In [1]:

```
from os import listdir

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

In [2]:

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

In [3]:

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

In [4]:

```
import re
from string import punctuation

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"Ww{2,}@Ww{3,}.(.Ww{2,})+")
pattern3 = re.compile(r"Ww{2,}@(.?Ww{2,})+")
#corpus = pattern3.sub("", corpus)
pattern["email"] = pattern3

#도메인
pattern4 = re.compile(r"(.?Ww{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"Ws{2,}")
#corpus = pattern6.sub(" ", corpus)
pattern["whitespace"] = pattern6
```

In [6]:

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

ma = Komoran()

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 [7]:

```
from collections import defaultdict
from konlpy.tag import Komoran

ma = Komoran()

DTM = defaultdict(lambda:defaultdict(int))
for idx in fileids(''):
    for term in filecontent(idx).split():
        for token in ma.morphs(term):
            for term in ma.pos(token):
                if len(term[0]) > 1 and term[1].startswith("N"):
                    DTM[idx][""].join(term[0])] += 1
```

In [8]:

```
TDM = defaultdict(lambda:defaultdict(int))
for d, termList in DTM.items():
    for t, f in termList.items():
        TDM[t][d] = f
```

In [9]:

```
from math import log2

TWM = defaultdict(lambda:defaultdict(float))
N = len(DTM)
for term, docList in TDM.items():
    df = len(docList)
    for d,f in docList.items():
        maxFreq = max(DTM[d].values())
        TF = f/maxFreq
        IDF = log2(N/df)
        TWM[term][d] = TF*IDF
```

In [10]:

```
docVectorList = [[0 for _ in range(len(TDM))] for _ in range(N)]
D = list(DTM.keys())
V = list(TDM.keys())
for t, docList in TWM.items():
    for d,w in docList.items():
        docVectorList[D.index(d)][V.index(t)] = w
```

In [11]:

```
from math import sqrt

def distance(x1, x2):
    _sum = 0
    for i in range(len(x1)):
        _sum += (x1[i]-x2[i])**2
    return sqrt(_sum)

def angle(x1, x2):
    _innerProduct = 0
    for i in range(len(x1)):
        _innerProduct += x1[i]*x2[i]
    x1VecLength = distance(x1, [0 for _ in range(len(x1))])
    x2VecLength = distance(x2, [0 for _ in range(len(x2))])
    return _innerProduct/(x1VecLength*x2VecLength)
```

In [12]:

```
def expectation(x, c, opt=True):
    candidates = list()

    nearest = distance if not opt else angle
    best = min if not opt else max

    for _ in c:
        candidates.append(nearest(x, _))
    return candidates.index(best(candidates))
```

In [13]:

```
def maximization(X):
    _sum = [0 for _ in range(len(X[0]))]
    D = len(X)
    for x in X:
        for i in range(len(x)):
            _sum[i] += x[i]
    return [_/D for _ in _sum]
```

In [14]:

```
def sse(X, c, opt=True): # sum squard error
    error = 0
    nearest = distance if not opt else angle
    for x in X:
        error += nearest(x,c)
    return error
```

In [15]:

```
from random import randrange

K = 6
centroid = [[randrange(0,6) for _ in range(len(V))] for _ in range(K)]
_iter = 10
sseList = list()

for _ in range(_iter):
    rnk = list(list(0 for _ in range(K)) for _ in range(len(D)))
    for i,x in enumerate(docVectorList):
        idx = expectation(x, centroid)
        rnk[i][idx] = 1

    _sum = 0.0
    for k in range(K):
        _X = [docVectorList[i] for i in range(len(D)) if rnk[i][k]]
        _sum += sse(_X, centroid[k]) #sum squared error
        centroid[k] = maximization(_X)
    sseList.append(_sum)
```

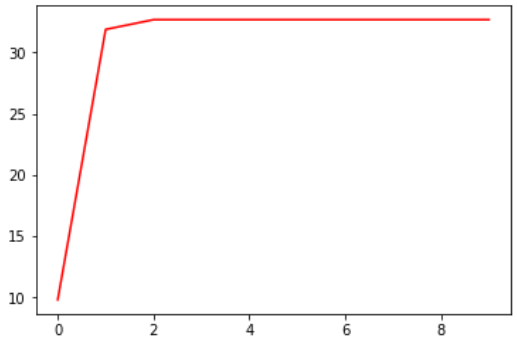
In [17]:

```
import matplotlib.pyplot as plt

plt.plot(range(_iter), sseList, "r-")
plt.show
```

Out[17]:

```
<function matplotlib.pyplot.show(*args, **kw)>
```



In [49]:

```
result = list()
for k in range(K):
    W = [(V[i],centroid[k][i]) for i in range(len(V)) if centroid[k][i]]
    result.append(sorted(W, key=lambda x:x[1], reverse=True)[:10])
    #print(sorted(W, key=lambda x:x[1], reverse=True)[:5])
result
```

Out[49]:

```
[[('여성', 0.8473285306249636),
  ('연극제', 0.8040714059027576),
  ('영화', 0.7576424302528226),
  ('연극', 0.6432571247222061),
  ('백운', 0.6010434373486335),
  ('이드', 0.6010434373486335),
  ('근로자', 0.6010434373486335),
  ('차등', 0.6010434373486335),
  ('황교안', 0.6010434373486335),
  ('설화', 0.6010434373486335)],
 [('정상회담', 1.1248318927695724),
  ('평양', 1.0027620400405004),
  ('김정은', 0.7718648601600209),
  ('중국', 0.7653794438393698),
  ('웹툰', 0.6965752286430307),
  ('시진핑', 0.6941712717204406),
  ('북한', 0.6533376675015913),
  ('국무위원', 0.6507880193506059),
  ('캡처', 0.6507880193506059),
  ('장이', 0.648439843549662)],
 [('샌드위치', 0.8322139901750311),
  ('사찰', 0.8322139901750311),
  ('환영', 0.6241604926312733),
  ('국립공원', 0.6185164660790444),
  ('관람료', 0.554809326783354),
  ('평양', 0.5370102939704072),
  ('증명사진', 0.49303007201059246),
  ('오디오북', 0.49303007201059246),
  ('중국', 0.47499428326928 3),
  ('마을', 0.4458289233080523)],
 [('성능', 1.1354063447917608),
  ('내일', 1.0888997628616552),
  ('노트북', 1.088097747092104),
  ('소식', 0.8073793363657151),
  ('지역', 0.7860317125866961),
  ('화면', 0.7524864494435026),
  ('고요', 0.7414124642968433),
  ('경북', 0.7064569193200007),
  ('그래픽', 0.6623203677951939),
  ('내륙', 0.6403619580586839)],
 [('종교', 0.7727701337339574),
  ('신뢰', 0.662374400343392),
  ('동물', 0.6557010869564154),
  ('산업', 0.6299129908768145),
  ('잡지', 0.5795776003004681),
  ('지역', 0.557579145874 3067),
  ('지용', 0.5519786669528267),
  ('소장', 0.5519786669528267),
  ('미디어', 0.5424250754772569),
  ('성락원', 0.5349947079696628)],
 [('직장인', 2.1637563744550805),
  ('코리안', 1.6228172808413106),
  ('인성', 1.4425042496367204),
  ('채용', 1.4425042496367204),
  ('지원자', 1.4425042496367204),
  ('구직', 1.4425042496367204),
  ('학창', 1.4425042496367204),
  ('자사', 1.2818781872275404),
  ('공교육', 1.2818781872275404),
  ('아이', 1.1297713741666182)]]
```

In [63]:

```
from wordcloud import WordCloud

font = 'C:/Windows/Fonts/malgun.ttf'


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[63]:

In [64]:

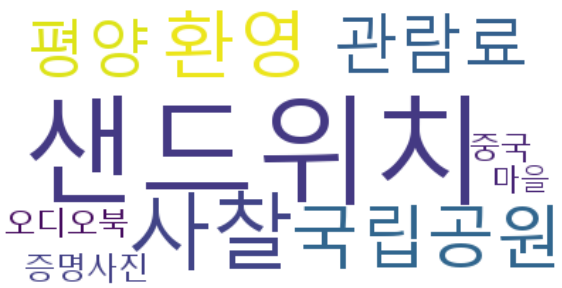
```
data = {x[0]:x[1] for x in result[1]}
wc.generate_from_frequencies(data)
wc.to_image()
```

Out[64]:

In [65]:

```
data = {x[0]:x[1] for x in result[2]}
wc.generate_from_frequencies(data)
wc.to_image()
```

Out[65]:



In [66]:

```
data = {x[0]:x[1] for x in result[3]}
wc.generate_from_frequencies(data)
wc.to_image()
```

Out[66]:

In [67]:

```
data = {x[0]:x[1] for x in result[4]}
wc.generate_from_frequencies(data)
wc.to_image()
```

Out[67]:

In [68]:

```
data = {x[0]:x[1] for x in result[5]}
wc.generate_from_frequencies(data)
wc.to_image()
```

Out[68]: