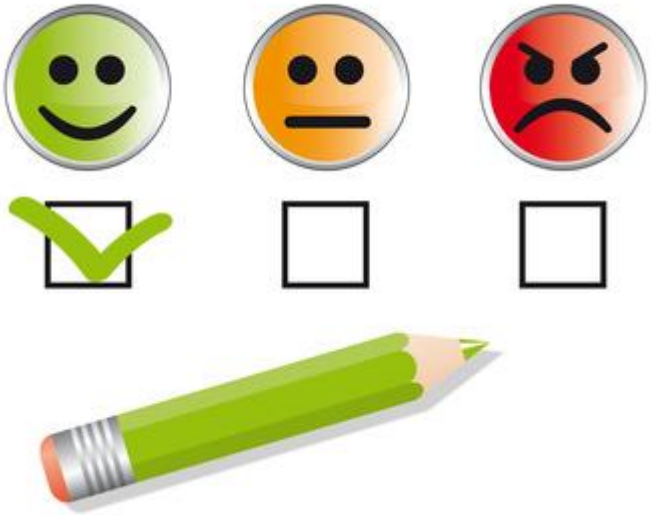
Text Analysis 실습

감성분석

박진수 교수

Big Data Institute,
Seoul National University

감성분석(Sentiment Analysis) 개요



감성분석(Sentiment Analysis)

· 감성분석: 발화자의 **감정 상태(affective states)**나 발화자에 대한 **주관적인 정보**를 밝혀내고, 추출하고, 계량화하기 위한 시도[source: https://en.wikipedia.org/wiki/Sentiment_analysis]

- 이를 위해 자연어 처리(natural language processing), 텍스트 분석(text analysis), 전산언어학(computational linguistics), 생물 측정학(biometrics) 등이 활용된다

- 고객들의 후기나 인터뷰 결과 등을 분석하는 "고객의 소리(voice of customer)"나 온라인 커뮤니티 및 소셜 미디어 분석, 헬스케어 등 다양한 분야에 폭넓게 활용된다

- 기본적으로 텍스트가 '긍정(positive)' 혹은 '부정(negative)'인지를 밝혀내기 위함이었지만, 나아가 다양한 감정 상태(happy, angry, sad, …) 및 주관성/객관성을 판별해내려는 시도도 자주 행해지고 있다

감성분석(Sentiment Analysis)

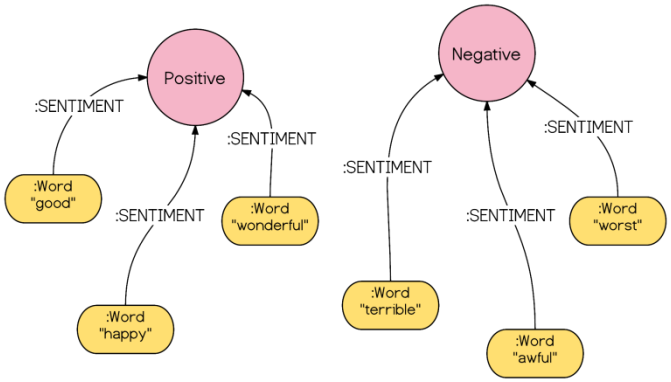
· 감성분석의 방법론: 지식 기반 테크닉(knowledge-based techniques), 통계적 방법론(statistical methods), 그리고 하이브리드 방식(hybrid approaches)의 세 가지 방식이 있다

- **지식 기반 테크닉(knowledge-based techniques):** 기존에 존재하는 단어들의 소위 '감정 사전(emotion dictionary)'을 활용해 단어와 글의 감정 상태를 분류

- 센티워드넷(SentiWordNet): 대규모의 어휘 데이터베이스(lexical database)인 워드넷(wordnet)을 확장해 각 단어의 긍정/부정 척도를 더함[Baccianella et al 2010]

  ■ 단어를 유의어 집단인 "synset"으로 분류하고 이들의 주관적인 감정 척도를 매김

  ■ Python의 nltk 패키지에 포함되어 있음



감성분석(Sentiment Analysis)

· 감성분석의 방법론: 지식 기반 테크닉(knowledge-based techniques), 통계적 방법론(statistical methods), 그리고 하이브리드 방식(hybrid approaches)의 세 가지 방식이 있다

- 센티워드넷 상의 긍정/부정 유의어 집단

Table 1: The 10 top-ranked positive synsets and the 10 top-ranked negative synsets in SENTIWORDNET 3.0.

Rank	Positive	Negative
1	good#n#2 goodness#n#2	abject#a#2
2	better_off#a#1	deplorable#a#1 distressing#a#2 lamentable#a#1 pitiful#a#2 sad#a#3 sorry#a#2
3	divine#a#6 elysian#a#2 inspired#a#1	bad#a#10 unfit#a#3 unsound#a#5
4	good_enough#a#1	scrimy#a#1
5	solid#a#1	cheapjack#a#1 shoddy#a#1 tawdry#a#2
6	superb#a#2	unfortunate#a#3
7	good#a#3	inauspicious#a#1 unfortunate#a#2
8	goody-goody#a#1	unfortunate#a#1
9	amiable#a#1 good-humored#a#1 good-humoured#a#1	dispossessed#a#1 homeless#a#2 roof-less#a#2
10	gainly#a#1	hapless#a#1 miserable#a#2 misfortu-nate#a#1 pathetic#a#1 piteous#a#1 pitiable#a#2 pitiful#a#3 poor#a#1 wretched#a#5

감성분석(Sentiment Analysis)

· 감성분석의 방법론: 지식 기반 테크닉(knowledge-based techniques), 통계적 방법론(statistical methods), 그리고 하이브리드 방식(hybrid approaches)의 세 가지 방식이 있다

- **통계적 방법론(statistical methods):** LSA(Latent semantic analysis), 지지벡터머신(support vector machines), “bag of words” 등 다양한 기계학습 방법론을 활용해 각 텍스트의 감정을 “분류(classify)”하는 것

- 통계적 방법론의 경우 사전 지식(knowledge base)이 없어도 되지만 학습 데이터(training data)가 필수적임

■ 인코딩된 학습 데이터와 검증 데이터를 가지고 감정 분류가 맞는지 틀린지 검증해볼 수 있음

■ Python의 nltk 패키지에 naïve bayes clasisfier를 활용한 감정 분석 툴이 제공됨

감성분석(Sentiment Analysis)

· 감성분석의 방법론: 지식 기반 테크닉(knowledge-based techniques), 통계적 방법론(statistical methods), 그리고 하이브리드 방식(hybrid approaches)의 세 가지 방식이 있다

- 하이브리드 방식(hybrid approaches): 지식 기반 테크닉과 통계적 방법론을 혼합하여 활용하는 방식

- 각 방식의 장단점을 잘 파악해 단점을 최소화하고 장점을 극대화하기 위한 시도

감성분석(Sentiment Analysis)

· 감성분석의 어려움: 자연어 데이터에 **노이즈(noise)**가 많다! → 문맥(context)을 고려해야 하는 경우가 많다!

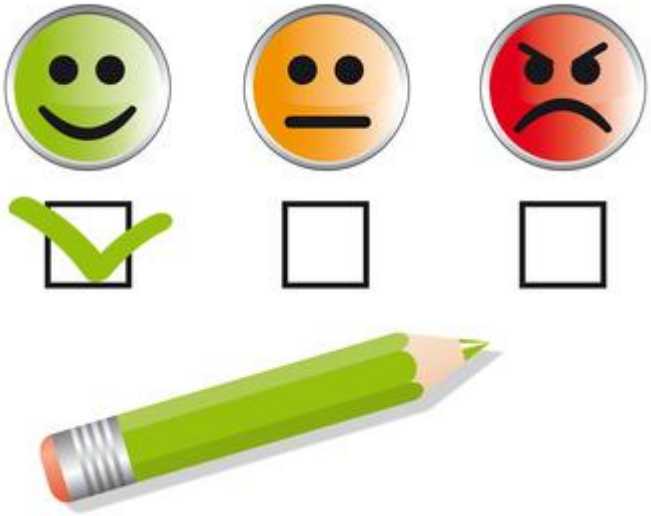
- 감성분석이 쉬운 텍스트

```
"I hate you."
"I dislike dogs"
"iPhone is the best invention of the 21st century
```

- 감성분석이 어려운 텍스트

```
"I do not hate you." (Negation handling)
"I pigged out last dinner and screwed up this morning" (slang)
"I dslike dogs" (typo)
"This movie is truly the best of 2017!" (sarcastic)
"I like the design of iPhone, but would not buy the iPhone 8 since it is too
wide" (qualification)
"Disliking small dogs is not my kind of thing" (Negation)
```

IMDb Large Movie Review Dataset



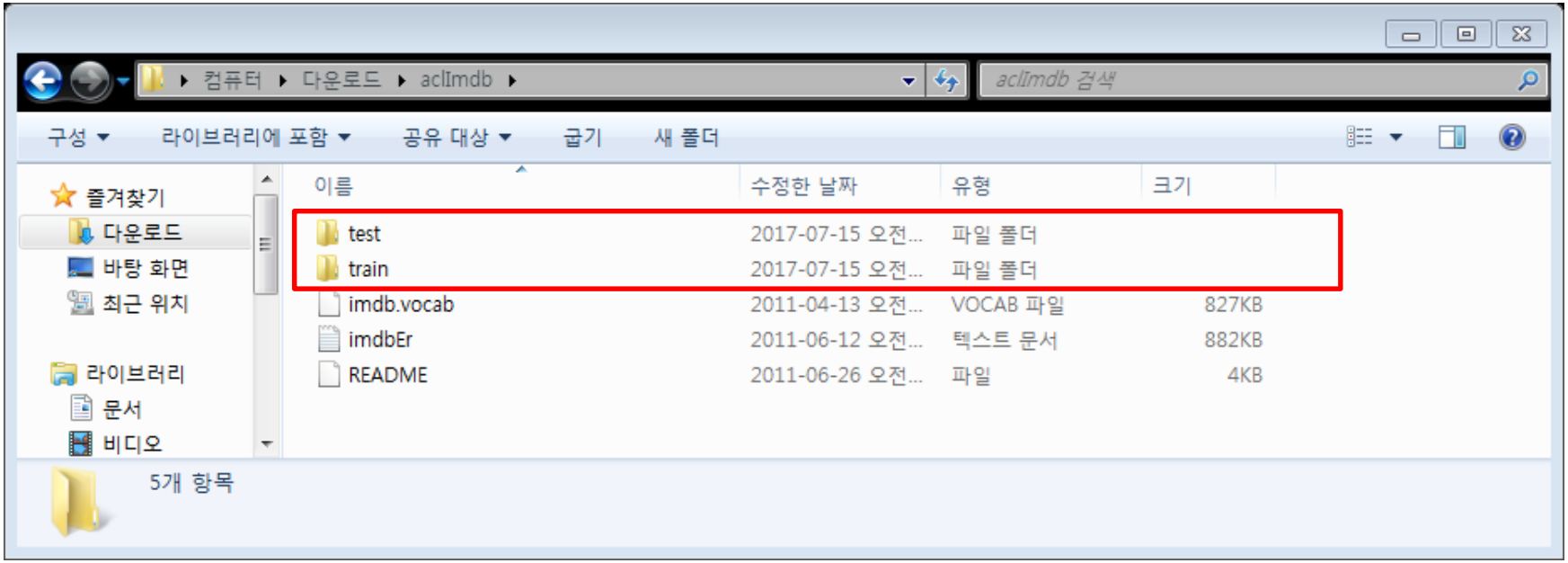
IMDb Large Movie Review Dataset

· IMDb의 영화 리뷰 50,000개를 모아 25,000개씩의 학습 데이터와 검증 데이터로 나눈 후 감성 지수를 1(positive)와 0(negative)로 분류하여 나눈 데이터셋

- 다운로드: http://ai.stanford.edu/~amaas/data/sentiment/

- 데이터 셋을 다운받아 압축을 푼다

■ 'aclImdb' 라는 이름의 디렉토리 아래 폴더와 파일들이 생긴다



IMDb Large Movie Review Dataset

· 데이터 불러오기

- 압축을 푼 폴더(aclImdb)를 Python 파일과 같은 디렉토리에 저장한다

- 긍정적인 리뷰의 트레이닝 데이터 중 첫 번째 리뷰의 텍스트 출력하기

In [1]:
```
import os
files = os.listdir('aclImdb/train/pos')
```

In [2]:
```
first_file = files[0]
with open('aclImdb/train/pos/{}'.format(first_file), 'r', encoding = 'utf-8') as f:
    review = f.read()
    f.close()
```

IMDb Large Movie Review Dataset

· 데이터 불러오기

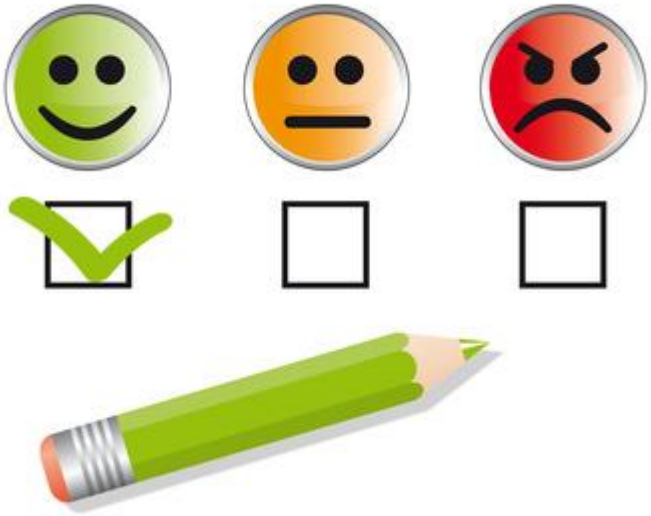
- 전체 긍정 트레이닝 데이터 저장하기

In [23]:
```
pos_train_list = []
for file in files:
    with open('aclImdb/train/pos/{}'.format(file), 'r', encoding = 'utf-8') as f:
        review = f.read()
        f.close()
    pos_train_list.append(review)
print(len(pos_train_list))
```

```
12500
```

- 위와 같은 방식으로 다른 데이터도 쉽게 저장할 수 있다

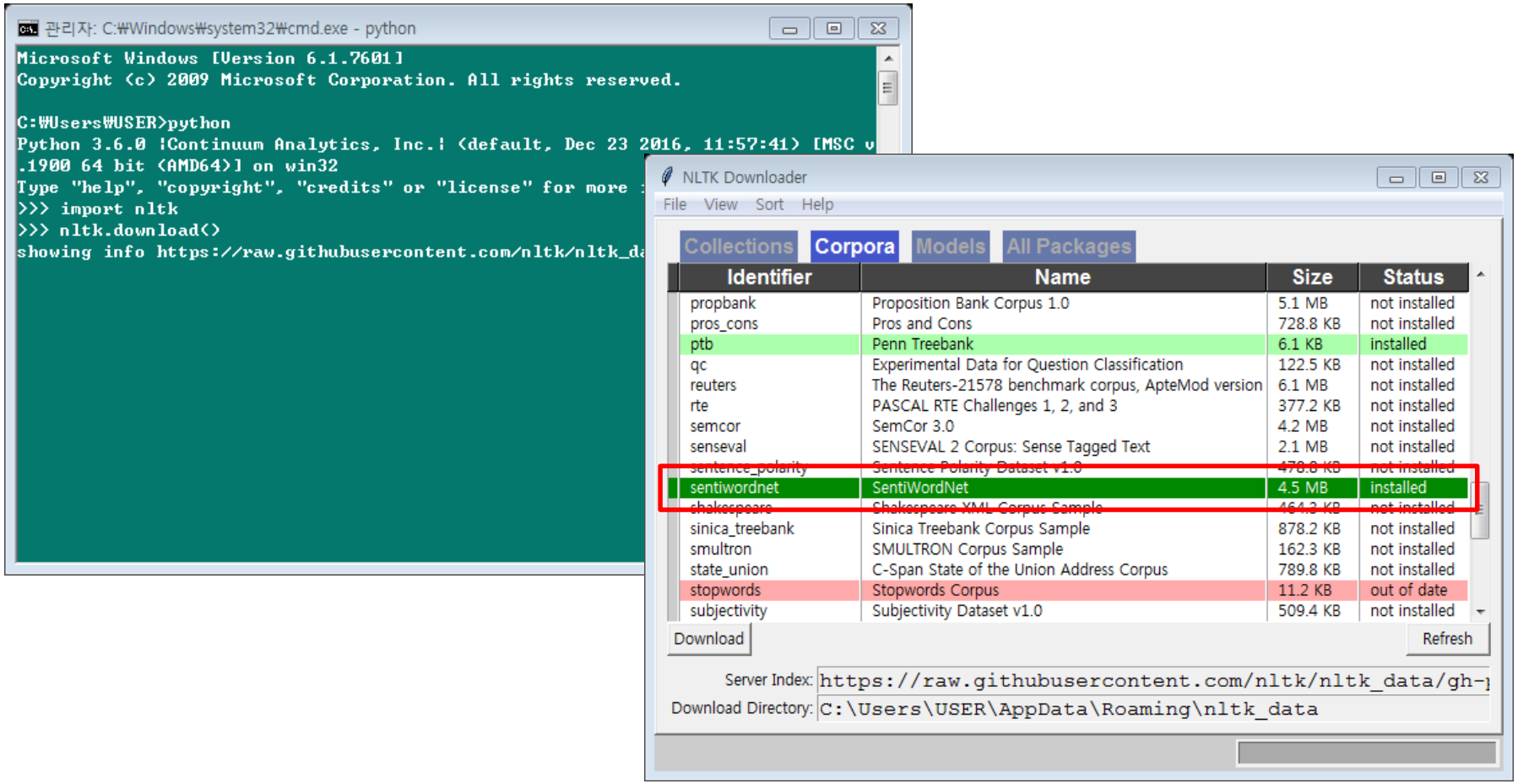
센티워드넷(SentiWordNet)



센티워드넷을 활용한 감성분석

· 센티워드넷 resource 다운로드

- nltk.download()를 통해 sentiwordnet corpora를 다운받는다



센티워드넷을 활용한 감성분석

· 센티워드넷 불러오기

```
In [1]: import nltk
        from nltk.corpus import sentiwordnet as swn
```

· 단어의 synset 찾기

```
In [6]: swn.senti_synsets('hate')
Out[6]: <filter at 0xf5986d8>

In [5]: list(swn.senti_synsets('hate'))
Out[5]: [SentiSynset('hate.n.01'), SentiSynset('hate.v.01')]

In [7]: list(swn.senti_synsets('hate', 'v'))
Out[7]: [SentiSynset('hate.v.01')]
```

- senti_synsets()의 결과는 filter object이므로 내용을 보고싶으면 리스트로 변환한다

- ‘hate’의 경우 명사(noun)와 동사(verb) 두 가지 유의어 집합이 존재한다

- 단어의 품사(‘v’, ‘n’, ‘r’, ‘a’ 등)를 인자값으로 줄 수도 있다

센티워드넷을 활용한 감성분석

· 단어의 긍정/부정 척도

```
In [8]: list(swn.senti_synsets('hate', 'v'))[0].pos_score()
Out[8]: 0.0

In [9]: list(swn.senti_synsets('hate', 'v'))[0].neg_score()
Out[9]: 0.75
```

- pos_score(): 단어의 긍정 척도

- neg_score(): 단어의 부정 척도

- ‘hate’라는 동사는 긍정 척도는 0이지만 부정 척도는 0.75이므로 부정적인 의미가 강한 단어라고 할 수 있다

센티워드넷을 활용한 감성분석

· 특정 단어의 긍정/부정 지수를 계산하는 함수 정의

In [43]:
```
def word_sentiment_calculator(word, tag):
    pos_score = 0
    neg_score = 0

    if 'NN' in tag and len(list(swn.senti_synsets(word, 'n')))>0:
        syn_set = list(swn.senti_synsets(word, 'n'))
    elif 'VB' in tag and len(list(swn.senti_synsets(word, 'v')))>0:
        syn_set = list(swn.senti_synsets(word, 'v'))
    elif 'JJ' in tag and len(list(swn.senti_synsets(word, 'a')))>0:
        syn_set = list(swn.senti_synsets(word, 'a'))
    elif 'RB' in tag and len(list(swn.senti_synsets(word, 'r')))>0:
        syn_set = list(swn.senti_synsets(word, 'r'))
    else:
        return (0,0)

    for syn in syn_set:
        pos_score += syn.pos_score()
        neg_score += syn.neg_score()
    return (pos_score/len(syn_set), neg_score/len(syn_set))
```

- 특정 단어가 속한 유의어 집합의 positive score와 negative score를 모두 합한 후 평균을 낸다

- Positive score와 negative score를 tuple로 표현하여 반환

센티워드넷을 활용한 감성분석

· 특정 단어('love')의 긍정/부정 지수 계산하기

```
In [44]: nltk.pos_tag(['love'])
Out[44]: [('love', 'NN')]

In [45]: word_sentiment_calculator('love', 'NN')
Out[45]: (0.22916666666666666, 0.0)

In [46]: word_sentiment_calculator('love', 'VB')
Out[46]: (0.625, 0.03125)
```

- 'love'라는 단어의 명사형의 긍정/부정 지수는 (0.229, 0)이다

- 'love'라는 단어의 동사형의 긍정/부정 지수는 (0.625, 0.03125)이다

- 즉, 'love'라는 단어는 전반적으로 긍정적인 감성을 담고 있다고 할 수 있다

센티워드넷을 활용한 감성분석

· 특정 문장의 긍정/부정 지수 계산하기

- 문장을 토큰화한 후 품사 태깅을 한다

In [47]:
```
sent = 'I hate you.'
```

In [49]:
```
tokens = nltk.word_tokenize(sent)
pos_tags = nltk.pos_tag(tokens)
pos_tags
```

Out[49]:
```
[('I', 'PRP'), ('hate', 'VBP'), ('you', 'PRP'), ('.', '.')]
```

- 각 토큰의 부정 지수와 긍정 지수를 모두 합한다

In [50]:
```
pos_score = 0
neg_score = 0
```

In [51]:
```
for word, tag in pos_tags:
    pos_score += word_sentiment_calculator(word, tag)[0]
    neg_score += word_sentiment_calculator(word, tag)[1]
print(pos_score)
print(neg_score)
```

```
0.0
0.75
```

센티워드넷을 활용한 감성분석

· 특정 문장의 긍정/부정 지수 계산하기

- 문장의 감성 지수를 계산하기 위한 함수 정의

In [52]:
```
def sentence_sentiment_calculator(pos_tags):
    pos_score = 0
    neg_score = 0
    for word, tag in pos_tags:
        pos_score += word_sentiment_calculator(word, tag)[0]
        neg_score += word_sentiment_calculator(word, tag)[1]
    return (pos_score, neg_score)
```

In [53]:
```
sentence_sentiment_calculator(pos_tags)
```

Out [53]:
```
(0.0, 0.75)
```

센티워드넷을 활용한 감성분석

· 실습 4-1-1. 아래 문장의 감성분석을 수행해 본다

- 문장1: “In the Echo Dot, Amazon has created a near perfect blend of hardware and software.”

- 문장2: “The author does a good job of presenting a wide range of psychological traps and irrational tendencies to which humans fall prey”

- 문장3: “Pulp Fiction is inane, self-indulgent, and bloated”

- 수행 예시

In [8]:
```
print(sentence_sentiment_calculator(sent1))
print(sentence_sentiment_calculator(sent2))
print(sentence_sentiment_calculator(sent3))
```

```
(1.1239035087719298, 0.2609649122807018)
(1.1782757173382172, 0.882884043040293)
(0.778846153846153 9, 0.994230769230769 3)
```

센티워드넷을 활용한 감성분석

· 실습 4-1-2. 센티워드넷을 활용한 영화 리뷰 감성분석 (1)

- 실습 1-3-4에서 수집했던 영화 '다크 나이트'의 첫 번째 리뷰를 불러와 감성분석을 수행해 본다

- 센티워드넷을 활용한다

- 수행 예시

```
In [10]: sentence_sentiment_calculator(review)
Out[10]: (11.405738623218884, 6.483186195266968)
```

센티워드넷을 활용한 감성분석

· 실습 4-1-3. 센티워드넷을 활용한 영화 리뷰 감성분석 (2)

- 실습 1-3-4에서 수집했던 영화 '다크 나이트'의 모든 리뷰의 감성분석을 수행한다

- 모든 리뷰의 감성 지수를 2차원 NumPy 배열에 저장한 후 긍정/부정 지수의 평균을 계산한다

- np.mean() 함수를 활용한다(axis 인자값을 활용할 것)

- 수행 예시

```
In [20]: mean_scores
Out[20]: array([ 13.4369763 ,  10.19789012])
```

센티워드넷을 활용한 감성분석

· 실습 4-1-4. 센티워드넷을 활용한 영화 리뷰 감성분석 (3)

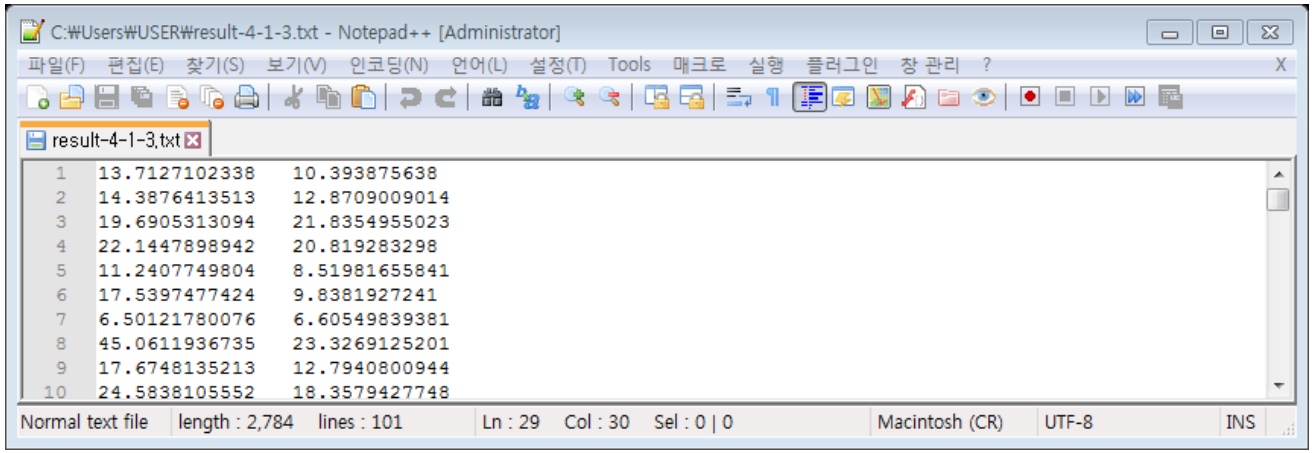
- 실습 1-3-5에서 수집했던 임의의 영화의 모든 리뷰의 감성분석을 수행한다

- 모든 리뷰의 감성 지수를 2차원 NumPy 배열에 저장한다

- 이를 open()함수를 활용해 텍스트 파일에 저장한다

  ■ 각 리뷰당 한 줄에 출력되고, 긍정과 부정 지수의 구분은 탭(tab)으로 한다

- 수행 예시



센티워드넷을 활용한 감성분석

· 실습 4-1-5. 센티워드넷을 활용한 IMDb Large Movie Review Dataset 감성분석 (1)

- IMDb Large Movie Review Dataset에서 첫 번째 긍정적인 학습 리뷰 데이터를 불러와 감성 분석을 해본다

- [aclImdb] > [train] > [pos]> 0_9.txt의 데이터를 분석해 본다

- 수행 예시

In [59]:
```
print(review)
sentence_sentiment_calculator(review)
```

```
Bromwell High is a cartoon comedy. It ran at the same time as some other programs about school life, such as "T
eachers". My 35 years in the teaching profession lead me to believe that Bromwell High's satire is much closer
 to reality than is "Teachers". The scramble to survive financially, the insightful students who can see right
 through their pathetic teachers' pomp, the pettiness of the whole situation, all remind me of the schools I kn
ew and their students. When I saw the episode in which a student repeatedly tried to burn down the school, I im
mediately recalled ......... at ........... High. A classic line: INSPECTOR: I'm here to sack one of your teache
rs. STUDENT: Welcome to Bromwell High. I expect that many adults of my age think that Bromwell High is far fetc
hed. What a pity that it isn't!
```

Out [59]:
```
(5.110205982026105, 3.502637966031258)
```

센티워드넷을 활용한 감성분석

· 실습 4-1-6. 센티워드넷을 활용한 IMDb Large Movie Review Dataset 감성분석 (2)

- IMDb Large Movie Review Dataset에서 긍정적인 학습 리뷰 데이터 10개와 부정적인 학습 리뷰 데이터 10개를 불러와 감성 분석을 해본다

- 총 20개 중에 정확도(accuracy)가 몇이나 되는지 확인해 본다

■ 긍정 척도가 부정 척도보다 높으면 positive로, 부정 척도가 긍정 척도보다 높으면 negative로 분류한다

- 수행 예시

In [68]:
```
print(actual)
print(predicted)
```

```
[1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
[1, 0, 1, 0, 1, 1, 1, 1, 1, 0, 1, 0, 1, 1, 1, 0, 0, 0, 0, 1]
```

In [70]:
```
print('Number of correct instances: ', correct)
print('Number of incorrect instances: ', incorrect)
```

```
Number of correct instances:  12
Number of incorrect instances:  8
```

■ 분류 결과 20개 중 12개를 정확히 분류하였고 8개를 틀리게 분류하였다(60%의 정확도)

센티워드넷을 활용한 감성분석

· 실습 4-1-7. 센티워드넷을 활용한 IMDb Large Movie Review Dataset 감성분석 (3)

- IMDb Large Movie Review Dataset에서 긍정적인 **검증** 리뷰 데이터(test data)와 부정적인 검증 **리뷰** 데이터(test data)를 1000개씩 불러와 감성 분석을 해본다

- 총 2000개 중에 정확도(accuracy)가 얼마나 되는지 살펴본다

■ 긍정 척도가 부정 척도보다 높으면 positive로, 부정 척도가 긍정 척도보다 높으면 negative로 분류한다

- 수행 예시

```
In [45]: print('Number of correct instances: ', correct)
         print('Number of incorrect instances: ', incorrect)

Number of correct instances:  1425
Number of incorrect instances:  575
```

■ 분류 결과 2000개 중 1425를 정확히 분류하였고 575개를 틀리게 분류하였다(71%의 정확도)

센티워드넷을 활용한 감성분석

· 실습 4-1-8. 센티워드넷을 활용한 IMDb Large Movie Review Dataset 감성분석 (4)

- IMDb Large Movie Review Dataset에서 긍정적인 **검증** 리뷰 데이터(test data)와 부정적인 검증 **리뷰** 데이터(test data) 전체를 불러와 감성 분석을 해본다

- 총 25000개 중에 정확도(accuracy)가 얼마나 되는지 살펴본다

  ■ 긍정 척도가 부정 척도보다 높으면 positive로, 부정 척도가 긍정 척도보다 높으면 negative로 분류한다

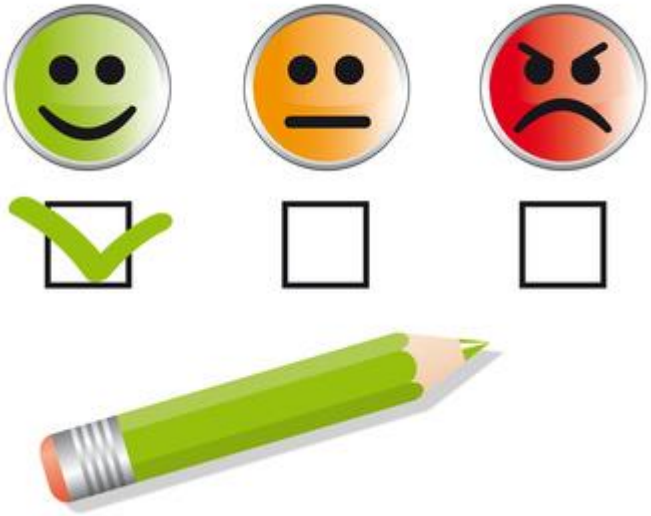
- 수행 예시

In [9]:
```
print('Number of correct instances: ', correct)
print('Number of incorrect instances: ', incorrect)
```

```
Number of correct instances:  17057
Number of incorrect instances:  7943
```

  ■ 분류 결과 25000개 중 17057를 정확히 분류하였고 7943를 틀리게 분류하였다(68%의 정확도)

나이브 베이즈 감정 분류



나이브 베이즈 분류기를 활용한 감성분석

· 벡터 공간 모형(Vector space model): 문서의 질의(query)가 모두 벡터로 표현되며, 각각의 차원은 개별 단어에 대응된다. 대표적으로 TF-IDF 모델이 있다
[source: https://ko.wikipedia.org/wiki/%EB%B2%A1%ED%84%B0_%EA%B3%B5%EA%B0%84_%EB%AA%A8%EB%8D%B8]

- 어떤 단어가 문서에 포함되면, 해당 단어에 해당하는 벡터값은 0이 아닌 값을 갖는다
  - 단어가 등장하면 무조건 1로 표현할 수도 있고(boolean model), 단어가 등장한 횟수로도 표현 가능함(frequency)
- 어떤 단어가 문서에 포함되지 않으면, 해당 단어에 해당하는 벡터값은 0으로 설정한다
- 예시: 두 개의 문장(document)을 벡터로 표현하기
  - 문장1: “This is a car that I gave Jane”
  - 문장2: “That car is for Tony, but that bicycle is for Jane”

	This	Car	That	I	Bicycle	Tony
문장1	1	1	1	1	0	0
문장2	0	1	1	0	1	1

나이브 베이즈 분류기를 활용한 감성분석

· 자주 등장하는 단어를 통해 feature 생성하기

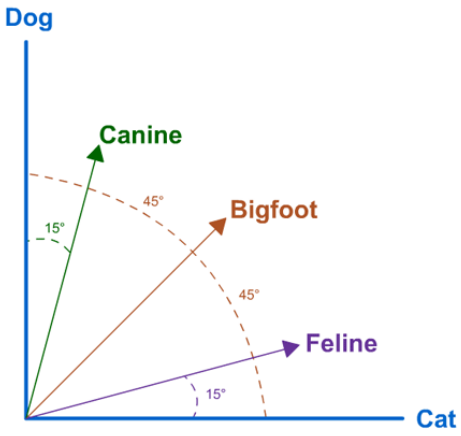
- Training corpus의 데이터 중 자주 등장하는 3000개만 선택한다

- 단어 하나가 하나의 dimension(feature)가 된다

  ■ 벡터 공간 모형(vector space model)의 가장 기초적인 형태

- 하나의 document에서 특정 word가 등장하면 그 word feature는 1(True), 등장하지 않으면 0(False)으로 인코딩함

- Feature를 활용해 기계학습 알고리즘을 통한 학습과 검증을 진행한다



나이브 베이즈 분류기를 활용한 감성분석

· 자주 등장하는 단어를 통해 feature 생성하기

- 긍정 학습 데이터와 부정 학습 데이터의 모든 토큰을 리스트에 저장한다

■ stopwords를 제거한다

In [74]:
```
import os
files = os.listdir('aclImdb/train/pos')
```

In [75]:
```
from nltk.corpus import stopwords
stopWords = set(stopwords.words('english'))
```

In [76]:
```
words = []
for file in files:
    with open('aclImdb/train/pos/{}'.format(file), 'r', encoding = 'utf-8') as f:
        review = nltk.word_tokenize(f.read())
        for token in review:
            if token not in stopWords:
                words.append(token)
print(len(words))
```

```
2293566
```

In [77]:
```
files = os.listdir('aclImdb/train/neg')
```

In [78]:
```
for file in files:
    with open('aclImdb/train/neg/{}'.format(file), 'r', encoding = 'utf-8') as f:
        review = nltk.word_tokenize(f.read())
        for token in review:
            if token not in stopWords:
                words.append(token)
print(len(words))
```

```
4557590
```

나이브 베이즈 분류기를 활용한 감성분석

· 자주 등장하는 단어를 통해 feature 생성하기

- 단어의 분포(distribution)를 활용해 3000차원의 word feature를 생성한다

In [89]:
```
words = nltk.FreqDist(words)
word_features = list(words.keys())[:3000]
```

- Feature를 생성하는 함수를 정의한다

In [87]:
```
def find_features(doc):
    words = set(doc)
    features = {}
    for w in word_features:
        features[w] = (w in words)
    return features
```

나이브 베이즈 분류기를 활용한 감성분석

· 자주 등장하는 단어를 통해 feature 생성하기

- 첫 번째 리뷰 데이터의 feature 생성 결과

```
In [88]: with open('aclImdb/train/neg/{}'.format(files[0]), 'r', encoding = 'utf-8') as f:
             review = nltk.word_tokenize(f.read())
         find_features(review)
```

```
Out [88]: {'Bromwell': False,
           'High': False,
           'cartoon': False,
           'comedy': True,
           '.': True,
           'It': False,
           'ran': False,
           'time': True,
           'programs': False,
           'school': False,
           'life': False,
           ',': True,
           '``': False,
           'Teachers': False,
           "''": False,
           'My': False,
           '35': False,
           'years': False,
           'teaching': False,
           'profession': False
```

나이브 베이즈 분류기를 활용한 감성분석

· 각 리뷰의 feature 값들로 이루어진 feature set을 생성하기

- 3000차원의 feature값과 함께 label ('pos' 혹은 'neg')를 튜플로 생성하여 리스트에 첨부한다

```
In [36]: feature_sets = []
         files = os.listdir('aclImdb/train/pos')[:1000]
         for file in files:
             with open('aclImdb/train/pos/{}'.format(file), 'r', encoding = 'utf-8') as f:
                 review = nltk.word_tokenize(f.read())
                 feature_sets.append((find_features(review), 'pos'))
```

```
In [43]: feature_sets[0]

Out[43]: ({'Bromwell': True,
           'High': True,
           'cartoon': True,
           'comedy': True,
           '.': True,
           'It': True,
           'ran': True,
           'time': True,
           'programs': True,
           'school': True,
           'life': True,
           ',': True,
           '``': True,
           'Teachers': True,
           "''": True,
           'My': True,
           '35': True,
           'years': True,
           'teaching': True,
           'profession': True
```

나이브 베이즈 분류기를 활용한 감성분석

· 나이브 베이즈 분류기(Naïve Bayes Classifier)[source: https://ko.wikipedia.org/wiki/%EB%82%98%EC%9D%B4%EB%B8%8C_%EB%B2%A0%EC%9D%B4%EC%A6%88_%EB%B6%84%EB%A5%98]

- 나이브 베이즈 분류(Naïve Bayes classification)은 특성들 사이의 독립을 가정하는 베이즈 정리를 적용한 확률 분류기의 일종으로 1950년대 이후 폭넓게 연구 및 활용되고 있다

- 특히, 나이브 베이즈 분류기는 텍스트를 다양한 범주 중 하나로 분류하는 작업에 흔히 사용된다

- nltk에는 나이브 베이즈 분류기가 내장되어 있으므로 쉽게 사용 및 검증할 수 있다

In [41]:
```
clf = nltk.NaiveBayesClassifier.train(training_set)
```

In [45]:
```
result = nltk.classify.accuracy(clf, test_set)*100
```

나이브 베이즈 분류기를 활용한 감성분석

· 실습 4-2.1. nltk의 Naïve Bayes classifier를 통해 2000개의 영화 리뷰를 긍정 혹은 부정으로 classify 하고 그 결과를 출력해 본다

- 학습 데이터와 검증 데이터는 각각 긍정 리뷰 1000개, 부정 리뷰 1000개로 이루어져야 한다

  ■ 즉, 학습 데이터 2000개로 학습을 하고 검증 데이터 2000개로 그 performance를 검증해 본다

- 학습을 위해서는 nltk.NaiveBayesClassifier.train()을 활용하고 검증을 위해서는 nltk.classify.accuracy()를 활용한다

- 수행 예시

```
In [46]: print('Accuracy of the Naive Bayes classification model: ', result)

Accuracy of the Naive Bayes classification model:  82.0
```

나이브 베이즈 분류기를 활용한 감성분석

· 실습 4-2.2. nltk의 Naïve Bayes classifier를 통해 전체 영화 리뷰를 긍정 혹은 부정으로 classify하고 그 결과를 출력해 본다

- 학습 데이터와 검증 데이터는 각각 긍정 리뷰 12500개, 부정 리뷰 12500개로 이루어져야 한다

  ■ 즉, 학습 데이터 25000개로 학습을 하고 검증 데이터 25000개로 그 performance를 검증해 본다

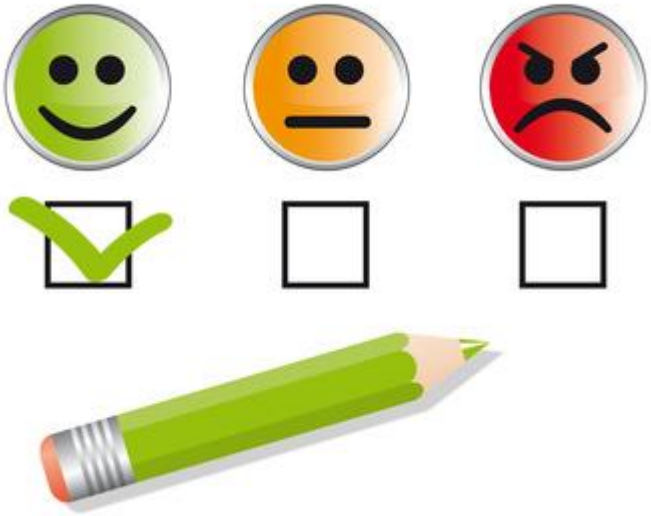
- 학습을 위해서는 nltk.NaiveBayesClassifier.train()을 활용하고 검증을 위해서는 nltk.classify.accuracy()를 활용한다

- 수행 예시

```
In [48]: print('Accuracy of the Naive Bayes classification model: ', result)

Accuracy of the Naive Bayes classification model:  82.76
```

Doc2Vec을 활용한 감성분석



Doc2Vec을 활용한 감성분석

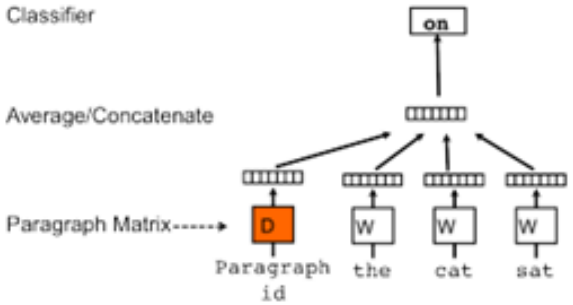
· Doc2Vec: 벡터 공간 모형의 일종으로 간단한 신경망(neural network) 모형으로 만들어지는 모델이다
[source: Le and Mikilov 2014]

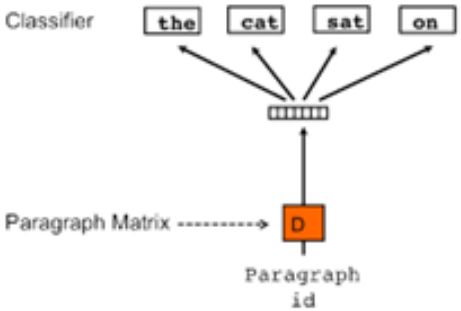
- Word2Vec은 각 단어를 벡터로 표현한다면, Doc2Vec은 문장, 문단, 문서 전체 등 문서의 구성요소를 한정된 차원 내의 벡터로 표현 가능

■ 단어 간의 유사도 뿐만 아니라, 문서의 모든 구성요소 간의 유사성(similarity)을 정의하고 계산할 수 있다

- Word2Vec 및 Doc2Vec 모델의 핵심적인 차이는 특정 구성요소의 “문맥(context)”를 고려할 수 있다는 것이다

■ 이를 통해 앞서 살펴본 감성분석의 많은 한계점(은어, 오타, 부정어 등)을 다소 극복할 수 있다!





Doc2Vec을 활용한 감성분석

· 텍스트 전처리 및 Doc2Vec 모델 학습을 위한 패키지 및 모듈 불러오기

In [74]:
```
import gensim
from gensim.models import Doc2Vec

import numpy as np
from random import shuffle
from sklearn.linear_model import LogisticRegression
```

In [13]:
```
from nltk.corpus import stopwords
from nltk import word_tokenize
from nltk.stem.wordnet import WordNetLemmatizer

import multiprocessing
import os
```

- 텍스트 전처리를 위해 nltk 패키지를, Doc2Vec 모델 생성을 위해 gensim 패키지를 활용한다

Doc2Vec을 활용한 감성분석

· 전처리 및 모델 생성을 위한 객체 생성

- stopwords 집합과 lemmatizer 생성

In [4]:
```
stop_words = set(stopwords.words('english'))
lemm = WordNetLemmatizer()
```

- LabeledSentence 클래스를 오버라이딩한다

In [7]:
```
LabeledSentence = gensim.models.doc2vec.LabeledSentence

class LabeledLineSentence(object):
    def __init__(self, doc_list, labels_list):
       self.labels_list = labels_list
       self.doc_list = doc_list
    def __iter__(self):
        for idx, doc in enumerate(self.doc_list):
            yield LabeledSentence(words=str(doc).split(),tags=[self.labels_list[idx]])
```

■ Doc2Vec 모델에는 각 paragraph의 label(id)를 지정해 주어야 한다 → 개발자가 알아보기 쉬운 임의의 id 등록 가능

Doc2Vec을 활용한 감성분석

· 데이터 전처리

- 학습 데이터 및 레이블 리스트 생성하기

In [37]:
```
review_list = []
labels_list = []
```

In [38]:
```
files = os.listdir('aclImdb/train/pos')[:1000]
for file in files:
    review = ''
    with open('aclImdb/train/pos/{}'.format(file), 'r', encoding = 'utf-8') as f:
        for word in word_tokenize(f.read()):
            if lemm.lemmatize(word) not in stop_words:
                review += ' ' + word
        f.close()
    review_list.append(review)
    labels_list.append('pos_' + file)
```

In [39]:
```
files = os.listdir('aclImdb/train/neg')[:1000]
for file in files:
    review = ''
    with open('aclImdb/train/neg/{}'.format(file), 'r', encoding = 'utf-8') as f:
        for word in word_tokenize(f.read()):
            if lemm.lemmatize(word) not in stop_words:
                review += ' ' + word
        f.close()
    review_list.append(review)
    labels_list.append('neg_' + file)
```

Doc2Vec을 활용한 감성분석

· 데이터 전처리

- 검증 데이터 및 레이블 생성하기

In [40]:
```
files = os.listdir('aclImdb/test/pos')[:1000]
for file in files:
    review = ''
    with open('aclImdb/test/pos/{}'.format(file), 'r', encoding = 'utf-8') as f:
        for word in word_tokenize(f.read()):
            if lemm.lemmatize(word) not in stop_words:
                review += ' ' + word
        f.close()
    review_list.append(review)
    labels_list.append('pos_' + file)
```

In [42]:
```
files = os.listdir('aclImdb/test/neg')[:1000]
for file in files:
    review = ''
    with open('aclImdb/test/neg/{}'.format(file), 'r', encoding = 'utf-8') as f:
        for word in word_tokenize(f.read()):
            if lemm.lemmatize(word) not in stop_words:
                review += ' ' + word
        f.close()
    review_list.append(review)
    labels_list.append('neg_' + file)
```

■ 학습 데이터 및 레이블과 동일한 방법으로 생성한다

Doc2Vec을 활용한 감성분석

· Doc2Vec 모델 생성하기

- 리뷰 리스트와 레이블 리스트를 넣어 LabeledLineSentence 객체를 생성한다

```
In [43]: it = LabeledLineSentence(doc_list = review_list, labels_list = labels_list)
```

- Doc2Vec 모델을 학습한다

```
In [44]: model = Doc2Vec(size = 3000, window = 10, dm = 0, alpha=0.025, min_alpha=0.025, min_count=5, workers = multiprocessing.cpu_count())

         model.build_vocab(it)
         model.train(it, total_examples = 4000, epochs = 20)

         model.save('partial_Doc2Vec.model')
```

■ 파라미터에 대한 설명은 다음을 참고한다: https://radimrehurek.com/gensim/models/doc2vec.html

- 모델 불러오기

```
In [54]: model = Doc2Vec.load('partial_Doc2Vec.model')
```

Doc2Vec을 활용한 감성분석

· 실습 4-3-1. 생성한 Doc2Vec 모델을 활용하여, 2000개의 검증 데이터에 대한 감성분석을 수행한다

- 로지스틱 회귀 분류기(logistic regression classifier)를 활용하여 각 리뷰를 긍정/부정으로 분류한다

- 기계학습 알고리즘의 변수(feature)로는 Doc2Vec의 결과로 생성된 3000차원의 배열(array)를 활용한다

- 수행 예시

```
In [73]: clf.score(x_test, y_test)
Out [73]: 0.91849999999999998
```

Doc2Vec을 활용한 감성분석

· 실습 4-3-2. 생성한 Doc2Vec 모델을 활용하여, 전체 검증 데이터에 대한 감성분석을 수행한다

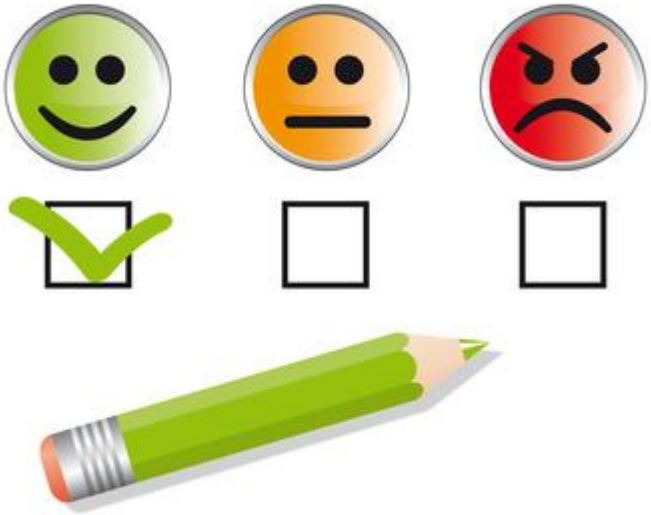
- 로지스틱 회귀 분류기(logistic regression classifier)를 활용하여 각 리뷰를 긍정/부정으로 분류한다

- 기계학습 알고리즘의 변수(feature)로는 Doc2Vec의 결과로 생성된 3000차원의 배열(array)를 활용한다

- 수행 예시

```
In [87]: clf.score(x_test, y_test)
Out [87]: 0.79012000000000004
```

인공신경망 모델을 활용한 감성분석



인공신경망 모델을 활용한 감성분석

· Artificial Neural Networks with Keras[source: Brownlee 2016]

- 인공신경망 모델은 다수의 뉴런(neurons)으로 이루어진 층(layers)들을 쌓아 올린 구조를 뜻한다

■ 뉴런 내부는 input을 받아들여 activation function(sigmoid, relu 등)을 거쳐 output을 내보낸다

■ 은닉층(hidden layer)가 두 개 이상 쌓이면 딥 러닝(deep learning)이라 한다

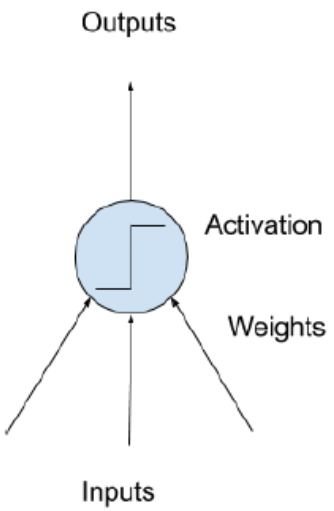


Figure 6.1: Model of a Simple Neuron

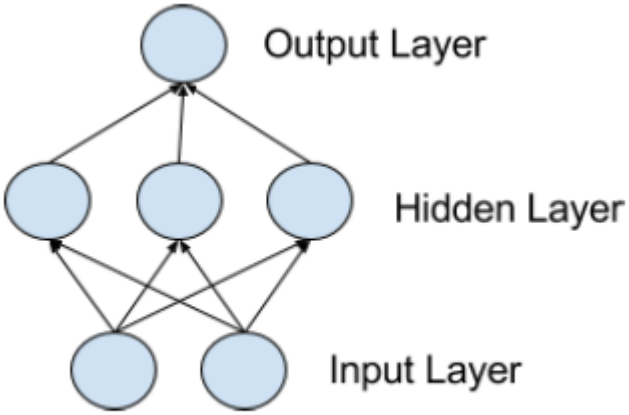


Figure 6.2: Model of a Simple Network

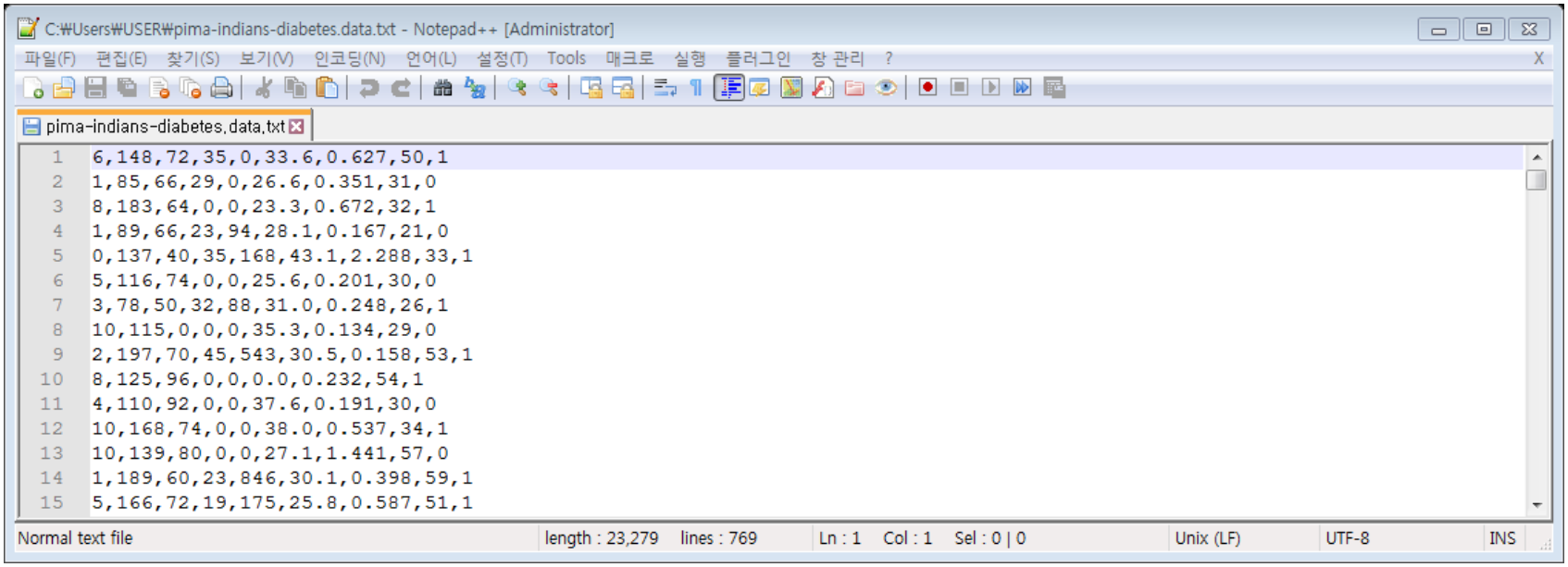
인공신경망 모델을 활용한 감성분석

· Artificial Neural Networks with Keras[source: Brownlee 2016]

- 데이터 가져오기: 아래 URL에서 데이터를 가져와 텍스트 파일(.txt)로 저장한다

■ URL: http://archive.ics.uci.edu/ml/machine-learning-databases/pima-indians-diabetes/pima-indians-diabetes.data

■ 파일 이름은 'pima-indians-diabetes.data.txt'로 지정한다



인공신경망 모델을 활용한 감성분석

· Artificial Neural Networks with Keras[source: Brownlee 2016]

- 인공신경망 모델을 위한 기본적인 모듈과 패키지 불러오기

In [2]:
```
from keras.models import Sequential
from keras.layers import Dense
import numpy as np

np.random.seed(7)
```

■ Sequential(): (종류와 관계없이) 인공신경망 모델을 생성하는 데 쓰임

■ Dense(): 완전 연결된 층(fully connected layers)들을 생성하는 데 쓰임

- 데이터 가져오기 및 슬라이싱

In [5]:
```
data = np.loadtxt('pima-indians-diabetes.data.txt', delimiter = ',')
```

In [7]:
```
X = data[:, 0:-1]
Y = data[:, -1]
```

■ 수치형 데이터만 있으므로 numpy의 loadtxt() 함수를 활용한다

■ 마지막 열이 y 데이터(암 발생 여부)이다

인공신경망 모델을 활용한 감성분석

· Artificial Neural Networks with Keras[source: Brownlee 2016]

- 데이터 크기(shape)

```
In [10]: X.shape
Out[10]: (768, 8)

In [11]: Y.shape
Out[11]: (768,)
```

■ X 데이터는 8개의 변수로 이루어져 있으며, Y 데이터는 768개의 0 혹은 1의 이진 데이터(binary data)이다

- 모델 생성하기

```
In [13]: model = Sequential()
         model.add(Dense(12, input_dim = 8, kernel_initializer = 'uniform', activation = 'sigmoid'))
         model.add(Dense(8, kernel_initializer = 'uniform', activation = 'sigmoid'))
         model.add(Dense(1, kernel_initializer = 'uniform', activation = 'sigmoid'))
```

■ 노드의 크기가 각각 12, 8, 1인 fully connected layer 들을 생성한다

■ kernel initializers: https://keras.io/initializers/

■ activations (activation functions: https://keras.io/activations/

인공신경망 모델을 활용한 감성분석

· Artificial Neural Networks with Keras[source: Brownlee 2016]

- 생성된 인공신경망 구조

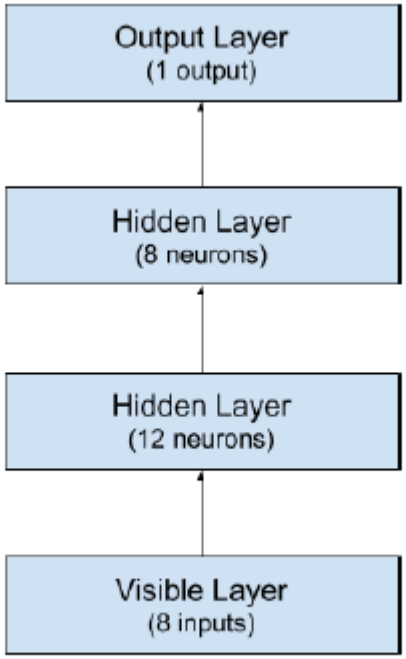


Figure 7.1: Visualization of Neural Network Structure.

인공신경망 모델을 활용한 감성분석

· Artificial Neural Networks with Keras[source: Brownlee 2016]

- 모델 컴파일하기

```
In [14]: model.compile(loss = 'binary_crossentropy', optimizer = 'sgd', metrics = ['accuracy'])
```

- optimizers: https://keras.io/optimizers/
- metrics: https://keras.io/metrics/

- 모델 학습하기

```
In [16]: model.fit(X, Y, epochs = 150, batch_size = 10)
```

```
768/768 [==============================] - 0s - loss: 0.6329 - acc: 0.6510

Epoch 2/150
768/768 [==============================] - 0s - loss: 0.6324 - acc: 0.6510

Epoch 3/150
768/768 [==============================] - 0s - loss: 0.6320 - acc: 0.6510

Epoch 4/150
768/768 [==============================] - 0s - loss: 0.6319 - acc: 0.6510
```

인공신경망 모델을 활용한 감성분석

· Artificial Neural Networks with Keras[source: Brownlee 2016]

- 모델 평가하기

```
In [17]: scores = model.evaluate(X,Y)

         32/768 [>.............................] - ETA: 0s('acc', 69.791666666666657)

In [22]: model.metrics_names[1], scores[1]*100    # printing the accuracy of the prediction

Out[22]: ('acc', 69.791666666666657)
```

■ 모델의 분류 정확도(classification accuracy)를 측정한다

■ 본 모델의 학습 정확도(training accuracy)는 69.79%이다

인공신경망 모델을 활용한 감성분석

· Artificial Neural Networks with Keras[source: Brownlee 2016]

- 전체 코드

```
from keras.models import Sequential
from keras.layers import Dense
import numpy as np

np.random.seed(7)

data = np.loadtxt('pima-indians-diabetes.data.txt', delimiter = ',')

X = data[:, 0:-1]
Y = data[:, -1]

model = Sequential()
model.add(Dense(12, input_dim = 8, kernel_initializer = 'uniform', activation = 'sigmoid'))
model.add(Dense(8, kernel_initializer = 'uniform', activation = 'sigmoid'))
model.add(Dense(1, kernel_initializer = 'uniform', activation = 'sigmoid'))

model.compile(loss = 'binary_crossentropy', optimizer = 'sgd', metrics = ['accuracy'])
model.fit(X, Y, epochs = 150, batch_size = 10)

scores = model.evaluate(X,Y)
print(model.metrics_names[1], scores[1]*100)    # printing the accuracy of the prediction
```

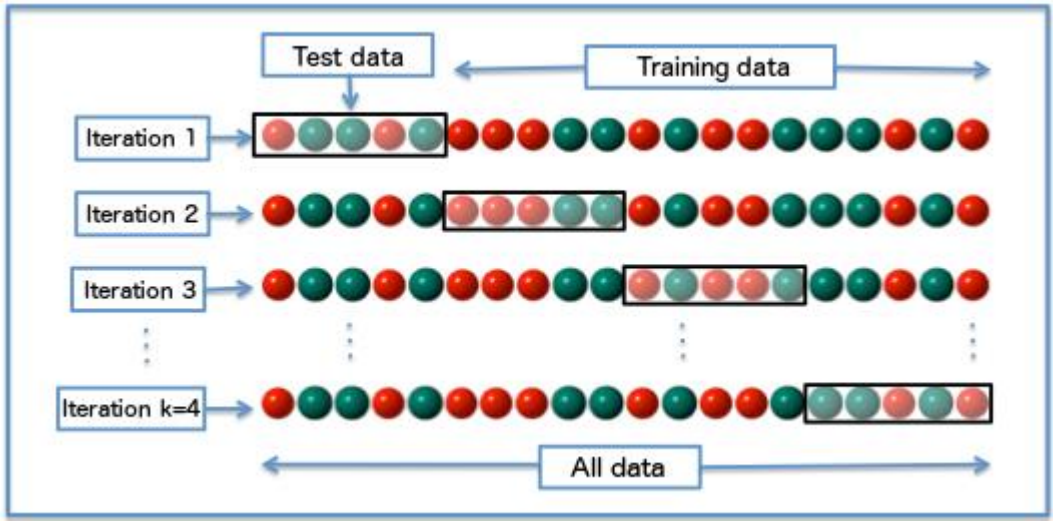
인공신경망 모델을 활용한 감성분석

· k-fold cross-validation[source: Brownlee 2016, https://en.wikipedia.org/wiki/Cross-validation_(statistics)]

- 모델의 확장성(generalizability), 과적합(overfitting) 정도 등을 검증하고 파라미터 조정(parameter estimation)을 위하여 k-fold cross validation을 흔히 사용한다

■ 전체 데이터셋을 k개의 집단으로 나누어 분류기(clasifier)를 k-1개의 집합으로 학습을 시키고, 나머지 1개의 집합으로 검증하여 성능을 측정한다

■ 이러한 과정을 k번 반복하여 발생한 k개의 정확도를 평균하여 분류기의 성능으로 정의한다



인공신경망 모델을 활용한 감성분석

· k-fold cross-validation with Keras & Sklearn [source: Brownlee 2016]

- 모듈과 패키지 불러오기

In [25]:
```
from keras.models import Sequential
from keras.layers import Dense
from keras.wrappers.scikit_learn import KerasClassifier
from sklearn.model_selection import StratifiedKFold
from sklearn.model_selection import cross_val_score
import numpy as np

np.random.seed(7)
```

■ Keras에서는 scikit-learn을 활용하는 케라스 분류기(KerasClassifier) wrapper를 제공한다

■ cross validation with sklearn: http://scikit-learn.org/stable/modules/cross_validation.html

인공신경망 모델을 활용한 감성분석

· k-fold cross-validation with Keras & Sklearn [source: Brownlee 2016]

- 모델 생성 함수 정의하기

In [24]:
```
def create_model():
    model = Sequential()
    model.add((Dense(12, input_dim = 8, kernel_initializer = 'he_normal', activation = 'relu')))
    model.add((Dense(8, kernel_initializer = 'he_normal', activation = 'relu')))
    model.add(Dense(1, kernel_initializer = 'he_normal', activation = 'sigmoid'))

    model.compile(loss = 'binary_crossentropy', optimizer = 'adam', metrics = ['accuracy'])
    return model
```

- 데이터 불러오기

In [26]:
```
data = np.loadtxt('pima-indians-diabetes.data.txt', delimiter = ',')
X = data[:, :-1]
Y  = data[:, -1]
```

인공신경망 모델을 활용한 감성분석

· k-fold cross-validation with Keras & Sklearn [source: Brownlee 2016]

- 모델 생성하기

```
In [28]: model = KerasClassifier(build_fn = create_model, epochs = 150, batch_size = 10, verbose = 0)
```

- k-fold cross validation 수행하기

```
In [30]: kfold = StratifiedKFold(n_splits = 10, shuffle = True, random_state = 7)
```

```
In [31]: results = cross_val_score(model, X, Y, cv = kfold)
```

- 결과

```
In [32]: results
Out[32]: array([ 0.68831169,  0.64935065,  0.76623376,  0.64935065,  0.76623377,
                 0.68831169,  0.64935065,  0.35064936,  0.72368422,  0.68421053])
```

```
In [33]: results.mean()
Out[33]: 0.6615686970739354
```

인공신경망 모델을 활용한 감성분석

· k-fold cross-validation with Keras & Sklearn [source: Brownlee 2016]

- 전체 코드

```
from keras.models import Sequential
from keras.layers import Dense
from keras.wrappers.scikit_learn import KerasClassifier
from sklearn.model_selection import StratifiedKFold
from sklearn.model_selection import cross_val_score
import numpy as np

np.random.seed(7)

def create_model():
    model = Sequential()
    model.add((Dense(12, input_dim = 8, kernel_initializer = 'he_normal', activation = 'relu')))
    model.add((Dense(8, kernel_initializer = 'he_normal', activation = 'relu')))
    model.add(Dense(1, kernel_initializer = 'he_normal', activation = 'sigmoid'))

    model.compile(loss = 'binary_crossentropy', optimizer = 'adam', metrics = ['accuracy'])
    return model

data = np.loadtxt('pima-indians-diabetes.data.txt', delimiter = ',')
X = data[:, :-1]
Y = data[:, -1]

model = KerasClassifier(build_fn = create_model, epochs = 150, batch_size = 10, verbose = 0)
kfold = StratifiedKFold(n_splits = 10, shuffle = True, random_state = 7)
results = cross_val_score(model, X, Y, cv = kfold)

print(results)
print(results.mean())
```

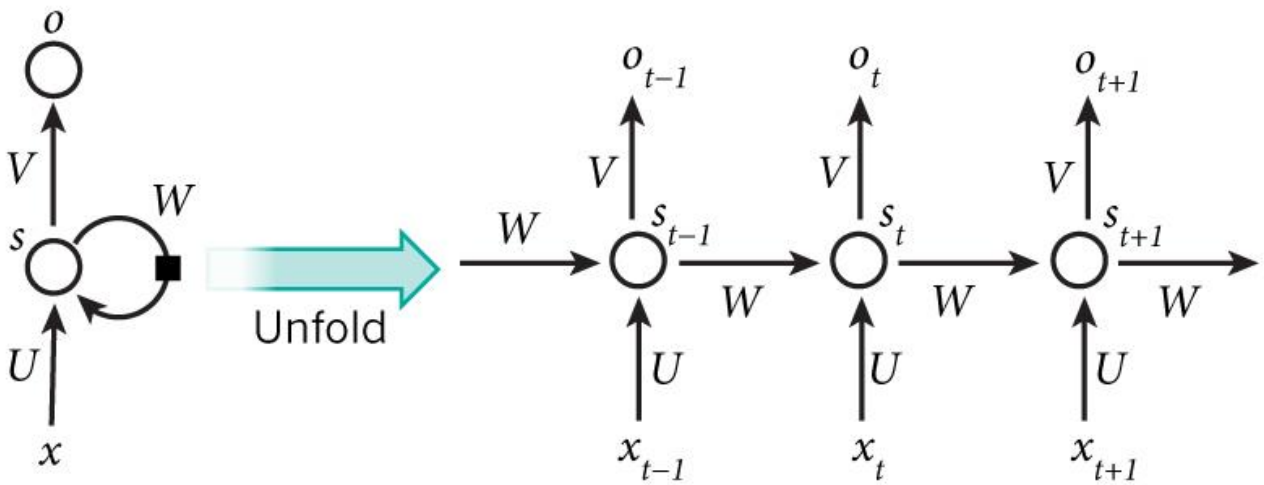
인공신경망 모델을 활용한 감성분석

· LSTM with Keras [source: Brownlee 2016, http://aikorea.org/blog/rnn-tutorial-1/]

- RNN(Recurrent Neural Networks): 다양한 자연어처리(NLP) 문제에 대해 뛰어난 성능을 보이고 있는 모델

■ RNN이 “recurrent”한 이유는 동일한 태스크를 한 시퀀스의 모든 요소마다 적용하고, 출력 결과는 이전의 계산 결과에 영향을 받기 때문

■ 다시 말해, RNN은 현재 계산된 결과에 대한 “메모리” 정보를 담고 있음

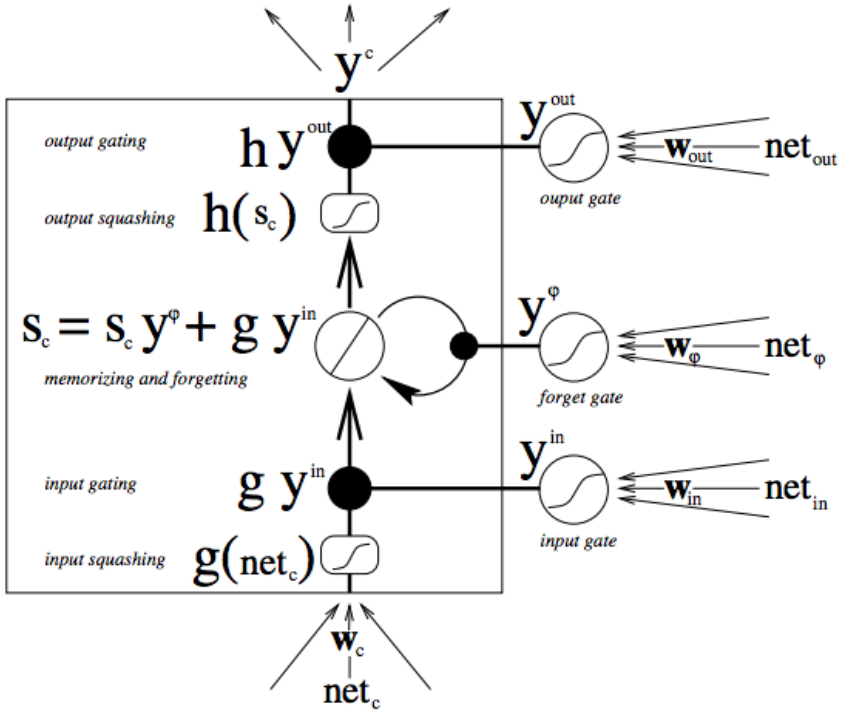


인공신경망 모델을 활용한 감성분석

· LSTM with Keras [source: Brownlee 2016, http://aikorea.org/blog/rnn-tutorial-1/, https://deeplearning4j.org/kr/lstm]

- LSTM: RNN의 변형으로, 오차의 back-propagation이 이전 단계로 잘 흘러갈 수 있도록 도와줌

■ LSTM유닛은 여러 개의 게이트(gate)가 붙어있는 셀(cell)로 이루어져있으며 이 셀의 정보를 새로 저장/셀의 정보를 불러오기/셀의 정보를 유지하는 기능이 있다

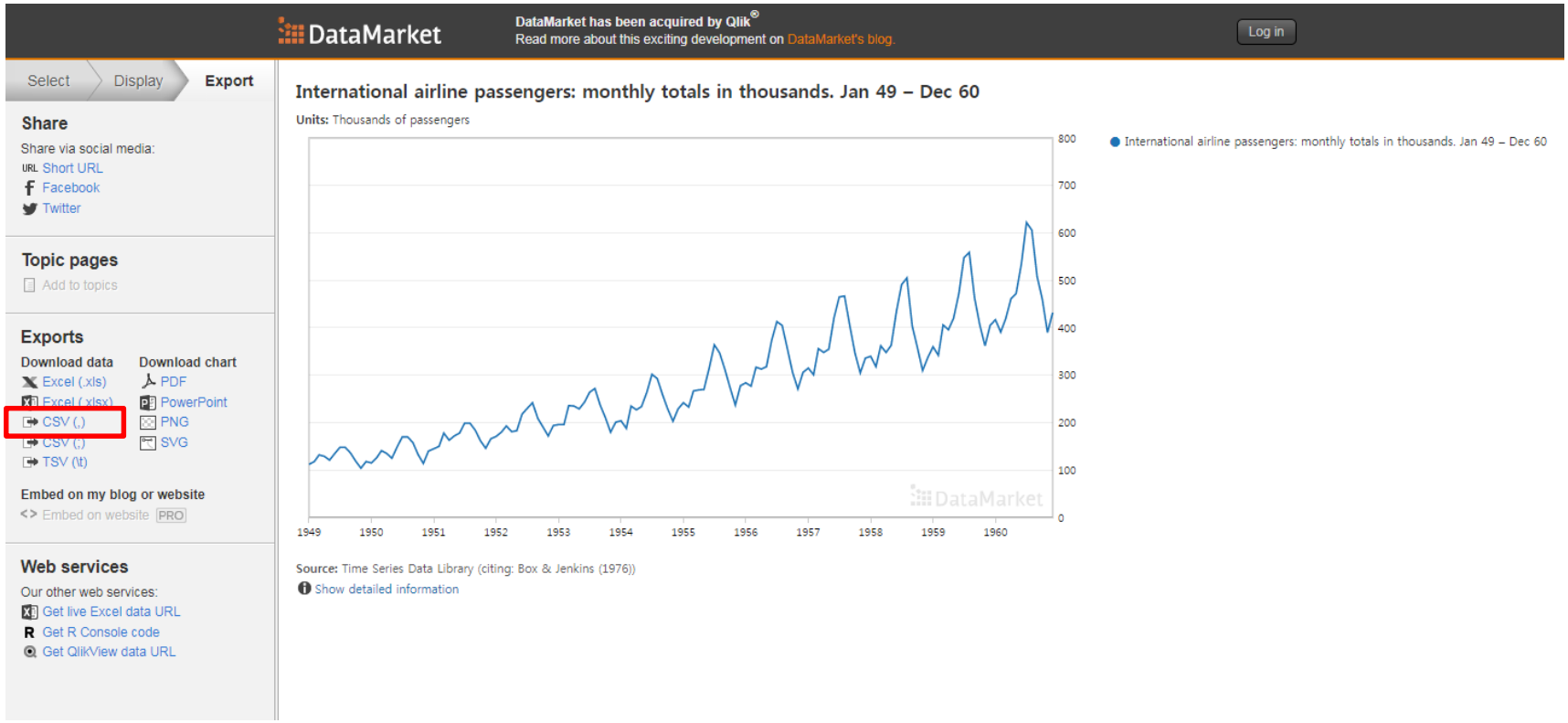


인공신경망 모델을 활용한 감성분석

· Keras를 활용한 LSTM 모델 [source: Brownlee 2016]

- 데이터 가져오기: 아래 URL에서 콤마로 구분된 csv 형식의 데이터를 export해 다운로드한다

■ URL: https://datamarket.com/data/set/22u3/international-airline-passengers-monthly-totals-in-thousands-jan-49-dec-60#!ds=22u3&display=line



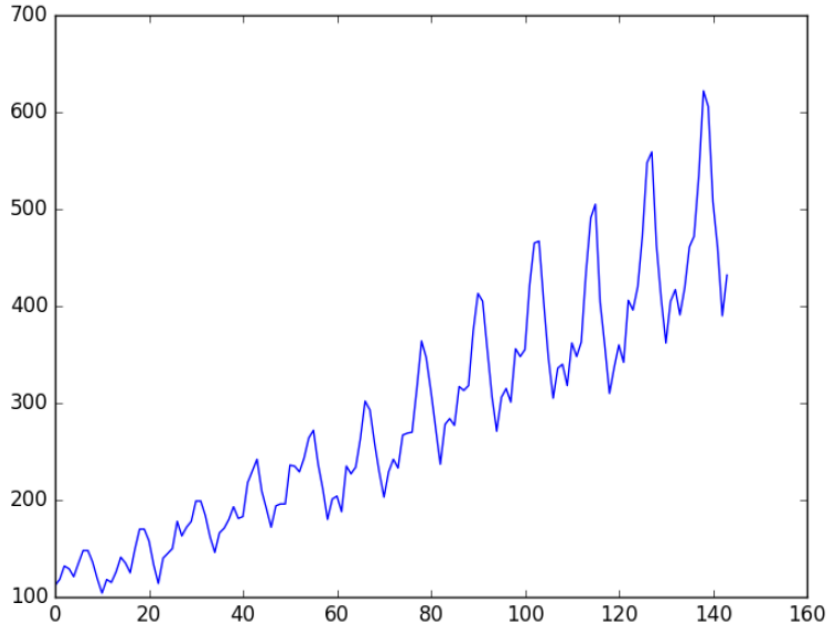
인공신경망 모델을 활용한 감성분석

· Keras를 활용한 LSTM 모델 [source: Brownlee 2016]

- 문제 설명(Problem description): 국제 항공 승객 수 예측하기

■ 데이터셋은 1949년 1월부터 1960년 12월까지 월별 국제 항공 승객 수를 1000단위로 측정한 것

```
"Month","International airline passengers: monthly totals in thousands. Jan 49 ? Dec 60"
"1949-01",112
"1949-02",118
"1949-03",132
"1949-04",129
"1949-05",121
```



인공신경망 모델을 활용한 감성분석

· Keras를 활용한 LSTM 모델 [source: Brownlee 2016]

- 문제 설명(Problem description): 국제 항공 승객 수 예측하기

■ 우리의 목적은 이번 달(t)의 승객 명수를 통해 다음 달(t+1)의 승객 명수를 예측하고자 하는 것

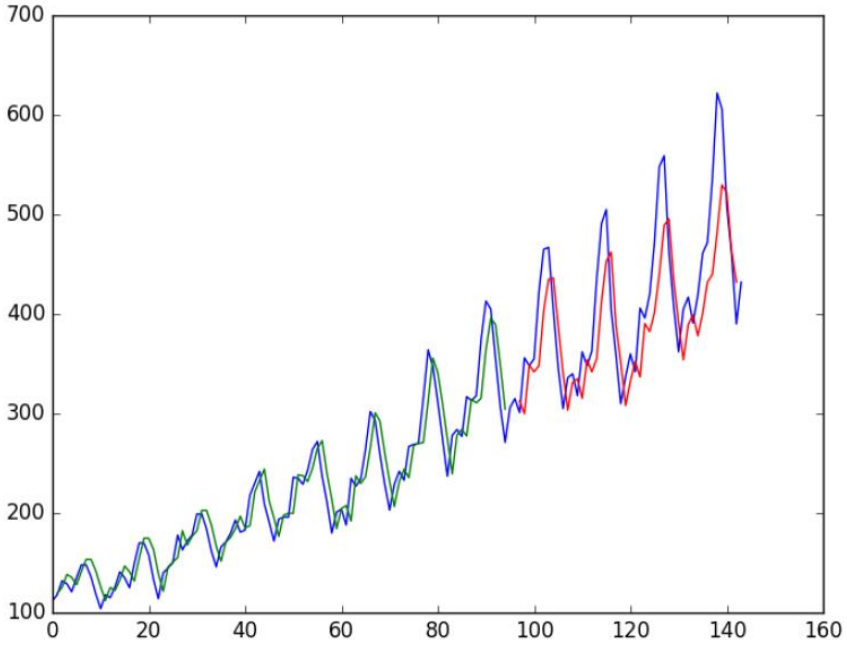
주어진 데이터	예측하고자 하는 값
1949년 1월의 승객 수	1949년 2월의 승객 수
1949년 2월의 승객 수	1949년 3월의 승객 수
1949년 3월의 승객 수	1949년 4월의 승객 수
…	…
1960년 10월의 승객 수	1960년 11월의 승객 수
1960년 11월의 승객 수	1960년 12월의 승객 수
1960년 12월의 승객 수	1961년 1월의 승객 수

인공신경망 모델을 활용한 감성분석

· Keras를 활용한 LSTM 모델 [source: Brownlee 2016]

- 문제 설명(Problem description): 국제 항공 승객 수 예측하기

■ 이를 RNN의 일종인 LSTM을 활용하여 예측을 해보고자 한다



*** Blue: Whole data, Green: Training, Red: Predictions**

인공신경망 모델을 활용한 감성분석

· Keras를 활용한 LSTM 모델 [source: Brownlee 2016]

- 필요한 패키지와 모듈 불러오기

In [36]:
```
import numpy as np
import pandas as pd
import math
from keras.models import Sequential
from keras.layers import Dense
from keras.layers import LSTM
from sklearn.preprocessing import MinMaxScaler
from sklearn.metrics import mean_squared_error

np.random.seed(7)
```

■ LSTM 모델에서는 Dense 레이어 대신 LSTM 레이어를 활용한다

■ sklearn의 MinMaxScaler를 활용해 데이터를 scaling 한다

■ regression 문제이므로 mean_squared_error를 통해 오차를 계산한다

인공신경망 모델을 활용한 감성분석

· Keras를 활용한 LSTM 모델 [source: Brownlee 2016]

- 데이터 불러오기

```
In [50]: df = pd.read_csv('international-airline-passengers.csv', usecols = [1], engine='python', skipfooter=3)

In [52]: data = df.values.astype('float32')
         data.shape

Out[52]: (144, 1)

In [77]: scaler = MinMaxScaler(feature_range = (0,1))
         data = scaler.fit_transform(data)
         data

Out[77]: array([[ 0.01544401],
                [ 0.02702703],
                [ 0.05405405],
                [ 0.04826255],
                [ 0.03281853],
                [ 0.05984557],
                [ 0.08494207],
                [ 0.08494207],
                [ 0.06177607],
                [ 0.02895753]
```

■ 결측치(NaN)를 핸들링하기 위해 pandas 모듈의 read_csv() 함수를 활용한다

■ MinMaxScaler()를 통해 (0,1)의 값으로 조정한다

인공신경망 모델을 활용한 감성분석

· Keras를 활용한 LSTM 모델 [source: Brownlee 2016]

- 학습 데이터와 검증 데이터로 나눈다

In [56]:
```
train_size = int(len(data) * 0.67)
train_data , test_data = data[:train_size, :], data[train_size:, :]
```

■ 학습 데이터와 검증 데이터의 비율을 2:1로 조정한

- 데이터셋을 원하는 형태로 바꾸기 위한 함수를 정의한다

In [37]:
```
def create_dataset(dataset, look_back = 1):
    dataX, dataY = [], []
    for i in range(len(dataset) - look_back -1):
        a = dataset[i:(i+look_back), 0]
        dataX.append(a)
        dataY.append(dataset[i + look_back, 0])
    return np.array(dataX), np.array(dataY)
```

■ look_back 변수를 통해 몇 달 후의 승객 수를 예측하고 싶은지를 설정한다

인공신경망 모델을 활용한 감성분석

· Keras를 활용한 LSTM 모델 [source: Brownlee 2016]

- create_dataset() 함수를 이용해 학습 데이터와 검증 데이터를 생성한다

In [58]:
```
look_back = 1
X_train, Y_train = create_dataset(train_data, look_back)
X_test, Y_test = create_dataset(test_data, look_back)
```

In [60]:
```
X_train.shape
```

Out [60]:
```
(94, 1)
```

In [61]:
```
X_test.shape
```

Out [61]:
```
(46, 1)
```

In [62]:
```
Y_train.shape
```

Out [62]:
```
(94,)
```

In [63]:
```
Y_test.shape
```

Out [63]:
```
(46,)
```

인공신경망 모델을 활용한 감성분석

· Keras를 활용한 LSTM 모델 [source: Brownlee 2016]

- X_train과 X_test 데이터의 shape를 재조정한다

In [78]:
```
X_train = np.reshape(X_train, (X_train.shape[0], 1, X_train.shape[1]))
X_test = np.reshape(X_test, (X_test.shape[0], 1, X_test.shape[1]))
print(X_train.shape)
print(X_test.shape)
```

```
(94, 1, 1)
(46, 1, 1)
```

- 모델 학습하기

In [67]:
```
model = Sequential()
model.add(LSTM(4, input_shape = (None, look_back)))
model.add(Dense(1))
model.compile(loss = 'mean_squared_error', optimizer = 'adam')
model.fit(X_train, Y_train, epochs = 100, batch_size = 1, verbose = 2)
```

인공신경망 모델을 활용한 감성분석

· Keras를 활용한 LSTM 모델 [source: Brownlee 2016]

- 학습 데이터와 검증 데이터로 승객 수를 예측해 본다

In [71]:
```
train_prediction = scaler.inverse_transform(model.predict(X_train))
test_prediction = scaler.inverse_transform(model.predict(X_test))
Y_train = scaler.inverse_transform([Y_train])
Y_test = scaler.inverse_transform([Y_test])
```

■ inverse_transform() 함수를 통해 변수 값의 scale을 되돌린다

- RMSE(root mean squared)로 오차를 계산한다

In [73]:
```
train_score = math.sqrt(mean_squared_error(Y_train[0], train_prediction[:, 0]))
print('Train Score: {} RMSE'.format(train_score))
```

```
Train Score: 23.64529144552407 RMSE
```

In [74]:
```
test_score = math.sqrt(mean_squared_error(Y_test[0], test_prediction[:, 0]))
print('Test Score: {} RMSE'.format(test_score))
```

```
Test Score: 52.926745085080775 RMSE
```

인공신경망 모델을 활용한 감성분석

· Keras를 활용한 LSTM 모델 [source: Brownlee 2016]

- 학습 데이터와 검증 데이터로 승객 수를 예측해 본다

In [71]:
```
train_prediction = scaler.inverse_transform(model.predict(X_train))
test_prediction = scaler.inverse_transform(model.predict(X_test))
Y_train = scaler.inverse_transform([Y_train])
Y_test = scaler.inverse_transform([Y_test])
```

■ inverse_transform() 함수를 통해 변수 값의 scale을 되돌린다

- RMSE(root mean squared)로 오차를 계산한다

In [73]:
```
train_score = math.sqrt(mean_squared_error(Y_train[0], train_prediction[:, 0]))
print('Train Score: {} RMSE'.format(train_score))
```

```
Train Score: 23.64529144552407 RMSE
```

In [74]:
```
test_score = math.sqrt(mean_squared_error(Y_test[0], test_prediction[:, 0]))
print('Test Score: {} RMSE'.format(test_score))
```

```
Test Score: 52.926745085080775 RMSE
```

인공신경망 모델을 활용한 감성분석

· Keras를 활용한 LSTM 모델 [source: Brownlee 2016]

- 전체 코드

```
import numpy as np
import pandas as pd
import math
from keras.models import Sequential
from keras.layers import Dense
from keras.layers import LSTM
from sklearn.preprocessing import MinMaxScaler
from sklearn.metrics import mean_squared_error

np.random.seed(7)

def create_dataset(dataset, look_back = 1):
    dataX, dataY = [], []
    for i in range(len(dataset) - look_back -1):
        a = dataset[i:(i+look_back), 0]
        dataX.append(a)
        dataY.append(dataset[i + look_back, 0])
    return np.array(dataX), np.array(dataY)

df = pd.read_csv('international-airline-passengers.csv', usecols = [1], engine='python', skipfooter=3)

data = df.values.astype('float32')
scaler = MinMaxScaler(feature_range = (0,1))
data = scaler.fit_transform(data)

train_size = int(len(data) * 0.67)
train_data , test_data = data[:train_size, :], data[train_size:, :]

look_back = 1
X_train, Y_train = create_dataset(train_data, look_back)
X_test, Y_test = create_dataset(test_data, look_back)
```

인공신경망 모델을 활용한 감성분석

· Keras를 활용한 LSTM 모델 [source: Brownlee 2016]

```
X_train = np.reshape(X_train, (X_train.shape[0], 1, X_train.shape[1]))
X_test = np.reshape(X_test, (X_test.shape[0], 1, X_test.shape[1]))
#print(X_train.shape)
#print(X_test.shape)

model = Sequential()
model.add(LSTM(4, input_shape = (None, look_back)))
model.add(Dense(1))
model.compile(loss = 'mean_squared_error', optimizer = 'adam')
model.fit(X_train, Y_train, epochs = 100, batch_size = 1, verbose = 2)

train_prediction = scaler.inverse_transform(model.predict(X_train))
test_prediction = scaler.inverse_transform(model.predict(X_test))
Y_train = scaler.inverse_transform([Y_train])
Y_test = scaler.inverse_transform([Y_test])

train_score = math.sqrt(mean_squared_error(Y_train[0], train_prediction[:, 0]))
print('Train Score: {} RMSE'.format(train_score))

test_score = math.sqrt(mean_squared_error(Y_test[0], test_prediction[:, 0]))
print('Test Score: {} RMSE'.format(test_score))
```

인공신경망 모델을 활용한 감성분석

· 실습 4-4-1. LSTM을 활용한 IMDB 영화 리뷰 감성분석 (1)

- 학습 데이터와 검증 데이터를 각각 2000개씩 배정한다

- 검증 데이터에 대한 정확도(accuracy)를 계산해 본다

- Imdb 리뷰 데이터 불러오기

In [117]:
```
top_words = 5000
(X_train, y_train), (X_test, y_test) = imdb.load_data(num_words = top_words)
```

■ Keras에는 imdb 리뷰 데이터가 이미 전처리된 상태로 내장되어 있다

■ 상위 5000개 단어만 활용한다

- 수행 예시

In [110]:
```
print(scores[1])
```

```
0.76
```

인공신경망 모델을 활용한 감성분석

· 실습 4-4-2. LSTM을 활용한 IMDB 영화 리뷰 감성분석 (2)

- 학습 데이터와 검증 데이터 전체를 활용한다

- 검증 데이터에 대한 정확도(accuracy)를 계산해 본다

- Imdb 리뷰 데이터 불러오기

In [117]:
```
top_words = 5000
(X_train, y_train), (X_test, y_test) = imdb.load_data(num_words = top_words)
```

■ Keras에는 imdb 리뷰 데이터가 이미 전처리된 상태로 내장되어 있다

■ 상위 5000개 단어만 활용한다

- 수행 예시

In [122]:
```
print(scores[1])
```

```
0.80944
```

Wrap-up

· 감성분석 결과 정리

- Accuracy of each model

모델	n=2000	n=25000
SentiWordNet	71%	68%
Naïve Bayes	82%	82.76%
Doc2Vec	91.8%	79%
LSTM	76%	80.9%

- 결론

■ “There is no free lunch” – 각 모델마다 장단점이 있으므로, 컨텍스트나 데이터에 따라 performance가 천차만별로 달라짐

■ Doc2Vec이나 LSTM은 prototypical한 모델만 세워봄 => 보다 복잡한 모델로 fine-tuning 가능

■ Constant & Iterative experiments make your model better!

References

· Baccianella, S., Esuli, A., & Sebastiani, F. (2010, May). SentiWordNet 3.0: An Enhanced Lexical Resource for Sentiment Analysis and Opinion Mining. In LREC (Vol. 10, pp. 2200-2204).

· Le, Q., & Mikolov, T. (2014). Distributed representations of sentences and documents. In Proceedings of the 31st International Conference on Machine Learning (ICML-14) (pp. 1188-1196).

· Brownlee, J. (2016). Deep Learning with Python. Machine Learning Mastery (https://machinelearningmastery.com/deep-learning-with-python/)