Data Analytics Week 10 Assignment

202011431 산업공학과 차승현

Topic Modeling

Analysis Procedure

1) Data 및 Module Import

In [53]:
```
import gensim
from gensim import corpora
from gensim import models
from gensim.models import CoherenceModel
import matplotlib.pyplot as plt
```

1. Data import

In [40]:
```
file = open('data_week10.txt', 'r')
data_lst = []

while True:
    line = file.readline()
    if not line:
        break
    data_lst.append(line.strip())

file.close()
```

In [41]:
```
data_lst
```

Out[41]:
```
['cute kitty',
 'eat rice cake',
 'kitty hamster',
 'eat bread',
 'rice bread cake',
 'cute hamster eat bread cake']
```

Data_lst 라는 리스트 안에 6개의 문장이 성공적으로 담겨있다.

2) 토큰화 및 벡터화 (Dictionary 생선 및 말뭉치 생성)

2. 토큰화 및 벡터화 (사전생성 및 말뭉치 생성)

```
In [43]: texts = [[word for word in document.split()]
                  for document in data_lst]

         dictionary = corpora.Dictionary(texts)                    # 사전 생성 (토큰화)
         print(dictionary)

         Dictionary(7 unique tokens: ['cute', 'kitty', 'cake', 'eat', 'rice']...)

         7 unique tokesn: ['cute', 'kitty', 'cake', 'eat', 'rice', 'bread', 'hamster']

In [45]: corpus = [dictionary.doc2bow(text) for text in texts]     # 말뭉치 생성 (벡터화)
         print('corpus : {}'.format(corpus))

         corpus : [[(0, 1), (1, 1)], [(2, 1), (3, 1), (4, 1)], [(1, 1), (5, 1)], [(3, 1), (6,
         1)], [(2, 1), (4, 1), (6, 1)], [(0, 1), (2, 1), (3, 1), (5, 1), (6, 1)]]
```

Dictionary에 unique한 7개의 token이 생성되었다.
Dictionary: [‘cute’, ‘kitty’, ‘rice, ‘eat’, ‘cake, ‘hamster, ’bread’]

해당 Dictionary를 벡터화하여 말뭉치를 생성하였다.

말뭉치		(0,1)	cute
(1,1)	kitty	(2,1)	rice
(3,1)	eat	(4,1)	cake
(5,1)	hamster	(6, 1)	bread

3) 토픽 개수 결정

3-1) Perplexity

Perplexity는 사전적 의미로는 "당혹, 혼란, 곤혹" 등의 의미이다. 정보학에서 쓰일 때는 혼란도(혼란한 정도)의 의미로 쓰인다. 이 수치는 특정 확률모델이 실제로 관측되는 값을 얼마나 잘 예측하는지를 평가할 때 사용한다. 토픽 모델링도 문헌 집합 내 용어 출현횟수를 바탕으로, 문헌 내 주제 출현 확률과, 주제 내 용어 출현 확률을 계산하는 확률모델이므로, 확률모델을 평가할

때 사용하는 척도를 사용하는 것은 자연스러운 일이다. 그래서 토픽 모델링을 평가하는데에 있어서 Perplexity는 오랫동안 널리 사용되었다.

$$Perp = 2^{H(p)}$$ => perplexity의 일반적 정의

여기서 H(p)는 p의 정보 엔트로피 값이다. 토픽 모델링 과정에서는 흔히 다음과 같은 식으로 바꿔서 계산한다.

$$Perp = 2^{-\frac{\sum_w LL(w)}{N}}$$

LL(w)은 로그 가능도로써, 토픽 모델 내에서 특정 단어가 해당 주제로 부여될 확률값에 로그를 취한값이고, 그 값들을 모두 합쳐서 전체 단어의 개수인 N으로 나누었으니, 로그 가능도의 평균에 -를 붙이고 지수화한것이다.

대게 깁스샘플링 과정에서 반복횟수가 증가할수록 Perplexity는 감소하는 경향을 보인다. 그러다가 특정 시점을 지나면 더 이상 Perplexity는 감소하지 않고 증가, 감소를 반복하며 요동치는 지점이 등장하는데, 이때를 해당 깁스샘플링의 수렴 지점으로 보고 샘플링을 멈추는 경우가 많다. 그리고 이때의 Perplexity가 해당 모델의 최종 Perplexity가 된다. 이 값이 작으면 작을수록 해당 토픽 모델은 실제 문헌 결과를 잘 반영한다는 뜻이므로 학습이 잘 되었다고 평가를 할 수 있다. 이 값은 LDA 등에서 적절한 주제 개수를 정하기 어려울 때 유용하게 쓰인다. 다양한 주제 개수로 학습을 진행해보면 주제 개수가 몇 개일 때 Perplexity가 최소가 되는지 알 수 있기 때문이다.

문제는 Perplexity 값이 낮다는게 학습이 잘되었다는거지, 그 결과가 사람이 해석하기에 좋다는 것이 아니다. Chang의 논문에 따르면 낮은 Perplexity 값이 늘 해석에 적절한 결과를 보이지는 않는다고 한다. 따라서 사람이 해석하기에 적합한지를 확인하기 위해 다른 척도가 필요했는데, 이렇게 해서 제시된 척도가 Topic Coherence이다.

3-2) Topic Coherence

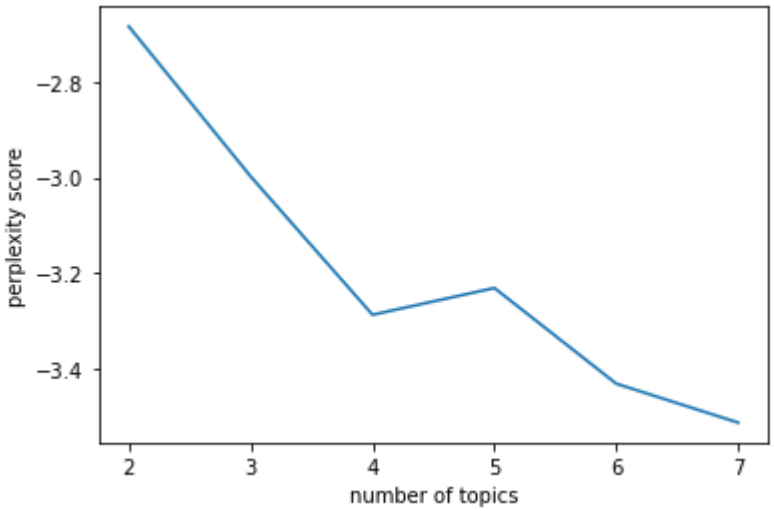
Topic Coherence는 실제로 사람이 해석하기에 적합한 평가 척도를 만들기 위해서 제시된 여러 척도들 중 D Newman에 의해 2010년에 제시된 것이다. Newman과 동료 연구진들은 이를 위해 뉴스와 책 데이터를 수집하여 토픽 모델링을 실시하고, 코더들을 통해 그 결과로 나온 토픽들이 유의미한지 수작업으로 점수를 매기게 했다. 그리고 이렇게 매겨진 점수와 가장 유사한 결과를 낼 수 있는 척도를 찾았다. 그의 주제 평가방법은 다음과 같다.

토픽 모델링 결과로 나온 주제들에 대해 각각의 주제에서 상위 N개의 단어를 뽑고, 모델링이 잘 되어있을수록 한 주제 안에는 의미론적으로 유사한 단어가 많이 모여있게 마련이다. 따라서 상위 단어 간의 유사도를 계산하면 실제로 해당 주제가 의미론적으로 일치하는 단어들끼리 모여있는지 알 수 있다.

	Perplexity	Topic Coherence
의미	확률 모델이 얼마나 정확하게 예측하는지를 의미. 낮을수록 정확한 예측이다.	토픽이 얼마나 의미론적으로 일관적인지를 의미. 높을수록 의미론적 일관성이 높다.
주용도	토픽 모델링 기법이 얼마나 빠르게 수렴하는지 확인할 때, 확률 모델이 다른 모델에 비하여 얼마나 개선되었는지 평가할 때, 동일 모델 내에서 파라미터에 따른 성능을 평가할 때	해당 모델이 실제로 얼마나 의미있는 결과를 내는지 확인하기 위하여 사용
한계	**Perplexity가 낮다고 해서, 결과가 해석이 용이하다는 의미가 아니다.**	**평가를 진행하기 위해서 다른 외부 데이터(Corpus, 시소러스 등)가 필요하다.**

In [50]:
```
perplexity_values = []
for i in range(2, 8):
    ldamodel = gensim.models.ldamodel.LdaModel(corpus, num_topics = i,
                                               id2word = dictionary)
    perplexity_values.append(ldamodel.log_perplexity(corpus))
```

In [51]:
```
x = range(2, 8)
plt.plot(x, perplexity_values)
plt.xlabel('number of topics')
plt.ylabel('perplexity score')
plt.show()
```



Perplexity 결과값

값이 음수로 좋지 못한 결과를 보이므로, Topic Coherence기법을 사용하기로 결정했다.

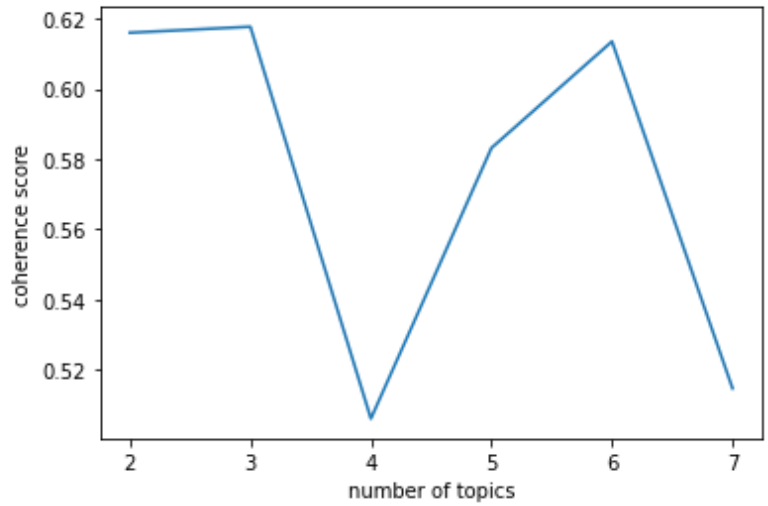
In [61]:
```
coherence_values = []
for i in range(2, 8):
    ldamodel = gensim.models.ldamodel.LdaModel(corpus, num_topics = i,
                                               id2word = dictionary)

    coherence_model_lda = CoherenceModel(model=ldamodel, texts=texts,
                                         dictionary = dictionary, topn=3)

    coherence_lda = coherence_model_lda.get_coherence()

    coherence_values.append(coherence_lda)
```

In [62]:
```
x = range(2, 8)
plt.plot(x, coherence_values)
plt.xlabel('number of topics')
plt.ylabel('coherence score')
plt.show()
```



Coherence 결과값

In [74]:
```
for m, cv in zip(x, coherence_values):
    print("Num Topics =", m, " has Coherence Value of", round(cv, 4))
```

```
Num Topics = 2  has Coherence Value of 0.616
Num Topics = 3  has Coherence Value of 0.6177
Num Topics = 4  has Coherence Value of 0.506
Num Topics = 5  has Coherence Value of 0.5832
Num Topics = 6  has Coherence Value of 0.6135
Num Topics = 7  has Coherence Value of 0.5146
```

Topic의 개수가 3일 때, cv(Coherence value)가 0.6177로 가장 높았다.
따라서, Optimal topic 개수는 3개로 결정하였다.

4) Topic Modeling 시행

```
In [86]: lda = models.ldamodel.LdaModel(corpus=corpus, id2word=dictionary,
                num_topics=3)                    # 모델구축

         for t in lda.show_topics(num_words = 3):  # 주제마다 출현 확률이 높은 단어 순으로 출
             print(t)
```

```
(0, '0.157*"eat" + 0.148*"bread" + 0.146*"rice"')
(1, '0.201*"cute" + 0.201*"hamster" + 0.195*"kitty"')
(2, '0.224*"cake" + 0.219*"rice" + 0.219*"eat"')
```

TOPIC 1: 0.157*”eat” + 0.148*”bread” + 0.146*”rice”

TOPIC 2: 0.201*”cute” + 0.201*”hamster” + 0.195*”kitty”

TOPIC 3: 0.224*”cake” + 0.219*”rice” + 0.219*”eat”

TOPIC	LABELING
1	탄수화물
2	귀여운 동물
3	떡