plt.axis("off")
    plt.show()
```

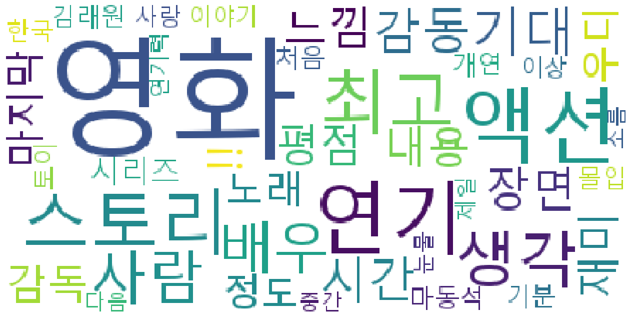
```
1번째 토픽 => 알라딘
['영화', '액션', '연기', '최고', '기대', '스토리', '생각', '배우', '사람', '재미']
```



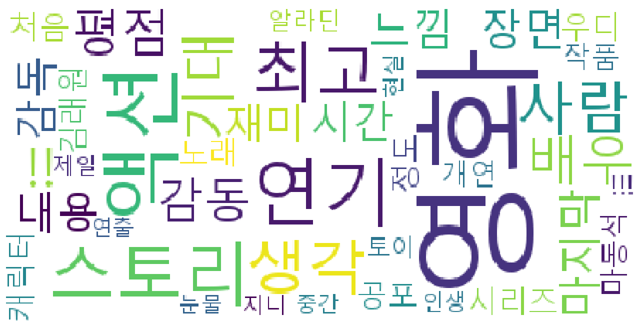
```
2번째 토픽 => 토이스토리
['영화', '액션', '최고', '스토리', '연기', '생각', '기대', '배우', '사람', '시간']
```



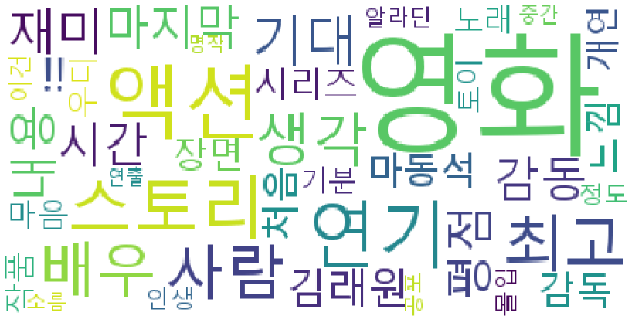
```
3번째 토픽 => 공포
['영화', '액션', '연기', '스토리', '최고', '생각', '사람', '배우', '기대', '시간']
```



```
4번째 토픽 => 드라마
['영화', '액션', '연기', '스토리', '최고', '생각', '사람', '기대', '!!', '배우']
```



```
5번째 토픽 => 액션
['영화', '액션', '연기', '스토리', '최고', '사람', '배우', '생각', '기대', '재미']
```



```
In [ ]:
```