자연어 처리 (NLP)

4기 박의연

Growth Hackers

CONTENTS

Growth Hackers

자연어, 자연어 처리

- 자연어 (Natural Language)
  - 사람들이 일상적으로 쓰는 언어를 인공적으로 만들어진 인공어(e. 프로그래밍 언어)와 구분하여 부르는 개념

- 자연어 처리(Natural Language Processing)
  - 텍스트에서 의미 있는 정보를 분석, 추출하고 이해하는 일련의 기술 집합
  - 인공언어와 인간 언어의 소통을 위해 자연어를 다루는 과정.

텍스트 전처리 과정

1. HTML 태그 제거 (있을 경우)
2. 특수 문자 제거
3. Stopwords 처리
4. Stemming or Lemmatizing

텍스트 전처리 (1) HTML 태그 제거

- 웹에서 추출한 데이터의 경우 HTML 태그가 남아있는 경우가 있음

- HTML 태그 (ex. <script src='..'>, </script>, <a href='..'>, </br> 들은 처리 및 분석의 대상이 아니므로 제거해야 함.

  → (1) 정규표현식을 활용해 태그 제거 → import re

  → (2) BeautifulSoup 활용해 태그 제거 → import BeautifulSoup

텍스트 전처리 (1) HTML 태그 제거

- BeautifulSoup 활용

In [24]:
```
from bs4 import BeautifulSoup

example1 = BeautifulSoup(train['review'][0], "html5lib")
print(train['review'][0][:700])
print("==========goes into==========")
print(example1.get_text()[:700])
```

```
"With all this stuff going down at the moment with MJ i've started listening
to his music, watching the odd documentary here and there, watched The Wiz an
d watched Moonwalker again. Maybe i just want to get a certain insight into t
his guy who i thought was really cool in the eighties just to maybe make up m
y mind whether he is guilty or innocent. Moonwalker is part biography, part f
eature film which i remember going to see at the cinema when it was originall
y released. Some of it has subtle messages about MJ's feeling towards the pre
ss and also the obvious message of drugs are bad m'kay.<br /><br />Visually i
mpressive but of course this is all about Michael Jackson so unless you remot
ely lik
==========goes into==========
"With all this stuff going down at the moment with MJ i've started listening
to his music, watching the odd documentary here and there, watched The Wiz an
d watched Moonwalker again. Maybe i just want to get a certain insight into t
his guy who i thought was really cool in the eighties just to maybe make up m
y mind whether he is guilty or innocent. Moonwalker is part biography, part f
eature film which i remember going to see at the cinema when it was originall
y released. Some of it has subtle messages about MJ's feeling towards the pre
ss and also the obvious message of drugs are bad m'kay.Visually impressive bu
t of course this is all about Michael Jackson so unless you remotely like MJ
in anyw
```

- 정규표현식 활용

In [20]:
```
import re
text = re.sub('<.+?>', '', train['review'][0][:700], 0, re.I|re.S)
print (train['review'][0][:700])
print("==========goes into==========")
print(text)
```

```
"With all this stuff going down at the moment with MJ i've started listening
to his music, watching the odd documentary here and there, watched The Wiz an
d watched Moonwalker again. Maybe i just want to get a certain insight into t
his guy who i thought was really cool in the eighties just to maybe make up m
y mind whether he is guilty or innocent. Moonwalker is part biography, part f
eature film which i remember going to see at the cinema when it was originall
y released. Some of it has subtle messages about MJ's feeling towards the pre
ss and also the obvious message of drugs are bad m'kay.<br /><br />Visually i
mpressive but of course this is all about Michael Jackson so unless you remot
ely lik
==========goes into==========
"With all this stuff going down at the moment with MJ i've started listening
to his music, watching the odd documentary here and there, watched The Wiz an
d watched Moonwalker again. Maybe i just want to get a certain insight into t
his guy who i thought was really cool in the eighties just to maybe make up m
y mind whether he is guilty or innocent. Moonwalker is part biography, part f
eature film which i remember going to see at the cinema when it was originall
y released. Some of it has subtle messages about MJ's feeling towards the pre
ss and also the obvious message of drugs are bad m'kay.Visually impressive bu
t of course this is all about Michael Jackson so unless you remotely lik
```


Growth Hackers

텍스트 전처리 (2) 특수 문자 제거

- 1) 분석의 대상이 아닌 구두점 (“ ‘ ! , . ? ₩n) 및 특수문자 제거
- 2) 모든 대문자를 소문자로 변환(영어)

  → 정규표현식 활용

  ex) re.sub(‘[^a-zA-Z]’, ’ ‘, text)

텍스트 전처리 (3) Stopwords 처리

- Stopwords (불용어): 텍스트 분석에 있어서 의미를 갖지 않는 문법 및 기타 목적을 가진 단어들

  → 전처리 단계에서 제거해야 함

  1) Stopword들을 지정해 리스트로 정의하고, 단어 토큰들에서 stopword 리스트 원소들을 제거

  2) NLTK 내 Stopwords 프리셋을 리스트로 불러와 append 및 drop하여 활용 가능

텍스트 전처리 (3) Stopwords 처리

In [71]:
```
import nltk
from nltk.corpus import stopwords
stopwords.words('english')[:10]
```

Out[71]:
```
['i', 'me', 'my', 'myself', 'we', 'our', 'ours', 'ourselves', 'you', "you're"]
```

In [72]:
```
stopword_list=stopwords.words('english')
```

In [73]:
```
len(stopword_list)
```

Out[73]:
```
179
```

In [74]:
```
stopword_list.append('label')
```

In [75]:
```
# stopwords 를 제거한 토큰들
print(len(words),end='')
print("개의 단어에서"),
words = [w for w in words if not w in stopword_list]
print(len(words),end='')
print("개로 줄였습니다")
words[:20]
```

```
1094119개의 단어에서
536011개로 줄였습니다
```

Out[75]:
```
['great',
 'cd',
 'lovely',
 'pat',
 'one',
 'great',
 'voices',
 'generation',
 'listened',
 'cd',
 'years',
 'still',
```


Growth Hackers

텍스트 전처리 (4) Stemming

Stemming (어간 추출)

: 어간(stem)을 직접 추출하는 과정. 모든 단어에 대하여 정확한 어간 추출은 불가능하므로, 일반적인 규칙에 따라 단어의 어미를 자르는 작업을 수행함.

- NLTK 내에 세가지 Stemmer가 내장되어 있음
  (Porter, Lancaster, Snowball)

- 같은 어간을 가졌다고 항상 같은 의미를 갖는 것은 아님
  ex. organ ➔ organ / organization ➔ organ

텍스트 전처리 (4) Lemmatizing

Lemmatzing (표제어 추출)
: 단어들로부터 표제어를 찾는 행위

- 명사형 어미 등은 잘 보존하려고 함
  ex. Starting → starting
    cf) starting → start (stemming)
- 동사, 동명사를 같은 의미로 간주하지 않고 구분하는 한계가 있음
  ex. Starting과 started는 '시작'의 의미를 갖지만, 다른 단어로 인식하게 됨.

텍스트 전처리 (4) Stemming vs Lemmatizing

	Porter Stemmer	Lancaster Stemmer	Snowball Stemmer	Lemmatizer	Stemmed & Lemmatized
elder	elder	eld	elder	elder	elder
brothers	brother	broth	brother	brother	brother
one	one	on	one	one	one
lieutenant	lieuten	lieut	lieuten	lieutenant	lieuten
colonel	colonel	colonel	colonel	colonel	colonel
english	english	engl	english	english	english
regiment	regiment	regy	regiment	regiment	regiment
foot	foot	foot	foot	foot	foot
flanders	flander	fland	flander	flanders	flander
formerly	formerli	form	former	formerly	former
commanded	command	command	command	commanded	command
famous	famou	fam	famous	famous	famous

Growth Hackers

자연어 처리 패키지 - 영어 - NLTK

- 영어 텍스트의 경우 NLTK라는 패키지 사용
- NLTK: 교육용으로 개발된 자연어 처리 및 문서 분석용 파이썬 패키지
- 토큰생성, 형태소 분석, 품사 tagging 등의 기능

자연어 처리 패키지 - 한글 -KoNLPy

- 한글 텍스트의 경우KoNLPy라는 패키지 활용
- KoNLPy: 한국어 정보처리를 위한 파이썬 패키지
- 내부에 꼬꼬마(Kkma), 한나눔(Hannanum), 트위터(Twitter) 등의 한글 자연어 처리 패키지가 내장 되어있음
- 한글 문서를 토큰화하거나, 품사를 태깅할 때 활용함.

자연어 처리 패키지 실습 (NLTK - 0. 데이터 로드)

- Session11 자연어처리 - 2. 패키지 실습.ipynb 참조

```
#0. NLTK import & loading the data
import nltk
import re
```

```
nltk.download('gutenberg')
```

```
[nltk_data] Downloading package gutenberg to
[nltk_data]     C:₩Users₩EdwinPark₩AppData₩Roaming₩nltk_data...
[nltk_data]   Package gutenberg is already up-to-date!

True
```

```
from nltk.corpus import gutenberg
files_en = gutenberg.fileids()      # Get file ids
files_en
```

```
['austen-emma.txt',
 'austen-persuasion.txt',
 'austen-sense.txt',
 'bible-kjv.txt',
 'blake-poems.txt',
 'bryant-stories.txt',
 'burgess-busterbrown.txt',
 'carroll-alice.txt',
 'chesterton-ball.txt',
 'chesterton-brown.txt',
 'chesterton-thursday.txt',
 'edgeworth-parents.txt',
 'melville-moby_dick.txt',
 'milton-paradise.txt',
 'shakespeare-caesar.txt',
 'shakespeare-hamlet.txt',
 'shakespeare-macbeth.txt',
 'whitman-leaves.txt']
```

```
doc_en = gutenberg.open('austen-emma.txt').read()
doc_en
```

```
'[Emma by Jane Austen 1816]₩n₩nVOLUME I₩n₩nCHAPTER I₩n₩n₩nEmma Woodhouse, handsome, clever, and rich, with a comfortable home₩nand happy disposition, seemed to unite some of the best blessings₩nof existence; and had lived nearly twenty-one years in the world₩nwith very little to distress or vex her.₩n₩nShe was the youngest of the two daughters of a most affectionate,₩nindulgent father; and had, in consequence of her sister₩'s marriage,₩nbeen mistress of his house from a very early period.  Her mother₩nhad died too long ago for her to have more than an indistinct₩nremembrance of her caresses; and her place had been supplied₩nby an excellent woman as governess, who had fallen little short₩nof a mother in affection.₩n₩nSixteen years had Miss Taylor been in Mr. Woodhouse₩'s family,₩nless as a governess than a friend, very fond of both daughters,₩nbut particularly of Emma.  Between _them_ it was more the intimacy₩nof sisters.  Even before Miss Taylor had ceased to hold the nominal₩noffice of governess, the mildness of her temper had hardly allowed₩nher to impose any restraint; and the shadow of authority being₩nnow long passed away, they had been living together as friend and₩nfriend very mutually attached, and Emma doing just what she liked;₩nhighly esteeming Miss Taylor₩'s judgment, but directed chiefly by₩nher own.₩n₩nThe real evil
```


Growth Hackers

자연어 처리 패키지 실습 (NLTK - 1. tokenize)

- (1) word_tokenize : 불러온 텍스트를 단어 단위로 쪼개자

```
In [6]: tokens_en = nltk.word_tokenize(doc_en) #token단위로 쪼개고 리스트로 객체화
        print(tokens_en)
```

```
['[', 'Emma', 'by', 'Jane', 'Austen', '1816', ']', 'VOLUME', 'I', 'CHAPTE
R', 'I', 'Emma', 'Woodhouse', ',', 'handsome', ',', 'clever', ',', 'and',
'rich', ',', 'with', 'a', 'comfortable', 'home', 'and', 'happy', 'dispositi
on', ',', 'seemed', 'to', 'unite', 'some', 'of', 'the', 'best', 'blessing
s', 'of', 'existence', ';', 'and', 'had', 'lived', 'nearly', 'twenty-one',
'years', 'in', 'the', 'world', 'with', 'very', 'little', 'to', 'distress',
'or', 'vex', 'her', '.', 'She', 'was', 'the', 'youngest', 'of', 'the', 'tw
o', 'daughters', 'of', 'a', 'most', 'affectionate', ',', 'indulgent', 'fath
er', ';', 'and', 'had', ',', 'in', 'consequence', 'of', 'her', 'sister',
"'s", 'marriage', ',', 'been', 'mistress', 'of', 'his', 'house', 'from',
'a', 'very', 'early', 'period', '.', 'Her', 'mother', 'had', 'died', 'too',
'long', 'ago', 'for', 'her', 'to', 'have', 'more', 'than', 'an', 'indistinc
t', 'remembrance', 'of', 'her', 'caresses', ';', 'and', 'her', 'place', 'ha
d', 'been', 'supplied', 'by', 'an', 'excellent', 'woman', 'as', 'governes
s', ',', 'who', 'had', 'fallen', 'little', 'short', 'of', 'a', 'mother', 'i
n', 'affection', '.', 'Sixteen', 'years', 'had', 'Miss', 'Taylor', 'been',
'in', 'Mr.', 'Woodhouse', "'s", 'family', ',', 'less', 'as', 'a', 'governes
s', 'than', 'a', 'friend', ',', 'very', 'fond', 'of', 'both', 'daughters',
',', 'but', 'particularly', 'of', 'Emma', '.', 'Between', '_them_', 'it',
'was', 'more', 'the', 'intimacy', 'of', 'sisters', ',', 'Even', 'before',
```

```
In [7]: en = nltk.Text(tokens_en)
        type(en)
```

```
Out[7]: nltk.text.Text
```

- nltk.word_tokenize(~~)를 통해 모든 단어와 문자를 토큰화할 수 있음(특수문자, 구두점 포함)
- (1) nltk.word_tokenize를 통해 토큰화 한 것을 리스트로 불러올 수 있고,
  (2) nltk.Text(~~)를 통해 nltk.Text라는 개체로 정의해 nltk에 내장된 다양한 기능을 사용할 수 있음

자연어 처리 패키지 실습 (NLTK - 1. tokenize)

- (1) sent_tokenize : 불러온 텍스트를 문장 단위로 쪼개자

```
In [57]: no_n_doc = re.sub("₩n", ' ', doc_en)
sentences=nltk.sent_tokenize(no_n_doc)
len(sentences)
sentences[0:5]

Out[57]: ['[Emma by Jane Austen 1816]  VOLUME I  CHAPTER I  Emma Woodhouse, handsome,
clever, and rich, with a comfortable home and happy disposition, seemed to un
ite some of the best blessings of existence; and had lived nearly twenty-one
years in the world with very little to distress or vex her.',
 "She was the youngest of the two daughters of a most affectionate, indulgent
father; and had, in consequence of her sister's marriage, been mistress of hi
s house from a very early period.",
 'Her mother had died too long ago for her to have more than an indistinct re
membrance of her caresses; and her place had been supplied by an excellent wo
man as governess, who had fallen little short of a mother in affection.',
 "Sixteen years had Miss Taylor been in Mr. Woodhouse's family, less as a gov
erness than a friend, very fond of both daughters, but particularly of Emm
a.",
 'Between _them_ it was more the intimacy of sisters.']
```

- 문장들이 원소로 구성된 리스트를 얻을 수 있음.
- 문장 단위로 리스트를 만들고, 각 리스트 단어들을 토큰화하여 Word2Vec에 활용할 수 있음.

자연어 처리 패키지 실습 (NLTK - 2. pos_tag)

- tokenize한 단어들의 품사를 인식하여 태그해줌

```
In [36]: # nltk -2. pos_tag
         tags_en = nltk.pos_tag(tokens_en)

In [39]: tags_en[:20]

Out[39]: [('[', 'NNS'),
          ('Emma', 'NNP'),
          ('by', 'IN'),
          ('Jane', 'NNP'),
          ('Austen', 'NNP'),
          ('1816', 'CD'),
          (']', 'NNP'),
          ('VOLUME', 'NNP'),
          ('I', 'PRP'),
          ('CHAPTER', 'VBP'),
          ('I', 'PRP'),
          ('Emma', 'NNP'),
          ('Woodhouse', 'NNP'),
          (',', ','),
          ('handsome', 'NN'),
          (',', ','),
          ('clever', 'NN'),
          (',', ','),
          ('and', 'CC'),
          ('rich', 'JJ')]
```

- ('단어', '품사')의 튜플 형태로 묶은 리스트 형태
- 특정 품사 단어들만 뽑아내어 분석하고 싶을 때 활용 가능
- How?

자연어 처리 패키지 실습 (NLTK - 3. concordance)

- 목표 단어를 중심으로 하여 주변 글자들을 나열해주는 기능

```
In [44]: #nltk -3. concordance
         #타겟 단어를 찾아 주변 글자들을 나타내주어 사용된 문맥을 보여줌
         en.concordance('Emma', lines=15)
```

```
Displaying 15 of 25 matches:
 Emma by Jane Austen 1816 ] VOLUME I CHAPTE
 Emma Woodhouse , handsome , clever , and r
 both daughters , but particularly of Emma . Between _them_ it was more the inti
d friend very mutually attached , and Emma doing just what she liked ; highly es
er own . The real evils , indeed , of Emma 's situation were the power of having
dding-day of this beloved friend that Emma first sat in mournful thought of any
oing only half a mile from them ; but Emma was aware that great must be the diff
ay . It was a melancholy change ; and Emma could not but sigh over it , and wish
l the rest of her life at Hartfield . Emma smiled and chatted as cheerfully as s
 able to tell her how we all are . '' Emma spared no exertions to maintain this
. ' I have a great regard for you and Emma ; but when it comes to the question o
iful , troublesome creature ! '' said Emma playfully . `` That is what you have
he few people who could see faults in Emma Woodhouse , and the only one who ever
his was not particularly agreeable to Emma herself , she knew it would be so muc
ng thought perfect by every body . `` Emma knows I never flatter her , '' said M
```

- Lines 에 설정한 숫자만큼 사용된 부분을 표시해줌.
- 타겟 단어가 어떠한 문맥에서 사용되었는지 보여줌.
- 최대 25개의 단어만 찾아줌

Growth Hackers

자연어 처리 패키지 실습 (NLTK - 4. similar)

- Input에 넣은 단어와 비슷한 환경, 위치에서 사용된 단어들을 나열해줌

```
In [61]: #nltk -4. similar
         #Distributional similarity: find other words which appear in the
         #same contexts as the specified word; list most similar words first.

         en.similar('jane')

         he it emma she miss weston i harriet that her there you knightley
         elton all him me what then as
```

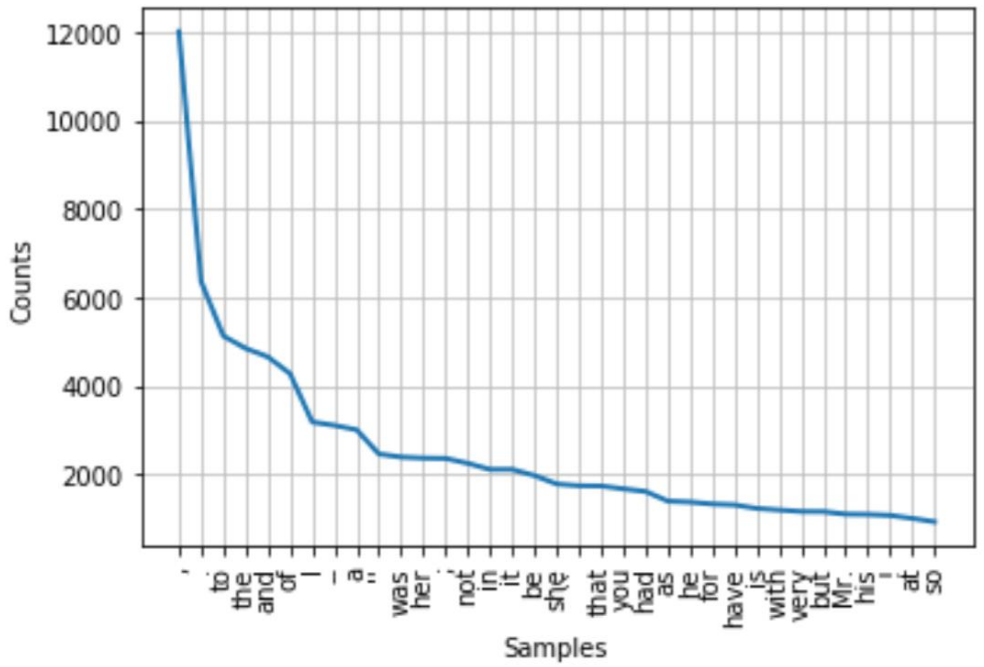
- Input 단어의 주변 단어들인 Context를 정의하고, 그 중 가장 많이 보인 단어들을 순서대로 출력함.

Growth Hackers

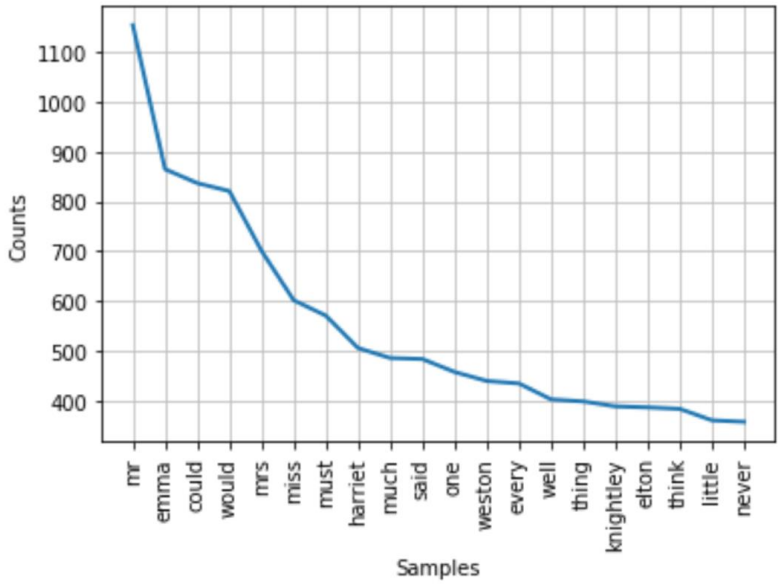
자연어 처리 패키지 실습 (NLTK - 5. plot)

- 1) plot - 가장 많이 사용된 단어들 순으로 꺾은선 그래프를 표현

```
In [33]: en.plot(35)      # Plot sorted frequency of top 50 tokens
```




```
tokenized_words = review_to_wordlist(doc_en,remove_stopwords=True)
en2 =nltk.Text(tokenized_words)
en2.plot(20)
```

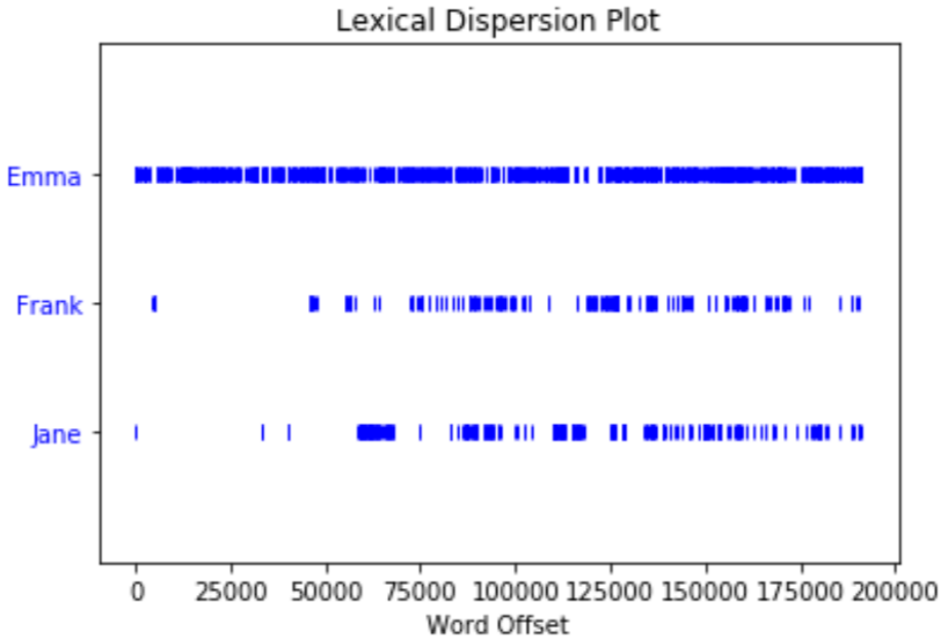


Growth Hackers

자연어 처리 패키지 실습 (NLTK - 5. plot)

- 2) dispersion plot - 선택한 단어들이 문서 전체에 분산 되어있는 위치를 표현

```
In [35]: #nltk -4.plot (2) dispersion plot
         en.dispersion_plot(['Emma', 'Frank', 'Jane'])
```



- 문서 안에서 선택한 단어가 어느 위치에서 많이 사용 되었는지 알 수 있음

Growth Hackers

WordCloud

- Wordcloud: 문서의 키워드, 개념 등을 직관적으로 파악할 수 있도록 핵심 단어를 시각적으로 돋보이게 하는 기법

- Python에서는 wordcloud 패키지가 따로 존재하여, 이를 불러와 생성 가능

- pip install wordcloud를 통해 wordcloud 패키지 설치

WordCloud - 주요 hyper parameters

- background_color: 배경 색깔 (default: 'black')
- max_words: Wordcloud에 올릴 최대 단어 종류 개수
- mask: 사용 이미지 설정
- colormap: 색상 설정 (default: 'viridis', matplotlib의 colormap 사용)
- stopwords: 불용어 list 설정
- max_font_size: 최빈 단어의 최대 크기 설정
- random_state: 배치의 임의성을 위해 설정된 숫자(정수)

· Colormap 관련 참조: https://matplotlib.org/users/colormaps.html

Growth Hackers

WordCloud 실습

- ‘session11 자연어처리 - 3. wordcloud.ipynb’ 참조!

- GH 이미지 파일과 주어진 텍스트로 wordcloud를 실습해보자!

- WordCloud의 default 색상이 아닌 원하는 색상으로 만들어보자!

WordCloud 실습

```
from wordcloud import WordCloud, STOPWORDS
from PIL import Image
from wordcloud import ImageColorGenerator

text = open('robinson_crusoe.txt',encoding='utf-8').read()

colored_gh = np.array(Image.open("gh_colored_2.png"))
stopwords = set(STOPWORDS)
stopwords.add("said")

wc = WordCloud(background_color="white", max_words=800, mask=colored_gh,
               stopwords=stopwords, max_font_size=40, random_state=42)
# generate word cloud
wc.generate(text)

# create coloring from image
image_colors = ImageColorGenerator(colored_gh)

# show
plt.figure(figsize=(15,15))
# recolor wordcloud and show
# we could also give color_func=image_colors directly in the constructor
plt.imshow(wc.recolor(color_func=image_colors), interpolation="bilinear")
plt.axis('off')
plt.show()
```

Growth Hackers

LDA

- LDA(Latent Dirichlet Allocation) - 잠재 디리클레 할당

  - 주어진 문서에 대하여 각 문서에 어떤 주제들이 분포하는지에 대한 확률적 토픽 모델
  - 문서가 토픽과 가진 연관성 정도(A)와 문서 내의 단어가 토픽과 가진 연관성의 정도(B)의 곱(A*B)을 통해 문서 내 단어가 어느 토픽에 할당 될지 확률을 알 수 있음.
  - Gensim이라는 패키지에 LDA 모델이 내장되어 있어 활용 가능

LDA

$$p(z_{d,i}=j|z_{-i},w) = \frac{n_{d,k}+\alpha_k}{\sum_{i=1}^{K}(n_{d,i}+\alpha_i)} \times \frac{v_{k,w_{d,n}}+\beta_{w_{d,n}}}{\sum_{j=1}^{V}(v_{k,j}+\beta_j)} = AB$$

표기	내용
$n_{d,k}$	k번째 토픽에 할당된 d번째 문서의 단어 빈도
$v_{k,w_{d,n}}$	전체 말뭉치에서 k번째 토픽에 할당된 단어 $w_{d,n}$의 빈도
$w_{d,n}$	d번째 문서에 n번째로 등장한 단어
α_k	문서의 토픽 분포 생성을 위한 디리클레 분포 파라메터
β_k	토픽의 단어 분포 생성을 위한 디리클레 분포 파라메터
K	사용자가 지정하는 토픽 수
V	말뭉치에 등장하는 전체 단어 수
A	d번째 문서가 k번째 토픽과 맺고 있는 연관성 정도
B	d번째 문서의 n번째 단어($w_{d,n}$)가 k번째 토픽과 맺고 있는 연관성 정도

출처: https://ratsgo.github.io/from%20frequency%20to%20semantics/2017/06/01/LDA/

Growth Hackers

LDA

```
lda_model = gensim.models.ldamodel.LdaModel(corpus=corpus,
                                            id2word=id2word,
                                            num_topics=20,
                                            random_state=100,
                                            update_every=1,
                                            chunksize=100,
                                            passes=10,
                                            alpha='auto',
                                            per_word_topics=True)
```

```
[(0,
  '0.069*"air" + 0.038*"ohio_state" + 0.026*"window_manager" + 0.026*"lee" + '
  '0.025*"dos" + 0.019*"henry_spencer" + 0.018*"gate" + 0.018*"ltd" + '
  '0.017*"toolkit" + 0.017*"borland"'),
 (1,
  '0.040*"cpu" + 0.035*"docs" + 0.034*"logo" + 0.030*"patch" + 0.026*"tonight" '
  '+ 0.017*"phillie" + 0.004*"attain" + 0.001*"padre" + 0.000*"libxmu" + '
  '0.000*"instal"'),
 (2,
  '0.036*"use" + 0.031*"problem" + 0.020*"drive" + 0.019*"card" + '
  '0.018*"system" + 0.014*"bit" + 0.012*"work" + 0.011*"high" + 0.011*"power" '
  '+ 0.010*"standard"'),
 (3,
  '0.059*"hall" + 0.044*"astronomy" + 0.036*"dare" + 0.031*"troop" + '
  '0.028*"sunday" + 0.025*"libertarian" + 0.012*"gary_dare" + '
  '0.007*"celebrate" + 0.006*"phds" + 0.004*"souviens_gary"'),
 (4,
  '0.071*"tv" + 0.068*"insurance" + 0.046*"steven" + 0.036*"taxis" + '
  '0.016*"rent" + 0.009*"tocchet" + 0.006*"announcer" + 0.002*"baltimore" + '
  '0.000*"pts_pt" + 0.000*"pgh"'),
```

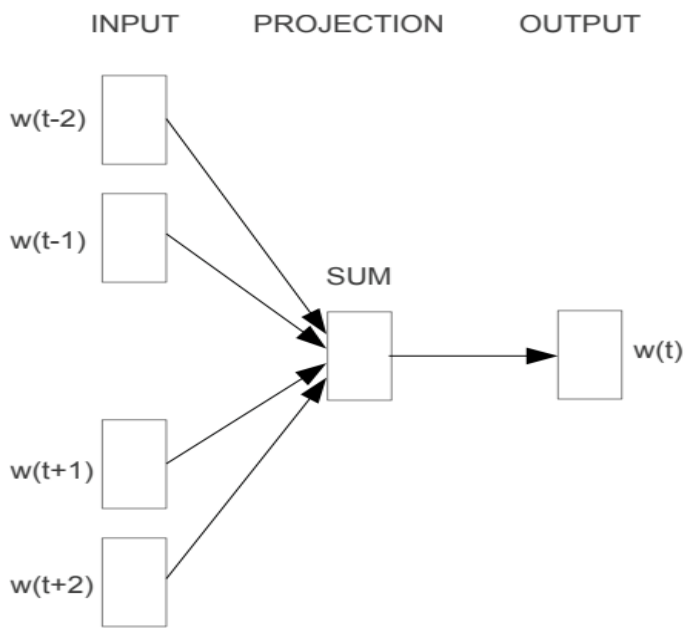

Growth Hackers

Word2Vec

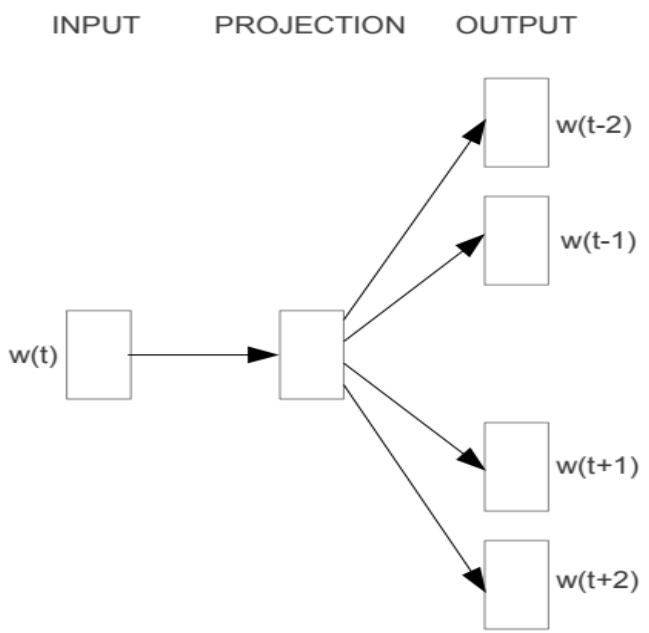
- Word2Vec
  - 단어를 벡터화하여 인공신경망을 형성하는 자연어처리의 방법론
  - 문서 안에서 단어 간의 유사도를 구할 수 있어 추천 시스템에도 사용되고 있음
  - Gensim 패키지에 내장된 Word2vec 모델 활용

  - ① CBOW(Continuous Bag of Words)와
    ② Skip-Gram 두가지 방식의 네트워킹 방법이 존재
    → Skip-Gram이 단어에 대한 업데이트의 기회가 2m배 많아져 모델 성능이 좋아 선호된다고 함.

Word2Vec



CBOW



Skip-gram

Figure 1: New model architectures. The CBOW architecture predicts the current word based on the context, and the Skip-gram predicts surrounding words given the current word.

Growth Hackers

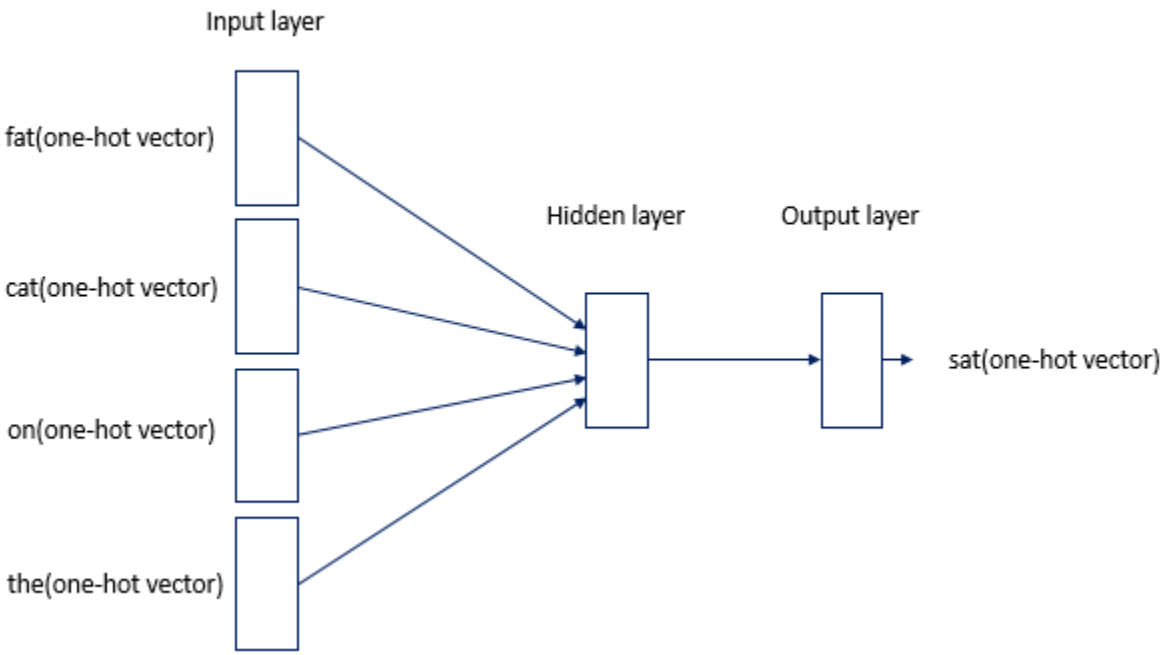
Word2Vec

- CBOW(Continuous Bag-of-Words)
  : 주변 단어(context word)들을 통해 중심 단어(center word)를 예측

중심 단어 / 주변 단어

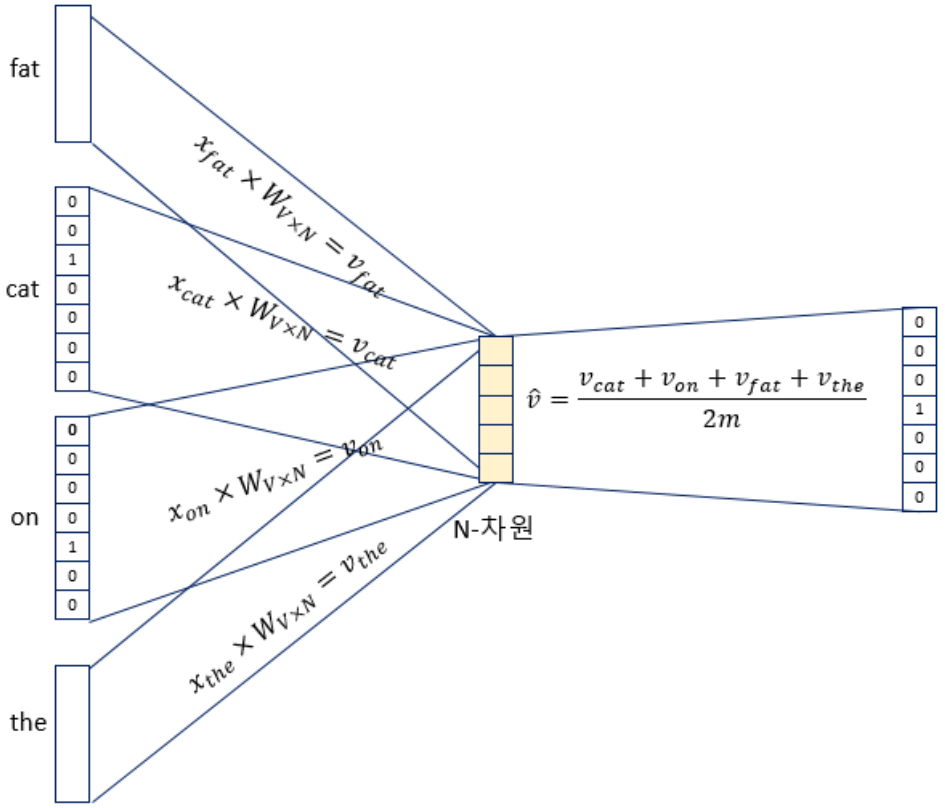
The fat cat sat on the mat
The fat cat sat on the mat
The fat cat sat on the mat
The fat cat sat on the mat
The fat cat sat on the mat
The fat cat sat on the mat
The fat cat sat on the mat
The fat cat sat on the mat

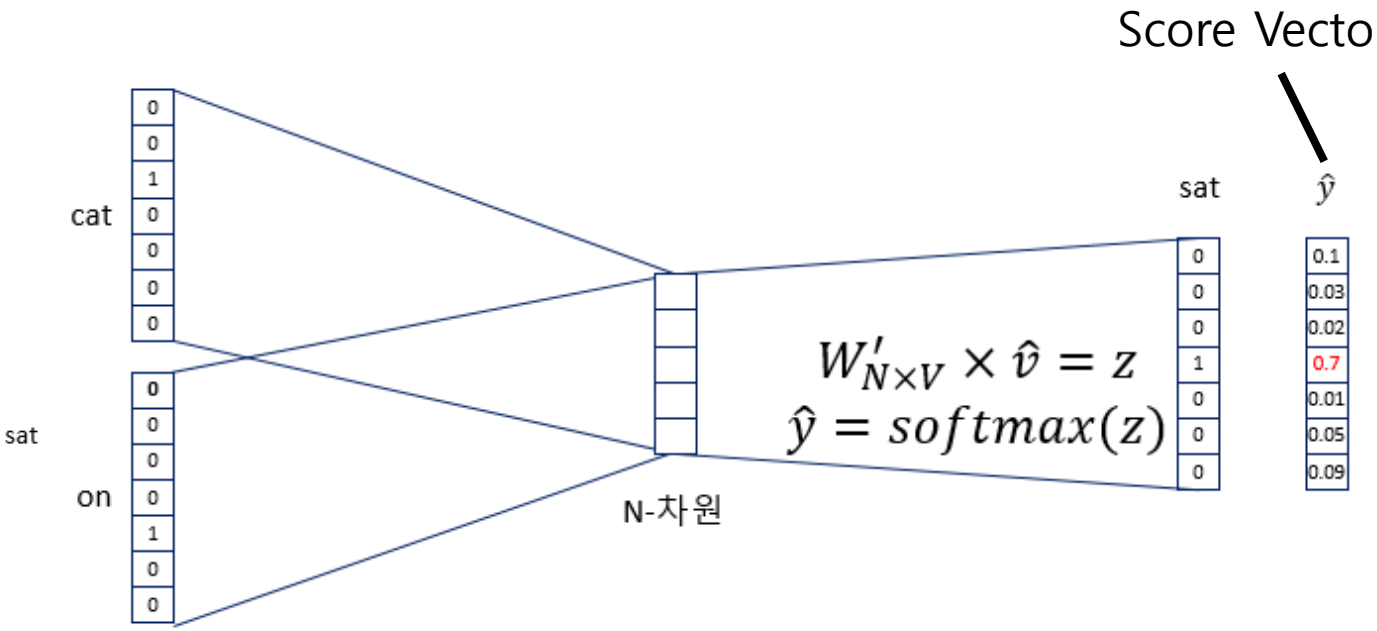
중심 단어	주변 단어
[1, 0, 0, 0, 0, 0, 0]	[0, 1, 0, 0, 0, 0, 0], [0, 0, 1, 0, 0, 0, 0]
[0, 1, 0, 0, 0, 0, 0]	[1, 0, 0, 0, 0, 0, 0], [0, 0, 1, 0, 0, 0, 0], [0, 0, 0, 1, 0, 0, 0]
[0, 0, 1, 0, 0, 0, 0]	[1, 0, 0, 0, 0, 0, 0], [0, 1, 0, 0, 0, 0, 0], [0, 0, 0, 1, 0, 0, 0], [0, 0, 0, 0, 1, 0, 0]
[0, 0, 0, 1, 0, 0, 0]	[0, 1, 0, 0, 0, 0, 0], [0, 0, 1, 0, 0, 0, 0], [0, 0, 0, 0, 1, 0, 0], [0, 0, 0, 0, 0, 1, 0]
[0, 0, 0, 0, 1, 0, 0]	[0, 0, 1, 0, 0, 0, 0], [0, 0, 0, 1, 0, 0, 0], [0, 0, 0, 0, 1, 0, 0], [0, 0, 0, 0, 0, 0, 1]
[0, 0, 0, 0, 0, 1, 0]	[0, 0, 1, 0, 0, 0, 0], [0, 0, 0, 0, 1, 0, 0], [0, 0, 0, 0, 0, 0, 1]
[0, 0, 0, 0, 0, 0, 1]	[0, 0, 0, 0, 1, 0, 0], [0, 0, 0, 0, 0, 1, 0]



Growth Hackers

Word2Vec



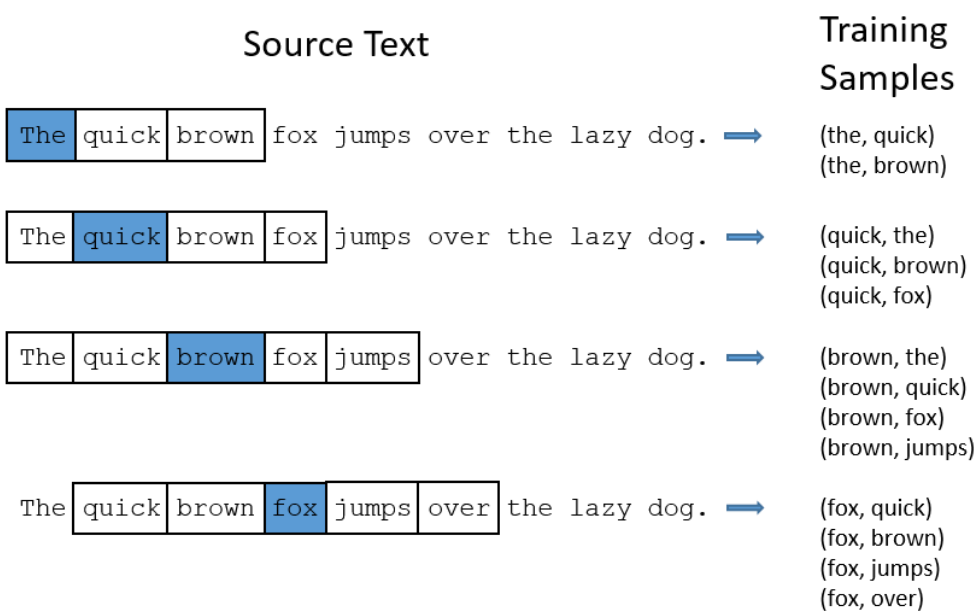


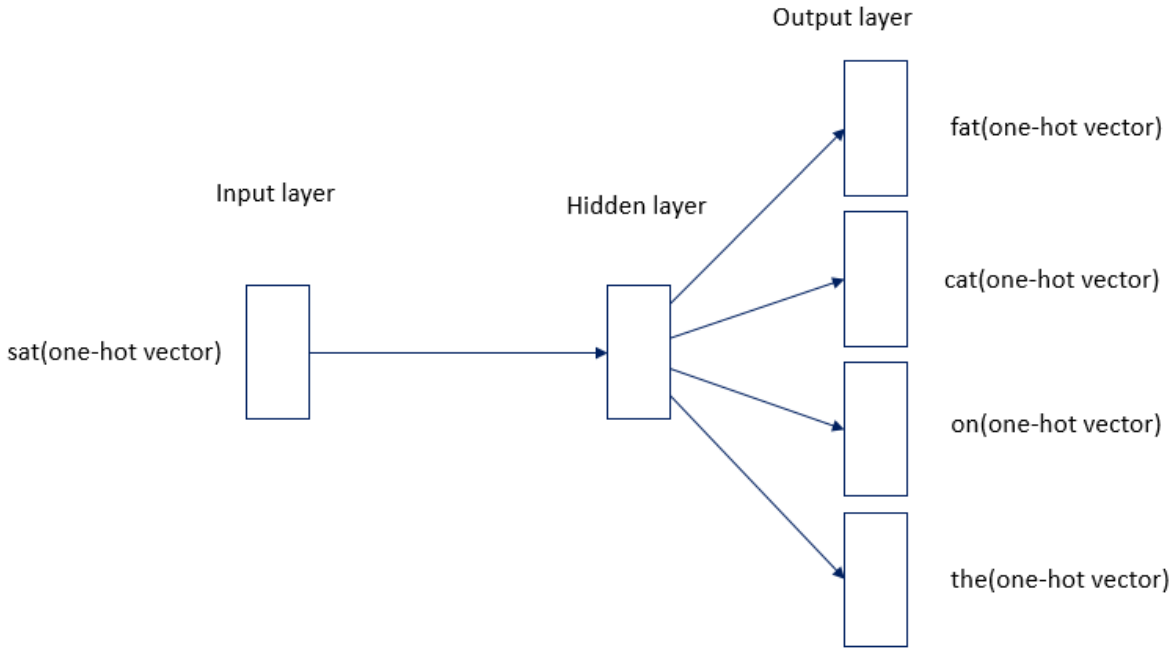
$$H(\hat{y}, y) = -\sum_{j=1}^{|V|} y_j \log(\hat{y}_j) \quad \rightarrow \quad H(\hat{y}, y) = -y_i \log(\hat{y}_i)$$

Growth Hackers

Word2Vec

- Skip-gram
  : 주변 단어(context word)들을 통해 중심 단어(center word)를 예측





Growth Hackers

Word2Vec

```
# Word2Vec 모델 만들기
# size = 워드 벡터의 특징 값. 즉, 임베딩 된 벡터의 차원.
# window = window의 수
# workers = 학습 프로세스 반복횟수. 일반적으로 3~4를 준다고 함.
# min_count --> 분석에 대상이 되기 위한 단어의 최소 등장 수
# sg =0 -> C-BOW 1--> Skip-Gram
from gensim.models.word2vec import Word2Vec
model = Word2Vec(sentences,size=100, window=3,workers=4, min_count=5, sg=1)

model.init_sims(replace=True)
```

```
print(model.wv.similarity('actor', 'actress'))
print(model.wv.similarity('music', 'soundtrack'))
print(model.wv.similarity('soundtrack','writer'))
```

```
0.8458925
0.766176
0.40383852
```

```
model.wv.most_similar('happy')
```

```
[('afraid', 0.7911931276321411),
 ('safe', 0.7901939153671265),
 ('starting', 0.7792547941207886),
 ('sappy', 0.7748895883560181),
 ('thrilled', 0.7744821310043335),
 ('cruel', 0.7713103294372559),
 ('scared', 0.7645026445388794),
 ('sad', 0.762326717376709),
 ('pleased', 0.7612833380699158),
 ('lucky', 0.7580583095550537)]
```

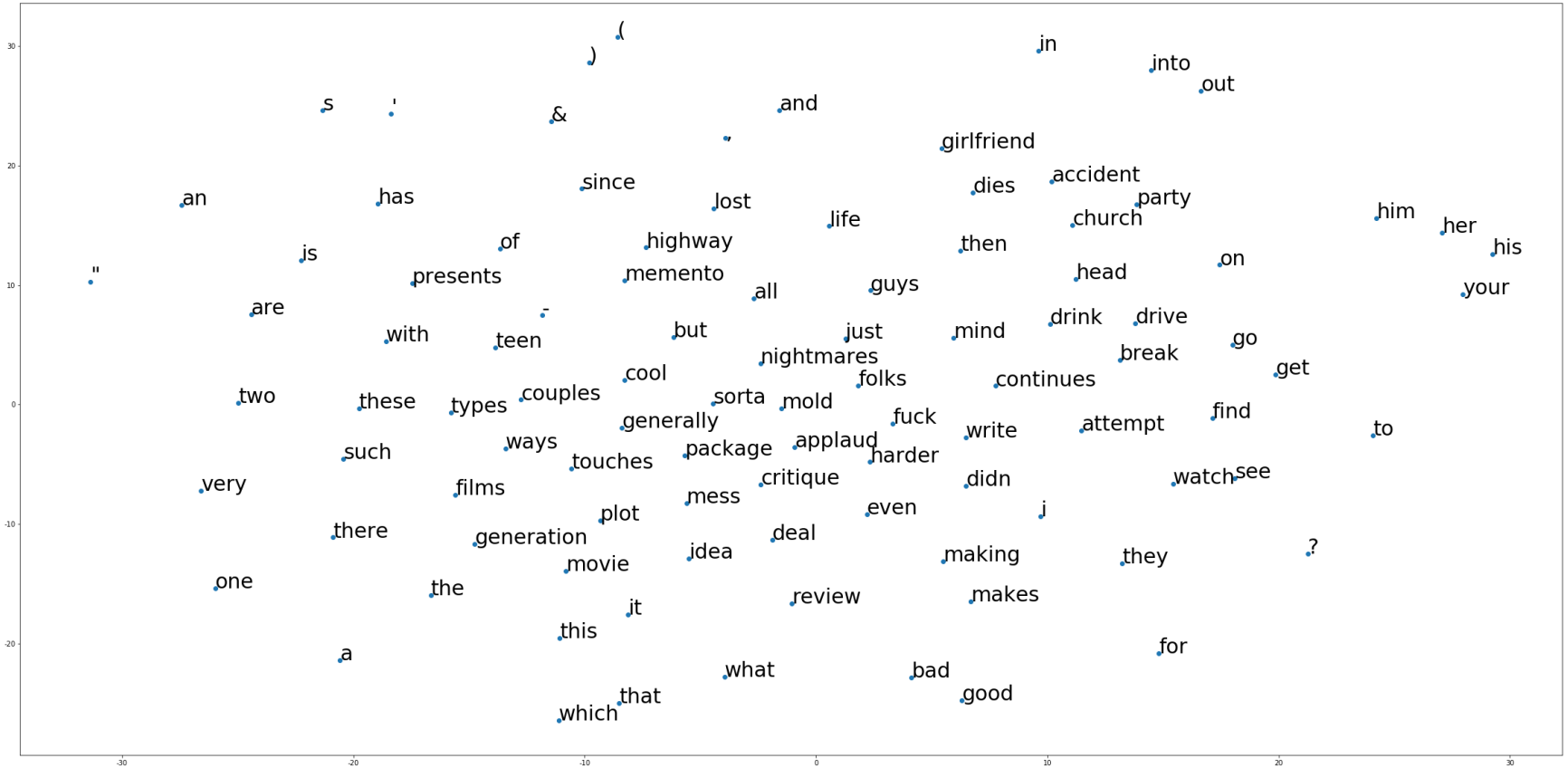
Word2Vec의 시각화 - t-SNE 활용

- t-SNE: 고차원의 벡터를 그래프에 나타내기 위해 차원을 낮추는 방법

  → 차원을 낮추면서, 그 값이 갖는 직접적인 의미는 없고, 벡터들 사이의 거리만 유효함

- Word2Vec 모델링 시 Workers 파라미터를 올릴수록 학습이 더욱 잘되어 시각화 시에도 비슷한 단어들끼리 더 뚜렷한 군집이 형성됨.

Word2Vec의 시각화 - t-SNE 활용



Growth Hackers

See also..

- KoNLPy (설치법)
  : 패키지 실습 - Lucy Park님의 블로그
  https://www.lucypark.kr/courses/2015-dm/text-mining.html
- Spacy
  : 패키지 설치 및 실습 - 유주원 님의 블로그
- 연어 처리를 통한 감정분석
  : https://programmers.co.kr/learn/courses/21/lessons/1692
- BoW, CounterVectorizer
  : https://programmers.co.kr/learn/courses/21/lessons/1695
- Word2Vec 미리 학습된 모델
  : 영어 모델 한국어 모델 사용법

Quest ☺

- 1. nltk.gutenburg에 있는 Corpus 중 원하는 텍스트를 불러와 전처리를 수행하고, 가장 많이 사용된 명사, 동사, 형용사를 보여주는 그래프를 만들어 보기 (top 20개)

- 2. nlp_data 폴더의 hotel-reviews.csv파일의 description 열에 있는 문장들을 Word2Vec 모델로 구현하고, t-sne를 활용하여 단어들의 유사성 분포를 시각화하기

References

- 자연어처리 패키지(NLTK & KoNLPy) 실습
  : Lucy Park님의 블로그
  https://www.lucypark.kr/courses/2015-dm/text-mining.html
- 데이터 전처리과정
  : 실습으로 배우는 데이터 사이언스 파트5 - 캐글 영화리뷰 분석 튜토리얼
  https://programmers.co.kr/learn/courses/21/lessons/946
- WordCloud & Word2Vec
  : 파이썬으로 데이터 주무르기
  https://github.com/PinkWink/DataScience
- LDA & Word2Vec 이론
  : ratsgo's blog for textmining
  https://ratsgo.github.io/blog/categories/#natural-language-processing
  딥 러닝을 이용한 자연어처리 입문
  https://wikidocs.net/22660