Social Data Analytics

서울대학교 융합과학부
서봉원
2014/2/12

bongwon@snu.ac.kr

* Everything you see in this presentation is solely my personal view
Data Driven Approach with Big Data

What is Big Data?

- **Volume**: Large amounts of data.
- **Velocity**: Needs to be analyzed quickly.
- **Variety**: Different types of structured and unstructured data.

What are the volumes of data that we are seeing today?

**Facebook**: 30 billion pieces of content were added to Facebook this past month by 600 million plus users.

**Zynga**: Zynga processes 1 petabyte of content for players every day; a volume of data that is unmatched in the social game industry.

**YouTube**: More than 2 billion videos were watched on YouTube... yesterday.

**LOL!**: The average teenager sends 4,782 text messages per month.

**Twitter**: 32 billion searches were performed last month... on Twitter.

What does the future look like?

Worldwide IP traffic will quadruple by 2015.

By 2015, nearly 3 billion people will be online, pushing the data created and shared to nearly 8 zettabytes.

http://www.flickr.com/photos/bbvatech/7979558647/lightbox/
How to Obtain Valuable Insights?

Data

Information

Insight

Advice
## Anscombe’s Quartet

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
</tr>
</thead>
<tbody>
<tr>
<td>x</td>
<td>y</td>
<td>x</td>
<td>y</td>
<td>x</td>
</tr>
<tr>
<td>10.0</td>
<td>8.04</td>
<td>10.0</td>
<td>9.14</td>
<td>10.0</td>
</tr>
<tr>
<td>8.0</td>
<td>6.95</td>
<td>8.0</td>
<td>8.14</td>
<td>8.0</td>
</tr>
<tr>
<td>13.0</td>
<td>7.58</td>
<td>13.0</td>
<td>8.74</td>
<td>13.0</td>
</tr>
<tr>
<td>9.0</td>
<td>8.81</td>
<td>9.0</td>
<td>8.77</td>
<td>9.0</td>
</tr>
<tr>
<td>11.0</td>
<td>8.33</td>
<td>11.0</td>
<td>9.26</td>
<td>11.0</td>
</tr>
<tr>
<td>14.0</td>
<td>9.96</td>
<td>14.0</td>
<td>8.10</td>
<td>14.0</td>
</tr>
<tr>
<td>6.0</td>
<td>7.24</td>
<td>6.0</td>
<td>6.13</td>
<td>6.0</td>
</tr>
<tr>
<td>4.0</td>
<td>4.26</td>
<td>4.0</td>
<td>3.10</td>
<td>4.0</td>
</tr>
<tr>
<td>12.0</td>
<td>10.84</td>
<td>12.0</td>
<td>9.13</td>
<td>12.0</td>
</tr>
<tr>
<td>7.0</td>
<td>4.82</td>
<td>7.0</td>
<td>7.26</td>
<td>7.0</td>
</tr>
<tr>
<td>5.0</td>
<td>5.68</td>
<td>5.0</td>
<td>4.74</td>
<td>5.0</td>
</tr>
</tbody>
</table>
## Anscombe’s Quartet

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
</tr>
</thead>
<tbody>
<tr>
<td>x</td>
<td>10.0</td>
<td>8.04</td>
<td>10.0</td>
<td>7.46</td>
</tr>
<tr>
<td>y</td>
<td>8.0</td>
<td>6.95</td>
<td>8.0</td>
<td>6.77</td>
</tr>
<tr>
<td></td>
<td>13.0</td>
<td>7.58</td>
<td>13.0</td>
<td>8.74</td>
</tr>
<tr>
<td>x</td>
<td>9.0</td>
<td>8.81</td>
<td>9.0</td>
<td>8.77</td>
</tr>
<tr>
<td>y</td>
<td>11.0</td>
<td>8.33</td>
<td>11.0</td>
<td>9.26</td>
</tr>
<tr>
<td></td>
<td>14.0</td>
<td>9.96</td>
<td>14.0</td>
<td>8.10</td>
</tr>
<tr>
<td>x</td>
<td>6.0</td>
<td>7.24</td>
<td>6.0</td>
<td>6.13</td>
</tr>
<tr>
<td>y</td>
<td>4.0</td>
<td>4.26</td>
<td>4.0</td>
<td>3.10</td>
</tr>
<tr>
<td></td>
<td>12.0</td>
<td>10.84</td>
<td>12.0</td>
<td>9.13</td>
</tr>
<tr>
<td>x</td>
<td>7.0</td>
<td>4.82</td>
<td>7.0</td>
<td>7.26</td>
</tr>
<tr>
<td>y</td>
<td>5.0</td>
<td>5.68</td>
<td>5.0</td>
<td>4.74</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Property</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mean of x</td>
<td>9.0</td>
</tr>
<tr>
<td>Variance of x</td>
<td>11.0</td>
</tr>
<tr>
<td>Mean of y</td>
<td>7.5</td>
</tr>
<tr>
<td>Variance of y</td>
<td>4.12</td>
</tr>
<tr>
<td>Correlation</td>
<td>0.816</td>
</tr>
<tr>
<td>Linear regression</td>
<td>$y = 3 + 0.5x$</td>
</tr>
</tbody>
</table>

Anscombe’s Quartet

http://www.cs.umd.edu/hcil/pubs/presentations/
Information Visualization

- 컴퓨터를 이용해 인간의 인지를 확장시키는 것
  - 못보던 것을 보게 한다!
- 정보를 인간이 효율적으로 이해할 수 있도록 시각화
정보 시각화의 Task

• Analysis (분석)
  – 비교 분석, 차이 비교
  – 이상점, 극단점 찾기
  – 패턴 분석
• Data assimilation (데이터 동화)
  – 날씨 모델등에서 사용
  – 여러 가지 관찰된 정보를 종합하여 현재 상황 예측
• Monitoring (관찰, 감독)
• Awareness (상황인지)

목표
  – 사람의 인지를 확장시킨다
  – 통찰의 발견
  – 스토리를 전달하고 설득시킨다
이산화 탄소와 지구 온난화

Temperature and CO₂ Records

http://en.wikipedia.org/wiki/An_Inconvenient_Truth
An Inconvenient Truth
Social Computing

• (Broader) Definition
  – an area of computer science concerning on the intersection of social behavior and computational systems

• 사회적으로 만들어진 정보를 다시 사용자에게 제공한다
  • 직접: Wikipedia
  • 간접: 유사사용자 정보를 이용한 추천

• 잠재력이 높지만 실제로 적용하기 어렵다
If all U.S. Internet time were condensed into one hour, how much time would be spent in the most heavily used sectors?

- Social Networks/ Blogs: 13m 36s
- Games: 6m 6s
- E-mail: 5m 00s
- Portals: 2m 36s
- Instant Messaging: 2m 24s
- Search/Movies: 2m 18s
- Software Info: 2m 00s
- Multi-category Entertainment: 1m 42s
- Classifieds/Auctions: 1m 36s
- *Other: 20m 36s

Source: Nielsen NetView, June 2010
Why Media Companies Must Embrace Facebook

Consumption in Total Minutes of U.S. Web Use (Indexed)

Facebook: 69% Growth

Rest of the Web: (9%) Decline

Source: Ben Elowitz, Wetpaint / comScore
Predicting revenues for movies

Prediction of second weekend box office gross

$r^2 = 0.94$

[Asur and Huberman 2010]
Twitter mood predict the stock market

[Bollen, Mao, & Zeng 2010]

87.6% in predicting the daily up and down changes in closing values of the DJIA

http://huff.to/e94Ter
<table>
<thead>
<tr>
<th>Category</th>
<th>Count</th>
<th>Percentage</th>
</tr>
</thead>
<tbody>
<tr>
<td>Total News</td>
<td>72</td>
<td>3.60%</td>
</tr>
<tr>
<td>Total Spam</td>
<td>75</td>
<td>3.75%</td>
</tr>
<tr>
<td>Total Self promotion</td>
<td>117</td>
<td>5.85%</td>
</tr>
<tr>
<td>Total Pointless Babble</td>
<td>811</td>
<td>40.55%</td>
</tr>
<tr>
<td>Total Conversational</td>
<td>751</td>
<td>37.55%</td>
</tr>
<tr>
<td>Total Pass along value</td>
<td>174</td>
<td>8.70%</td>
</tr>
<tr>
<td><strong>Total</strong></td>
<td>2000</td>
<td><strong>100.00%</strong></td>
</tr>
</tbody>
</table>

- **Winner** -

“Pointless Babble”

Mining the Social Web 2nd Ed.

- http://www.amazon.com/Mining-Social-Web-Facebook-LinkedIn/dp/1449367615
Goal: Build a Word Cloud Summarizing 500 Tweets
Preparation – Python과 pip (설치관리자) 설치

- Windows
  - Python
    - http://www.python.org/download/  2.7.X 다운로드 받아서 설치
    - 32 bit버전을 설치 (64 bit 버전인 X86-64를 설치하지 마세요!!!)
  - Easy_install
    - Download http://peak.telecommunity.com/dist/ez_setup.py
    - command line에서 c:\python27\python ez_setup.py
  - Pip
    - command line에서 c:\python27\scripts\easy_install pip

- Mac / Linux
  - Python
    - Mac: 기본으로 포함되어 있음
    - Linux:
      - sudo yum install python27
      - sudo apt-get install python2.7
    - CentOS의 경우는 복잡한 설치를 해야 함
  - pip
    - sudo easy_install pip
NLTK (Natural Language Toolkit) 설치

- Windows
  - Install Numpy (optional):
    - http://sourceforge.net/projects/numpy/files/NumPy/1.6.2/numpy-1.6.2-win32-superpack-python2.7.exe
  - Install NLTK:
    - http://pypi.python.org/pypi/nltk
    - XXXX.win32.exe 를 선택
  - Install PyYAML:
    - http://pyyaml.org/wiki/PyYAML
    - XXXX.win32-py2.7 을 선택
  - Test installation:
    - Start>Python27, then type import nltk

- Mac / Linux
  - 뒤 슬라이드의 virtualenv를 먼저 하고 다음을 실행한다.
  - pip install -U pyyaml nltk
**Virtualenv – 실험용 Python 공간**

- **Windows**
  - Command line에서 `c:\python27\scripts\pip install virtualenv`
  - `cd`
  - `mkdir python`
  - `cd python`
  - `c:\python27\scripts\virtualenv tutorial`
  - `c:\python\tutorial\scripts\activate.bat`

- **Mac / Linux**
  - `sudo pip install virtualenv`
  - `mkdir ~/.python`
  - `cd ~/.python`
  - `virtualenv tutorial`
  - `source ~/.python/tutorial/bin/activate`
  - **NLTK 설치**
    - `pip install -U pyyaml nltk`
Twitter API

- 프로그램으로 Twitter에 접속.
- 자료의 수집등을 자동으로 할 수 있다.
- 휴대폰의 Twitter 프로그램이나 데스크 탑의 프로그램들이 Twitter API를 이용한다.
- 이용규정: [https://twitter.com/tos](https://twitter.com/tos)
- Document: [https://dev.twitter.com/docs](https://dev.twitter.com/docs)
GET statuses/show/:id

Updated on Thu, 2013-03-07 09:37

Returns a single Tweet, specified by the id parameter. The Tweet’s author will also be embedded within the tweet.

See Embeddable Timelines, Embeddable Tweets, and GET statuses/embed for tools to render Tweets according to Display Requirements.

Resource URL
https://api.twitter.com/1.1/statuses/show.json

Parameters

**id**
required

The numerical ID of the desired Tweet.

Example Values: 123

**trim_user**
optional

When set to either true, t or 1, each tweet returned in a timeline will include a user object including only the status author’s numerical ID. Omit this parameter to receive the complete user object.

Example Values: true

**include_my_retweet**
optional

When set to either true, t or 1, any Tweets returned that have been retweeted by the authenticating user will include an additional current_user_retweet node, containing the ID of the source status for the retweet.

Example Values: true

**include_entities**
optional

The entities node will be disincluded when set to false.

Example Values: false

Resource Information

| Rate Limited? | Yes |
| Requests per rate limit window | 180/user 180/app |
| Authentication | Required |
| Response Formats | json |
| HTTP Methods | GET |
| Resource family | statuses |
| Response Object | Tweets |
| API Version | v1.1 |

OAuth tool

Please Sign in with your Twitter account in order to use the OAuth tool.

Tags

- finding tweets (49)
Twitter API v1.1

- 현재 104개의 API가 제공되고 있다
- https://dev.twitter.com/docs/api/1.1

GET status/show/:id
- https://dev.twitter.com/docs/api/1.1/get/statuses/show/%3Aid
- https://api.twitter.com/1.1/statuses/show.json?id=210462857140252672

Example Request

```json
GET https://api.twitter.com/1.1/statuses/show.json?id=210462857140252672
{
    "coordinates": null,
    "favorited": false,
    "truncated": false,
    "created_at": "Wed Jun 06 20:07:10 +0000 2012",
    "id_str": "210462857140252672",
    "entities": {
        "urls": [
            {
                "expanded_url": "https://dev.twitter.com/terms/display-guidelines",
                "url": "https://t.co/Ed4omjYs",
                "indices": [76, 97]
            }
        ]
    }
}
```
Twitter API v1.1

- GET status/show/:id
  - RESTful API
  - https://dev.twitter.com/docs/api/1.1/get/statuses/show/%3Aid
  - https://api.twitter.com/1.1/statuses/show/210462857140252672.json

- Twitter.com
  - https://twitter.com/twitterapi/status/210462857140252672
Creating a Twitter API connection

- RESTful API
- Python Library
  - Windows / Mac / Linux: pip install twitter
- Twitter는 OAuth (Open Authentication) protocol을 사용
- Facebook / Twitter 계정을 사용해 다른 어플리케이션에서 로그인

http://tutorials.jenkov.com/oauth2/index.html
HCCLab

Application settings
Keep the “API secret” a secret. This key should never be human-readable in your application.

API key: 8VpKo1GLyFjWFRJ4ODXQ
API secret: j1NkG56YZDyiO65Cp9gFk7wMLX3m09qFgQWuZjhnS4
Access level: Read-only (modify app permissions)
Owner: billsh
Owner ID: 15934568

Application actions
Regenerate API keys  Change App Permissions

Your access token
You haven’t authorized this application for your own account yet.

By creating your access token here, you will have everything you need to make API calls right away. The access token generated will be assigned your application’s current permission level.

Token actions
Create my access token
## Application settings

*Keep the "API secret" a secret. This key should never be human-readable in your application.*

<table>
<thead>
<tr>
<th>API key</th>
<th>8VpK01GJ</th>
</tr>
</thead>
<tbody>
<tr>
<td>API secret</td>
<td>J1Nk6gYz</td>
</tr>
<tr>
<td>Access level</td>
<td>Read-only (modify app permissions)</td>
</tr>
<tr>
<td>Owner</td>
<td>billsuh</td>
</tr>
<tr>
<td>Owner ID</td>
<td>15934558</td>
</tr>
</tbody>
</table>

## Application actions

- Regenerate API keys
- Change App Permissions

## Your access token

*This is an access token, not your API key. You should never share it with anyone.*

<table>
<thead>
<tr>
<th>Access token</th>
<th>15934558-0th</th>
</tr>
</thead>
<tbody>
<tr>
<td>Access token secret</td>
<td>69l57M9nZfm</td>
</tr>
<tr>
<td>Access level</td>
<td>Read-only</td>
</tr>
<tr>
<td>Owner</td>
<td>billsuh</td>
</tr>
<tr>
<td>Owner ID</td>
<td>15934558</td>
</tr>
</tbody>
</table>
First Twitter connection - command line

- Command line
  
  ```python
  pip install twitter
  python
  ```

- (tutorial) 프롬프트에서 다음을 작성

  ```python
  import twitter
  CONSUMER_KEY = ''
  CONSUMER_SECRET = ''
  OAUTH_TOKEN = ''
  OAUTH_TOKEN_SECRET = ''
  
  auth = twitter.oauth.OAuth(OAUTH_TOKEN, OAUTH_TOKEN_SECRET, CONSUMER_KEY, CONSUMER_SECRET)
  twitter_api = twitter.Twitter(auth=auth)
  print(twitter_api)
  ```
Exploring Trending Topics

- https://dev.twitter.com/docs/api/1.1/get/trends/place

GET trends/place

Updated on Thu, 2013-03-07 10:40

Returns the top 10 trending topics for a specific WOEID, if trending information is available for it.

The response is an array of "trend" objects that encode the name of the trending topic, the query parameter that can be used to search for the topic on Twitter Search, and the Twitter Search URL.

This information is cached for 5 minutes. Requesting more frequently than that will not return any more data, and will count against your rate limit usage.

Resource URL
https://api.twitter.com/1.1/trends/place.json

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
<td>The Yahoo! Where On Earth ID of the location to return trending information for. Global information is available by using 1 as the WOEID.</td>
</tr>
</tbody>
</table>

Example Values: 1
Trending Topics and JSON

```
WORLD_WOE_ID = 1
US_WOE_ID = 23424977

world_trends = twitter_api.trends.place(_id=WORLD_WOE_ID)
us_trends = twitter_api.trends.place(_id=US_WOE_ID)

print world_trends
print
print us_trends

import json

print json.dumps(world_trends, indent=1)
print
print json.dumps(us_trends, indent=1)
```

Predefined ID

Fetch data via Twitter API

Parse JSON & print
JSON format

[]  list

{}  object block

name: value
Computing the intersection

```python
>>> for trend in world_trends[0]["trends"]:  
    ... print trend

world_trends_set = set([trend["name"]  
                        for trend in world_trends[0]["trends"]])

us_trends_set = set([trend["name"]  
                     for trend in us_trends[0]["trends"]])

common_trends = world_trends_set.intersection(us_trends_set)

print common_trends
```

**for loop in Python**

*Collect set names*

*Get the intersection*

*Print the intersection set*
Searching for Tweets

- https://dev.twitter.com/docs/api/1.1/get/search/tweets
- https://api.twitter.com/1.1/search/tweets.json?q=exo&since_id=24012619984051000&max_id=250126199840518145&count=100

GET search/tweets

Returns a collection of relevant Tweets matching a specified query.

Please note that Twitter's search service and, by extension, the Search API is not meant to be an exhaustive source of Tweets. Not all Tweets will be indexed or made available via the search interface.

In API v1.1, the response format of the Search API has been improved to return Tweet objects more similar to the objects you'll find across the REST API and platform. You may need to tolerate some inconsistencies and variance in perspectival values (fields that pertain to the perspective of the authenticating user) and embedded user objects.

To learn how to use Twitter Search® effectively, consult our guide to Using the Twitter Search API. See Working with Timelines to learn best practices for navigating results by since_id and max_id.

Resource URL
https://api.twitter.com/1.1/search/tweets.json

Parameters
- q (required)
  A UTF-8, URL-encoded search query of 1,000 characters maximum, including operators. Queries may additionally be limited by complexity.

Related open issues
- Search API: Count param does not work when result type is set to mixed

Resource Information
- Rate Limited? Yes
- Requests per rate limit window
  - 180/user
  - 450/app
- Authentication Required
- Response Formats json
- HTTP Methods GET
Python convention for lists of arguments

def f(a, b, c, d=None, e=None):
    print a, b, c, d, e

f(1, 2, 3)              # 1 2 3 None None
f(1, 3, 3, d=4)         # 1 2 3 4 None
f(1, 2, 3, d=4, e=5)    # 1 2 3 4 5

args = [1,2,3]
kwvars = {'d' : 4, 'e' : 5}

f(*args, **kwvars)      # 1 2 3 4 5
q = 'olympics'
#q = unicode('올림픽', 'euc-kr')  # Windows
#q = '올림픽'  # Mac or Linux

count = 100

search_results = twitter_api.search.tweets(q=q, count=count)

statuses = search_results['statuses']

for _ in range(5):
    print "Length of statuses", len(statuses)
    try:
        next_results = search_results['search_metadata']['next_results']
    except KeyError, e:  # No more results when next_results doesn't exist
        print search_results['search_metadata']
        break

    kwargs = dict([kv.split('=') for kv in next_results[1:].split('&')])

    search_results = twitter_api.search.tweets(**kwargs)
    statuses += search_results['statuses']

print json.dumps(statuses[0], indent=1)
```python
>>> q = 'olympics'
>>> count = 100

>>> search_results = twitter_api.search.tweets(q=q, count=count)

>>> statuses = search_results['statuses']

>>> for _ in range(5):
...    print "Length of statuses", len(statuses)
...    try:
...        next_results = search_results['search_metadata']['next_results']
...    except KeyError, e:  # No more results when next_results doesn't exist
...        print search_results['search_metadata']
...        break

...    kwargs = dict([kv.split('=') for kv in next_results[1:].split("&")])

...    search_results = twitter_api.search.tweets(**kwargs)
...    statuses += search_results['statuses']

Length of statuses 100
Length of statuses 200
Length of statuses 300
Length of statuses 400
Length of statuses 500
```
What’s in the Tweet JSON?

- "text": "Sochi Winter Olympics venues meet with enthusiasm - Two... http://t.co/fnTYQ4k6su
#FishtOlympicStadium #NicolHle #SochiWinterOlympics",
- "in_reply_to_status_id": null,
- "id": 432492176958693376,
- "favorite_count": 0,
- "source": "<a href="http://www.ajaymatharu.com/" rel="nofollow">Tweet Old Post</a>",
- "retweeted": false,
- "coordinates": null,
- "geo": null,
- "lang": "en",
- "created_at": "Sun Feb 09 12:32:24 +0000 2014",
- "user"
  - "screen_name": "LasAngelesTimez",
  - "followers_count": 186,
  - "id_str": "1375472371",
  - "statuses_count": 116624,
  - "description": "#LosAngeles #news #gangsta #Ebonics #Urban #hiphop
#followback",
  - "friends_count": 24,
  - "location": "Las Angeles",
  - "created_at": "Tue Apr 23 20:49:26 +0000 2013",
  - "contributors_enabled": false,
  - "time_zone": "Arizona"
Analyzing the 140 Characters

• JSON file에서 content에 관련된 정보를 수집
• text
  – Sochi Winter Olympics venues meet with enthusiasm - Two... http://t.co/fnTYQ4k6su #FishtOlympicStadium #NicoIhle #SochiWinterOlympics
• screen_name
  – user → screen_name
• “entities”
  – hashtag
    ["indices": [84,104], "text": "FishtOlympicStadium"],
    ["indices": [105,114], "text": "NicoIhle"]
  – URL
    ["url": "http://t.co/fnTYQ4k6su", "indices": [60,82],
    "expanded_url": "http://j.mp/1eyJJvt",
    "display_url": "j.mp/1eyJJvt"]
Examining Patterns in Retweets

```python
status_texts = [ status['text']
                 for status in statuses ]

screen_names = [ status['user']['screen_name']
                 for status in statuses ]

hashtags = [ hashtag['text']
             for status in statuses
             for hashtag in status['entities']['hashtags'] ]

# Compute a collection of all words from all tweets
words = [ w
          for t in status_texts
          for w in t.split() ]

print json.dumps(status_texts[0:5], indent=1)
print json.dumps(screen_names[0:5], indent=1)
print json.dumps(hashtags[0:5], indent=1)
print json.dumps(words[0:5], indent=1)
```
Creating basic frequency distribution from the words

from collections import Counter

for item in [words, screen_names, hashtags]:
    c = Counter(item)
    print c.most_common()[:10] # top 10
    print

Calculate the top ten most frequent words, screen names, and hashtags
Using prettytable to display tuples

- **Command line**에서
  pip install prettytable

- **Python**내에서 다음을 작성

  ```python
  from prettytable import PrettyTable

  for label, data in (('Word', words),
                      ('Screen Name', screen_names),
                      ('Hashtag', hashtags)):
    pt = PrettyTable(field_names=[label, 'Count'])
    c = Counter(data)
    print(label, len(data))
    for kv in c.most_common()[:10]:
      pt.add_row(kv)
    pt.align[label], pt.align['Count'] = 'l', 'r'  # Set column alignment
    print(pt)
  ```

  *Add the top ten most frequent words into the prettytable*
Retweet information in JSON format

- Tweet의 모든 정보가 포함되어 있다!
- "text": "RT @jennyjonessnow: Aaahhhhhh !! Did I just get bronze at the friggin Olympics!! #ridiculous #welldonejamieandenni http://t.co/Fm9jYG5NwL"
- "id": 432510152353546240
- "created_at": "Sun Feb 09 13:43:50 +0000 2014",
- "user"
  - "screen_name": "JakerKirkby"
- "retweet_count": 1767
- "retweeted_status"
  - "id": 432494297142018048
  - "created_at": "Sun Feb 09 12:40:50 +0000 2014"
  - "user"
    - "screen_name": "jennyjonessnow"
Finding the most popular retweets

retweets_dup = [
    # Store out a tuple of these three values ...
    (status['retweet_count'],
     status['retweeted_status']['user']['screen_name'],
     status['text'])

    # ... for each status ...
    for status in statuses
    # ... so long as the status meets this condition.
    if status.has_key('retweeted_status')
]

retweets = list(set(retweets_dup))

# Slice off the first 5 from the sorted results and display each item in the tuple

pt = PrettyTable(field_names=['Count', 'Screen Name', 'Text'])
for row in sorted(retweets, reverse=True)[:5]:
    pt.add_row(row)

pt.max_width['Text'] = 50
pt.align = 'l'
print pt

Add the top five most retweeted tweets Into the prettytable
Word Cloud 만들기
Natural Language Processing

• EOS detection
  – 문단을 문장단위로 자르기
• Tokenization
  – 단어 형태로 자르기
• Part-of-speech (POS) tagging
  – 단어가 명사인지, 동사, 천지사인지 등 품사를 확인하기
• Chunking
  – 몇 단어를 모아 한 단위로 만들기 (e.g. Mr. Green ➔ 한 단위)
• Extraction
  – 사용할 수 있는 형태로 추출
End of Sentence Detection

txt = "Mr. Green killed Colonel Mustard in the study with the candlestick. Mr. Green is not a very nice fellow."
print txt.split(".")

import nltk
nltk.download('punkt')
sentences = nltk.tokenize.sent_tokenize(txt)
print sentences
Tokenization & Part-Of-Speech Tagging

tokens = [nltk.tokenize.word_tokenize(s) for s in sentences]
print tokens

nltk.download('maxent_treebank_pos_tagger')
pos_tagged_tokens = [nltk.pos_tag(t) for t in tokens]
print pos_tagged_tokens
_CHUNKING – ENTITY EXTRACTION

```python
nltk.download('maxent_ne_chunker')
nltk.download('words')

ne_chunks = nltk.batch_ne_chunk(pos_tagged_tokens)
print(ne_chunks[0].pprint())
```
Stemming, Stopwords & Extraction

tokens = []
for s in sentences:
    tokens += nltk.tokenize.word_tokenize(s)

from nltk.corpus import stopwords
stop_words = nltk.corpus.stopwords.words('english') + [
    '\.', ',', '\', 's', '?', ')', '(', ',', 'n',
    '\re', '\', ',', '}', '{', 'u—', 'rt', 'http', 't', 'co', ]

from nltk.stem import PorterStemmer
stemmer=PorterStemmer()
stemmed = []
for token in tokens:
    if token.lower() in stop_words:
        continue
    stemmed.append(stemmer.stem(token.lower()))

>>> tokens
['Mr.', 'Green', 'killed', 'Colonel', 'Mustard', 'in', 'the', 'study', 'with', 'the', 'candlestick', 'Mr.', 'Green', 'is', 'not', 'a', 'very', 'nice', 'rel low', '.']
>>> stemmed
['mr.', 'green', 'kill', 'colonel', 'mustard', 'study', 'candlestick', 'mr.', 'g reen', 'nice', 'fellow']
Word Cloud Summarization

status_texts = [ status['text']
    for status in statuses ]

tokens = []
for s in status_texts:
    tokens += nltk.tokenize.word_tokenize(s.lower())

from nltk.corpus import stopwords
stop = stopwords.words('english')
stop_words = nltk.corpus.stopwords.words('english') + [
    ',', '.', '-r', '\s', '?', '!', ')', '(',
    '\re', '\''', '.-', '}', '{', u'—', 'rt', 'http', 't', 'co', '@', '#',
]

from nltk.stem import PorterStemmer
stemmer=PorterStemmer()

stemmed = []
for token in tokens:
    if '/t.co/' in token or token is '' or token in stop_words:
        continue
    stemmed.append(stemmer.stem(token))
f = file('words.txt', 'w')
for word in stemmed:
    try:
        f.write(word.encode('utf-8') + '\n')
    except UnicodeEncodeError, e:
        print 'Encoding error ' + word + '\n'

f.close()
Tagxedo - Creator

http://tagxedo.com/app.html 서비스 이용하여 Word Cloud 작성
How to Obtain Valuable Insights?

Data

Information

Insight

Advice
Human Computer Interaction

• Definition: the study, planning and design of the interaction between people and computers

• 사랑 + 컴퓨터 + 상호작용

• 사람과 컴퓨터 사이에 일어나는 여러가지 것에 대해 연구

• 전산학 + 심리학 + 디자인 + 산업공학

• UX (사용자 경험), Usability(사용성), Ubiquitous computing, UI, GUI, Human Factors, etc.
Dvorak vs. Qwerty Keyboard (2별식 vs. 3별식)

Uncanny Valley

Hands Free Cell Phones


사회적 현상: 더욱 복잡하다
Bystander Effect

Broken Window Theory

Dunbar’s Number

Image: http://www.blrk.net/cute-monkeys-photos/1/1280x720/
“A lot of times, people don’t know what they want until you show it to them.”
— Steve Jobs

Image: http://www.nilotpaldas.com/
Walmart circa 2009

Would you like Walmart aisles to be less cluttered?

사람에 대한 이해가 중요하다!
감사합니다!

서봉원
bongwon@snu.ac.kr