

미래에셋대우 빅데이터 페스티벌

키워드 추출 / 별난것들



- 트위터 운영, 별난 것들
- 필요성 및 목적
- 정확한 추출, LDA 알고리즘
- 알고리즘 아키텍처 구조도
- 키워드 추출 결과
- 프로토타입

1

트위터 운영, 별난것들

트위터 운영, 별난것들 별난것들 소개



■ 트위터 운영, 별난것들

- 저희는 트위터를 운영하고 있는 **별난것들** 팀입니다.

■ 뉴스 키워드 추출 및 제공

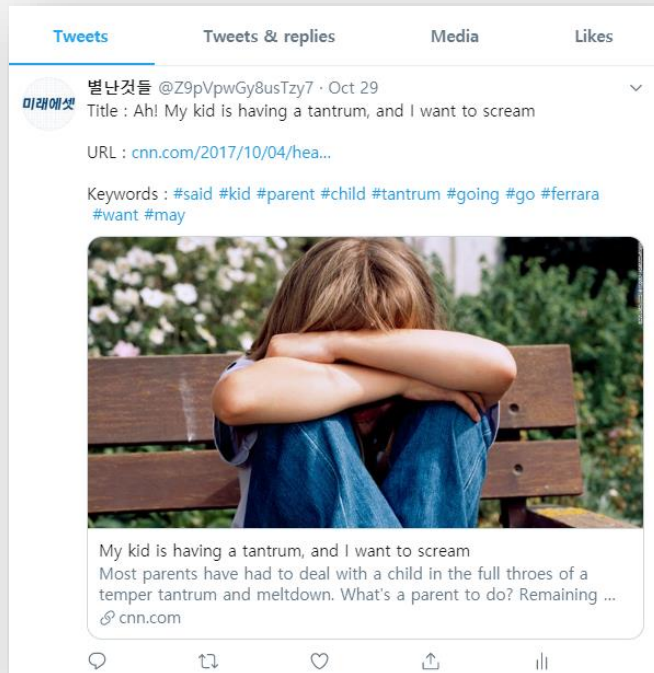
- 저희는 해외 뉴스의 키워드를 '**정확히**' 추출합니다.
- 저희는 뉴스의 키워드를 **자동으로** 업로드 합니다.

■ 별난것들@Z9pVpwGy8usTzy7

- 저희가 제공하는 콘텐츠를 설명 해 드리기 앞서,
FOLLOW 부탁드립니다.

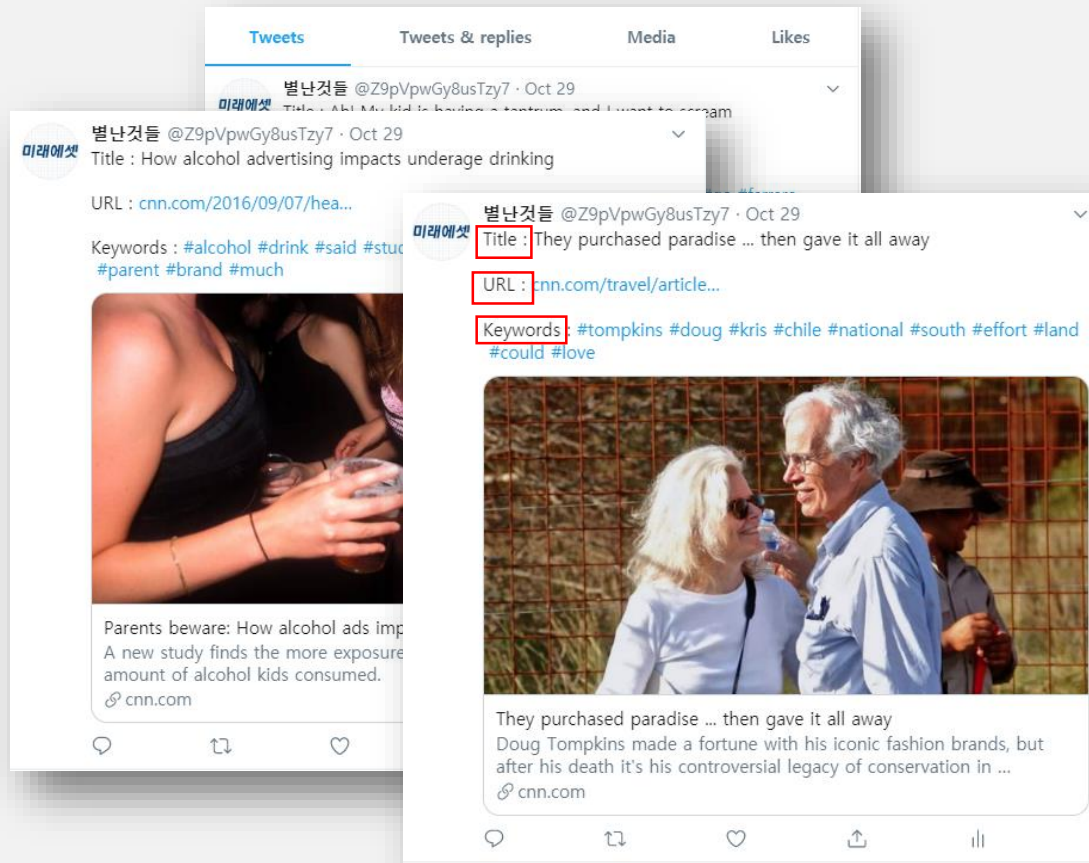
<https://twitter.com/Z9pVpwGy8usTzy7>

트위터 운영, 별난것들 콘텐츠 예시



트위터 별난것들에서 자동으로 제공하는 콘텐츠

트위터 운영, 별난것들 콘텐츠 소개



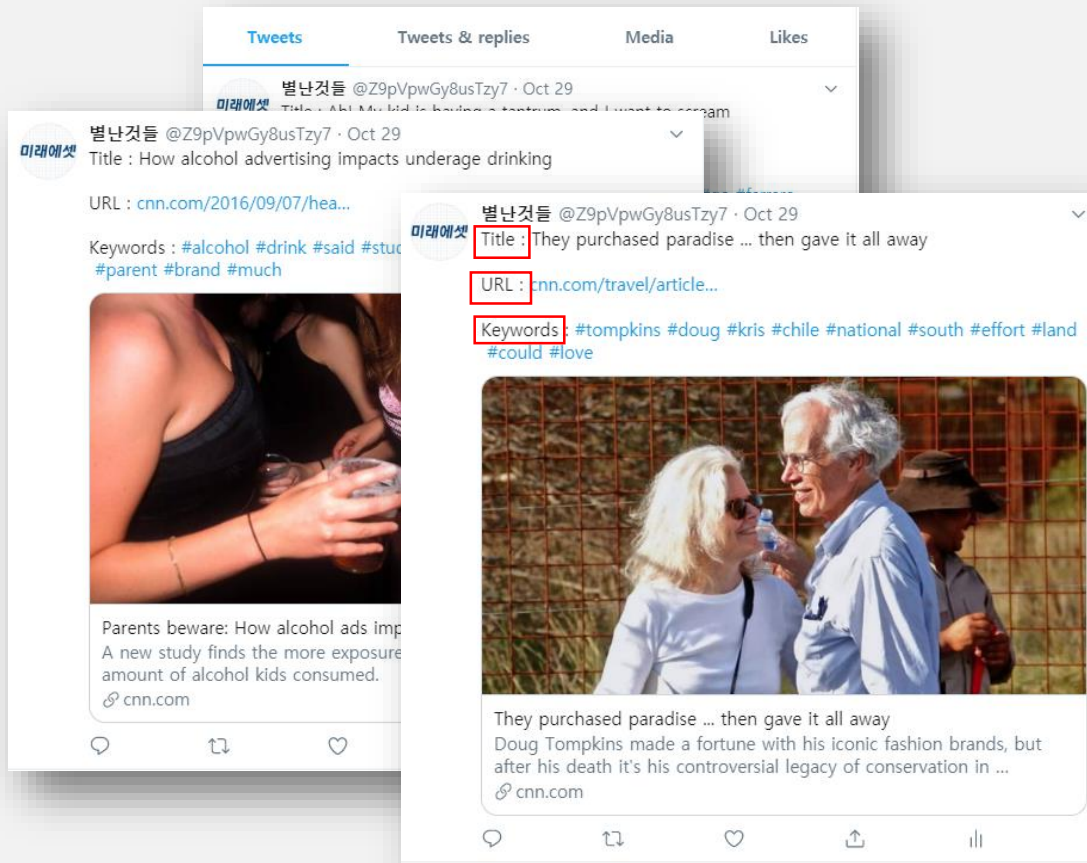
Title

- 저희는 글로벌 뉴스의 '**Title**'인 뉴스 제목을 제공해 줍니다.

URL

- 저희는 글로벌 뉴스의 '**URL**'을 제공합니다.
- 키워드를 보고 뉴스가 궁금한 독자가 들어가 볼 수 있도록 도와줍니다.

트위터 운영, 별난것들 콘텐츠 소개



Keywords

- 저희는 글로벌 뉴스의 **키워드 10개**를 제공합니다.
- 저희는 뉴스의 **정확한 키워드**를 제공합니다.
- 저희는 **해쉬태그**를 통해 해당 키워드의 연관 게시물
의 접근성을 용이하게 해줍니다.

자동 업로드

- 저희는 트위터 API를 이용해, **자동으로 업로드**를
합니다.

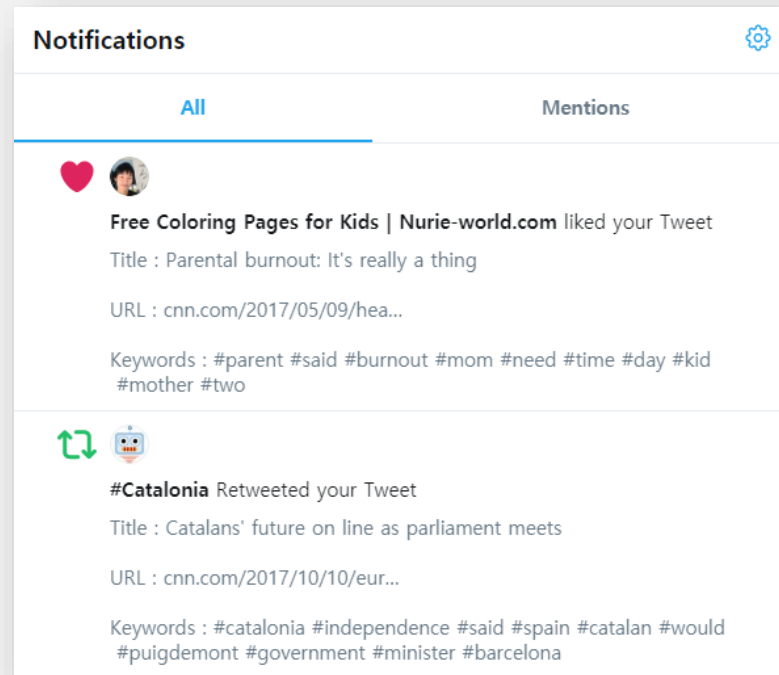
트위터 운영, 별난것들 별난것들 홍보



지금, 트위터 별난것들을 Follow 하고
무료로 뉴스의 키워드를 빠르게 제공 받으세요

#별난것들 #뉴스 #키워드 #정확성 #자동 #무료 #미래에셋대우 #빅데이터

트위터 운영, 별난것들 뜨거운 반응



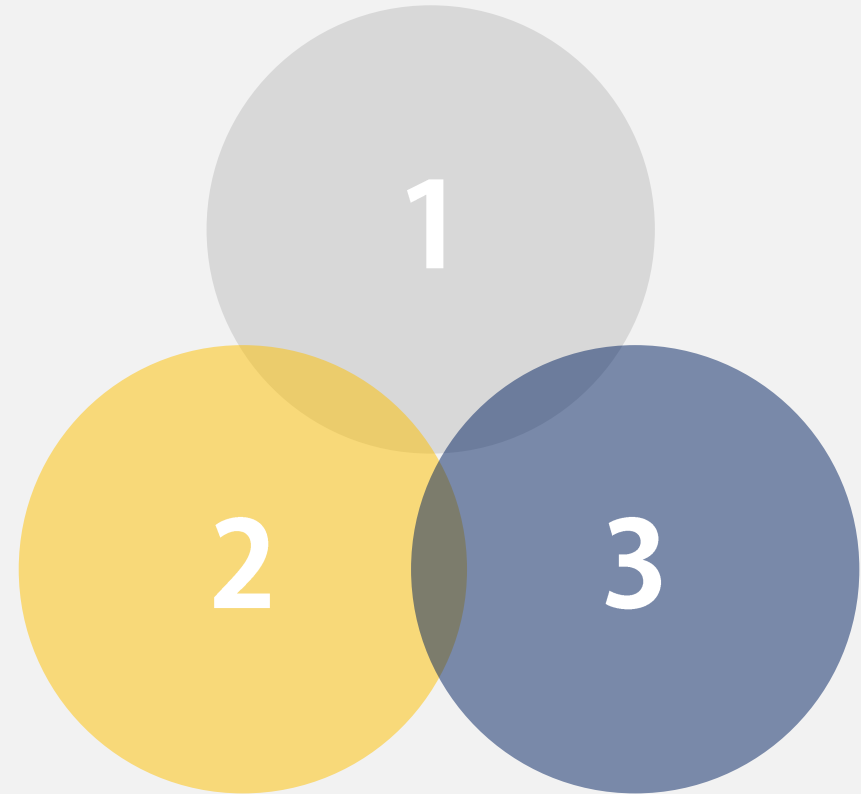
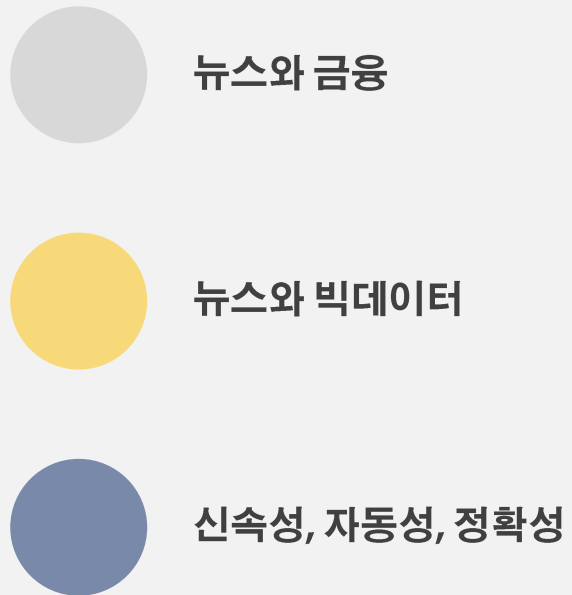
글로벌 독자들과 국내외 독자들의 뜨거운 반응!

을 기다리고 있습니다...

2

필요성 및 목적

필요성 및 목적 별난것들이 필요한 이유



필요성 및 목적

뉴스와 금융



- 금융권의 흐름은 경제, 사회, 문화, 정치의 상황을 반영합니다.
- 금융권의 흐름을 파악하기 위해 사회, 문화, 정치, IT, 스포츠 등 다양한 뉴스를 읽으며 경제 상황과 기업의 현황을 파악하는 것이 좋습니다.

필요성 및 목적

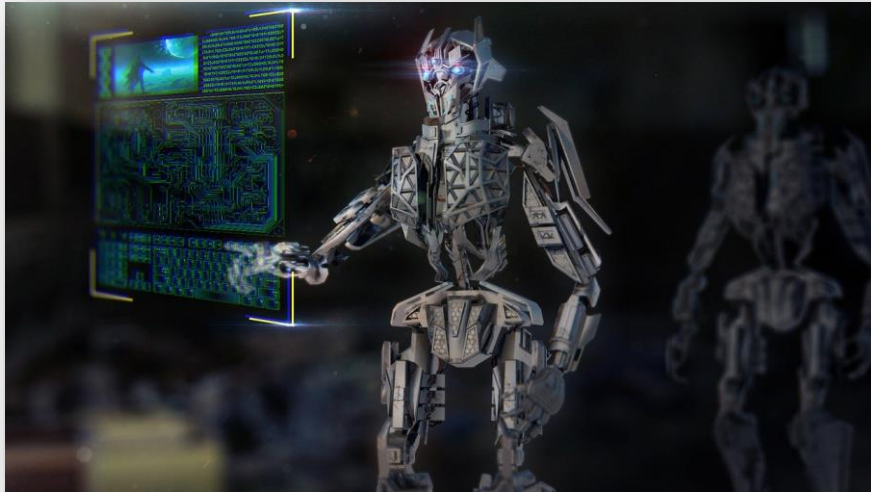
뉴스와 빅데이터



- 2016년 4월 기준 축적된 뉴스량은 **3천만건** 입니다. 데이터는 **하루 평균 1만 5천건**이 증가합니다.
- 하루에도 수 만 개씩 쏟아지는 뉴스 기사를 '**키워드**' 를 통해 요약해 줘서, 독자가 키워드를 통해 **뉴스를 선택**할 수 있도록 도와야 합니다.

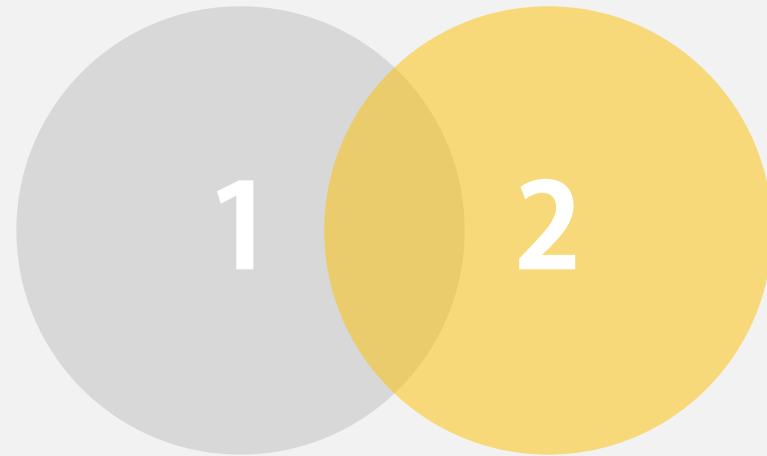
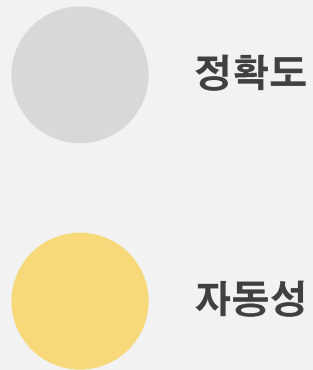
필요성 및 목적

신속성, 자동성, 정확성



- 트위터 운영을 통해 '뉴스 제목(Title), URL, 키워드' 를 독자들에게 **신속하고 정확**하게 제공합니다.
- 트위터 운영을 통해 **자동**으로 콘텐츠를 제공해서, 독자가 다양한 뉴스를 요약 제공 받을 수 있도록 도와줍니다.

필요성 및 목적 알고리즘 개발의 기본 방향



- 저희는 키워드 추출의 **정확도**를 올리기 위해 노력했습니다.
- 저희는 트위터로 뉴스와 키워드를 **자동 제공**할 수 있도록 노력했습니다.

3

정확한 추출, LDA 알고리즘

정확한 추출, LDA 알고리즘

LDA 알고리즘이란?

LDA 알고리즘

- LDA는 **토픽 모델링 알고리즘**입니다.
- LDA는 **주어진 문서에 어떤 주제**가 있는지 찾아주는 모델입니다.
- LDA는 **토픽별 단어의 분포**와 **문서별 토픽의 분포**를 추정해 냅니다.
- LDA는 토픽에서 단어가 나타날 **확률**을 알려줍니다.
- 저희는 뉴스의 키워드를 가장 잘 찾는 토픽 모델링 알고리즘을 찾기 위해 **LDA, LSI, HDP의 알고리즘**을 실험했습니다. 그 중 **LDA 알고리즘**이 가장 정확했습니다.

정확한 추출, LDA 알고리즘

LDA 알고리즘이란?

LDA 알고리즘 수식

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

- $n_{d,k}$: 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 : 문서의 토픽 분포 생성을 위한 디리클레 분포 파라미터
- β : 토픽의 단어 분포 생성을 위한 디리클레 분포 파라미터
- 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

정확한 추출, LDA 알고리즘

정확도를 높이기 위한 노력

데이터

```
In [4]: data['ALL_TEXT'] = data['TITLE'] + ' ' + data['TEXT'] # 제목과 Text 모두 이용  
data
```

- 키워드를 정확히 추출하기 위해, **타이틀**이 중요하다고 판단하였습니다.
- 키워드를 정확히 추출하기 위해 데이터로 **Title과 Text를 모두 이용**했습니다.

정확한 추출, LDA 알고리즘

정확도를 높이기 위한 노력

이상 데이터 처리

```
In [5]: def data_token(x):
        alltext = data.loc[x:x, 'ALL_TEXT'].tolist() * 5 # 리스트로 만들기
        # alltext를 5배, 10배.. 변경시켜도 output에 변화가 있다.
        # alltext가 키워드 추출하기에 너무 적은 단어 수로 구성되어 있을 경우, 5배를 해 주는게 효과적이다.
        alltext = [[wordnet_lemmatizer.lemmatize(z) for z in tokenizer.tokenize(str(t).lower()) if z not in STOPWORDS] for t in alltext] # 불분
        return alltext

        alltext = data_token(0)
        print(alltext)
```

- 뉴스 기사가 **굉장히 짧은 경우** 키워드 추출이 불가능 했습니다.
- 해당 데이터를 **5배 증식**시켜 키워드를 추출할 수 있었고, 정확도도 더 높아졌습니다.

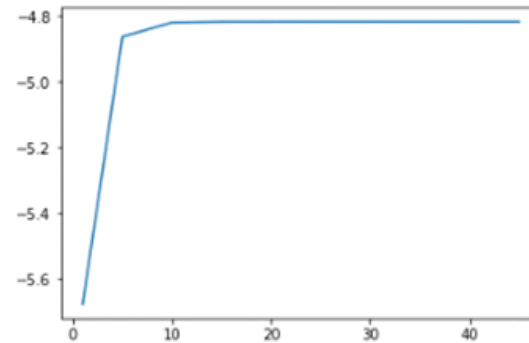
정확한 추출, LDA 알고리즘

정확도를 높이기 위한 노력

LDA 파라미터 튜닝

```
In [10]: x = find_pass  
y = find_perplexity  
plt.plot(x, y)  
  
# 결론 : perplexity는 낮을 수록 좋으므로, pass를 1로 설정하자.
```

Out[10]: [matplotlib.lines.Line2D at 0x1bb74c70f98]



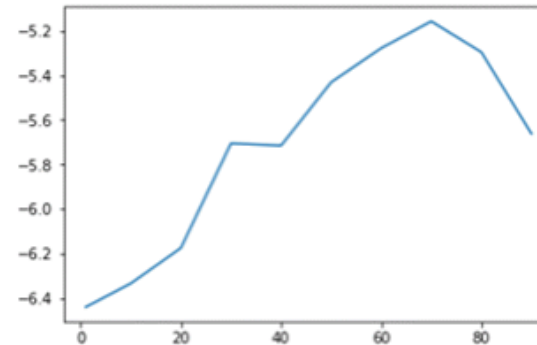
- LDA 파라미터 중 **pass**의 최적의 파라미터를 찾았습니다.
- LDA 파라미터인 pass의 값에 따른 Perplexity를 확인 해 본 결과 **최적의 값**인 1을 찾았습니다.

정확한 추출, LDA 알고리즘

정확도를 높이기 위한 노력

LDA 파라미터 튜닝

```
In [12]: x = find_iteration  
y = find_perplexity  
plt.plot(x, y)  
  
# 결론 : perplexity는 낮을 수록 좋으므로, iteration을 1로 설정하자.  
Out[12]: [<matplotlib.lines.Line2D at 0x1bb74cd8ef0>]
```



- LDA 파라미터 중 **Iteration**의 최적의 파라미터를 찾았습니다.
- LDA 파라미터인 Iteration의 값에 따른 Perplexity를 확인 해 본 결과 **최적의 값**인 1을 찾았습니다.

정확한 추출, LDA 알고리즘 최종 모델 성능은?

모델 성능

10. LDA 모델 평가

```
In [23]: lda.log_perplexity(corpus) # Perplexity는 낮을 수록 좋다.  
2019-10-29 00:47:09,451 : INFO : -6.437 per-word bound, 86.7 perplexity estimate based on a held-out corpus of 5 documents with 845 words  
Out[23]: -6.4372710650208425  
  
In [24]: coherence_model_lda = CoherenceModel(model=lda, texts=alltext, dictionary=dictionary) # Coherence Model은 높을 수록 좋다.  
  
In [25]: coherence_lda = coherence_model_lda.get_coherence()  
2019-10-29 00:47:09,725 : INFO : using ParallelWordOccurrenceAccumulator(processes=3, batch_size=64) to estimate probabilities from sliding windows  
2019-10-29 00:47:10,546 : INFO : 1 batches submitted to accumulate stats from 64 documents (300 virtual)  
2019-10-29 00:47:13,473 : INFO : 3 accumulators retrieved from output queue  
2019-10-29 00:47:13,489 : INFO : accumulated word occurrence stats for 300 virtual documents  
< _____ >  
  
In [26]: print(coherence_lda)  
0.4777908195146071
```

- LDA 모델 평가 결과 **-6.437의 Perplexity**를 얻을 수 있었습니다.
- LDA 모델 평가 결과 **0.477의 Coherence**를 얻을 수 있었습니다.

정확한 추출, LDA 알고리즘 평가 척도 두가지

평가 척도

10. LDA 모델 평가

```
In [23]: lda.log_perplexity(corpus) # Perplexity는 낮을 수록 좋다.  
2019-10-29 00:47:09,451 : INFO : -6.437 per-word bound, 86.7 perplexity estimate based on a held-out corpus of 5 documents with 845 words  
Out[23]: -6.4372710650208425  
  
In [24]: coherence_model_lda = CoherenceModel(model=lda, texts=alltext, dictionary=dictionary) # Coherence Model은 높을 수록 좋다.  
  
In [25]: coherence_lda = coherence_model_lda.get_coherence()  
2019-10-29 00:47:09,725 : INFO : using ParallelWordOccurrenceAccumulator(processes=3, batch_size=64) to estimate probabilities from sliding windows  
2019-10-29 00:47:10,546 : INFO : 1 batches submitted to accumulate stats from 64 documents (300 virtual)  
2019-10-29 00:47:13,473 : INFO : 3 accumulators retrieved from output queue  
2019-10-29 00:47:13,489 : INFO : accumulated word occurrence stats for 300 virtual documents  
< _____ >  
  
In [26]: print(coherence_lda)  
0.4777908195146071
```

- **Perplexity**는 혼란도를 의미합니다. 토픽모델링의 확률 모델을 평가하는 척도 중 하나로, 낮을 수록 좋습니다.
- **Coherence**는 주제 일관성을 의미합니다. 단어 유사도와 토픽의 유의미를 평가합니다. 높을 수록 좋습니다.

4

알고리즘 아키텍처 구조도

알고리즘 아키텍처 구조도



Input Text

Tokenize

Remove once

Corpus

파라미터 튜닝

LDA 생성 및 이용

키워드 추출

트위터 API

5

키워드 추출 결과

Windows 정품 인증
[설정]으로 이동하여 Windows를 정품 인증합니다.

키워드 추출 결과 CSV 파일 설명

KEYWORDS	TITLE	URL	PREDICT_KEYWORDS_PROB	PREDICT_KEYWORDS	HASHTAG
energy, sugar	https://[bar, 0.08965789]	energy, sugar	energy, sugar	energy, sugar	energy, sugar
facebook, tamagotchi	https://[unfolds, 0.114573814]	facebook, tamagotchi	facebook, tamagotchi	facebook, tamagotchi	facebook, tamagotchi
photo, star	https://[unfolds, 0.114573814]	photo, star	photo, star	photo, star	photo, star
clients, ar141	https://[unfolds, 0.114573814]	clients, ar141	clients, ar141	clients, ar141	clients, ar141
akufordso, seven	https://[unfolds, 0.114573814]	akufordso, seven	akufordso, seven	akufordso, seven	akufordso, seven
spanish, catalunya	https://[unfolds, 0.114573814]	spanish, catalunya	spanish, catalunya	spanish, catalunya	spanish, catalunya
pres, the	https://[unfolds, 0.114573814]	pres, the	pres, the	pres, the	pres, the
pollution, health	https://[unfolds, 0.114573814]	pollution, health	pollution, health	pollution, health	pollution, health
look, trump	https://[unfolds, 0.114573814]	look, trump	look, trump	look, trump	look, trump
okunon, yamadera	https://[unfolds, 0.114573814]	okunon, yamadera	okunon, yamadera	okunon, yamadera	okunon, yamadera
land, life	https://[unfolds, 0.114573814]	land, life	land, life	land, life	land, life
kids, parent	https://[unfolds, 0.114573814]	kids, parent	kids, parent	kids, parent	kids, parent
kids, terror	https://[unfolds, 0.114573814]	kids, terror	kids, terror	kids, terror	kids, terror
congress, ivanka	https://[unfolds, 0.114573814]	congress, ivanka	congress, ivanka	congress, ivanka	congress, ivanka
partner, when	https://[unfolds, 0.114573814]	partner, when	partner, when	partner, when	partner, when
hey, music	https://[unfolds, 0.114573814]	hey, music	hey, music	hey, music	hey, music
plywood, plywood	https://[unfolds, 0.114573814]	plywood, plywood	plywood, plywood	plywood, plywood	plywood, plywood
men, result	https://[unfolds, 0.114573814]	men, result	men, result	men, result	men, result
according, here	https://[unfolds, 0.114573814]	according, here	according, here	according, here	according, here
hollywood, donna	https://[unfolds, 0.114573814]	hollywood, donna	hollywood, donna	hollywood, donna	hollywood, donna
reckoned, faced	https://[unfolds, 0.114573814]	reckoned, faced	reckoned, faced	reckoned, faced	reckoned, faced
online, res	https://[unfolds, 0.114573814]	online, res	online, res	online, res	online, res
important, she	https://[unfolds, 0.114573814]	important, she	important, she	important, she	important, she
planet, books	https://[unfolds, 0.114573814]	planet, books	planet, books	planet, books	planet, books
james, tele	https://[unfolds, 0.114573814]	james, tele	james, tele	james, tele	james, tele
planned, the	https://[unfolds, 0.114573814]	planned, the	planned, the	planned, the	planned, the
rovers, hidis	https://[unfolds, 0.114573814]	rovers, hidis	rovers, hidis	rovers, hidis	rovers, hidis
house, main	https://[unfolds, 0.114573814]	house, main	house, main	house, main	house, main
in, strike	https://[unfolds, 0.114573814]	in, strike	in, strike	in, strike	in, strike
their, victim	https://[unfolds, 0.114573814]	their, victim	their, victim	their, victim	their, victim
group, art	https://[unfolds, 0.114573814]	group, art	group, art	group, art	group, art
users, too	https://[unfolds, 0.114573814]	users, too	users, too	users, too	users, too
impossible, a	https://[unfolds, 0.114573814]	impossible, a	impossible, a	impossible, a	impossible, a
camp, age	https://[unfolds, 0.114573814]	camp, age	camp, age	camp, age	camp, age
stuck, not	https://[unfolds, 0.114573814]	stuck, not	stuck, not	stuck, not	stuck, not
tried, job	https://[unfolds, 0.114573814]	tried, job	tried, job	tried, job	tried, job
hope, hater	https://[unfolds, 0.114573814]	hope, hater	hope, hater	hope, hater	hope, hater
uk, save	https://[unfolds, 0.114573814]	uk, save	uk, save	uk, save	uk, save
planning, china	https://[unfolds, 0.114573814]	planning, china	planning, china	planning, china	planning, china
men, house	https://[unfolds, 0.114573814]	men, house	men, house	men, house	men, house
exercising, parallel	https://[unfolds, 0.114573814]	exercising, parallel	exercising, parallel	exercising, parallel	exercising, parallel
tray, trans	https://[unfolds, 0.114573814]	tray, trans	tray, trans	tray, trans	tray, trans

PREDICT_KEYWORDS_PROB

- 모델을 통해 키워드를 추출하고, 키워드가 토픽일 확률을 알려줍니다.

PREDICT_KEYWORDS

- 모델을 통해 예측한 키워드입니다.

HASHTAG

- 모델을 통해 예측한 키워드에 HASHTAG를 붙여 트위터에 자동 업로드 할 수 있도록 했습니다.

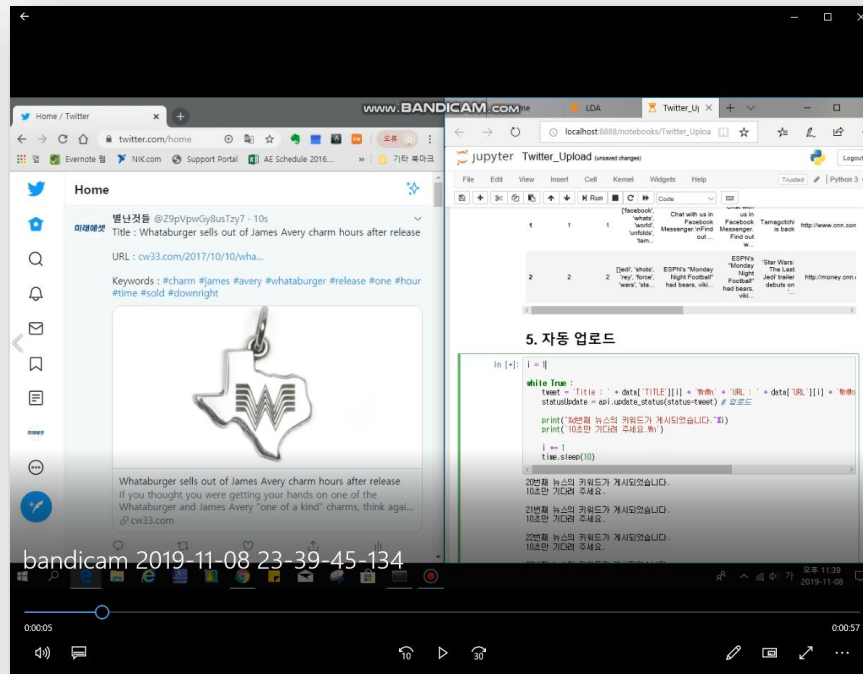
정답 키워드 : ['akufoaddo', 'tanker', 'incidents', 'dozens', 'streetsgovernment', 'work', 'station', 'seven', 'explosion', 'gas', 'ghana', 'nana', 'injured', 'ensure', 'killed']

예측 키워드: ['said', 'incident', 'ghana', 'gas', 'addo', 'ensure', 'seven', 'government', 'dozen', 'killed']

6

프로토 타입

프로토 타입 프로토 타입 설명



프로토 타입

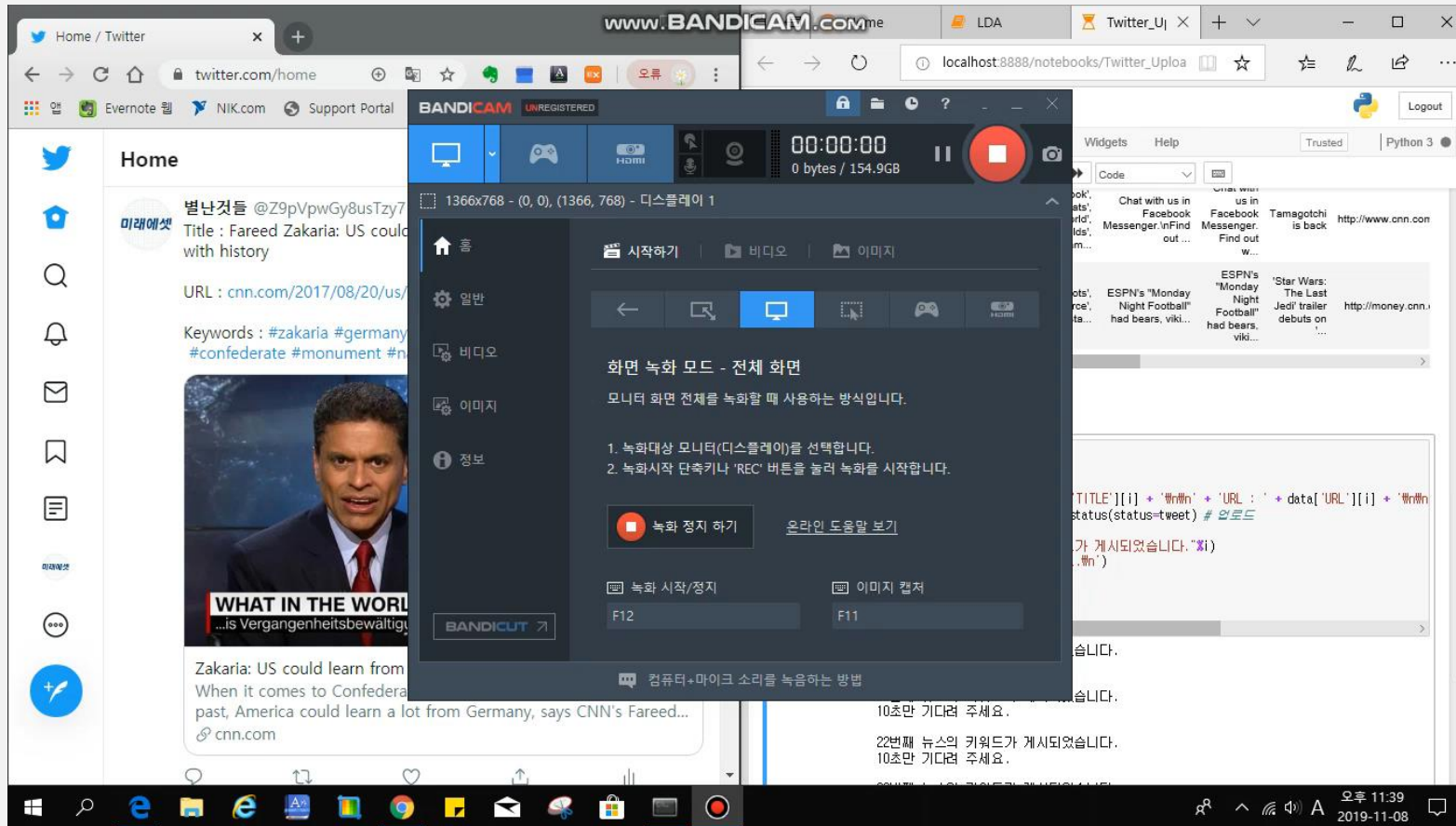
- 프로토 타입은 실제 시연 영상입니다.

시연

- 시연을 위해 업로드 시간을 10초마다로 설정했습니다.
- 시연 이후 jupyter notebook을 종료했기 때문에, 이후 계속해서 콘텐츠가 업로드 되지는 않습니다.

프로토 타입

프로토 타입 영상



레퍼런스

<http://www.bloter.net/archives/254773>

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

<https://bab2min.tistory.com/587>

감사합니다

