

# 정보교육을 위한 파이썬

정보 탐색

Version 0.0.9-d2

저자: Charles Severance  
번역: 이광춘, 한정수  
(xwmooc)

Copyright © 2009- Charles Severance.

Printing history:

**October 2013:** Major revision to Chapters 13 and 14 to switch to JSON and use OAuth.  
Added new chapter on Visualization.

**September 2013:** Published book on Amazon CreateSpace

**January 2010:** Published book using the University of Michigan Espresso Book machine.

**December 2009:** Major revision to chapters 2-10 from *Think Python: How to Think Like a Computer Scientist* and writing chapters 1 and 11-15 to produce *Python for Informatics: Exploring Information*

**June 2008:** Major revision, changed title to *Think Python: How to Think Like a Computer Scientist*.

**August 2007:** Major revision, changed title to *How to Think Like a (Python) Programmer*.

**April 2002:** First edition of *How to Think Like a Computer Scientist*.

This work is licensed under a Creative Commons Attribution-NonCommercial-ShareAlike 3.0 Unported License. This license is available at [creativecommons.org/licenses/by-nc-sa/3.0/](http://creativecommons.org/licenses/by-nc-sa/3.0/). You can see what the author considers commercial and non-commercial uses of this material as well as license exemptions in the Appendix titled Copyright Detail.

The L<sup>A</sup>T<sub>E</sub>X source for the *Think Python: How to Think Like a Computer Scientist* version of this book is available from <http://www.thinkpython.com>.

# Chapter 1

## 데이터 시각화

지금까지 파이썬 언어를 학습했고, 데이터를 다루기 위해서 파이썬, 네트워크, 그리고 데이터베이스를 어떻게 사용하는지 배웠다.

이번장에서는 데이터를 관리하고 시각화하기 위해서 이들 모두를 합친 완전한 세개의 응용프로그램을 살펴볼 것이다. 실제 문제를 해결하는데 시작할 수 있는 샘플 코드로 응용프로그램을 사용할 수도 있다.

각 응용프로그램은 ZIP 파일로 압축되어서 다운로드 받아 여러분의 컴퓨터에 압축을 풀고 실행한다.

### 1.1 지리정보 데이터로 구글맵 생성하기

이번 프로젝트에서 구글 지오코딩 API를 사용해서 사용자가 입력한 대학교 이름의 지리 정보를 정리하고나서 구글 지도에 데이터를 표시한다.



시작하기 위해서 다음 url에서 응용프로그램을 다운로드한다.

[www.py4inf.com/code/geodata.zip](http://www.py4inf.com/code/geodata.zip)

해결할 첫번째 문제는 무료 구글 지오코딩 API가 하루에 요청횟수에 제한이 있다는 것이다. 그래서, 만약 많은 데이터가 있다면, 여러번 중지하고 다시 시작하는 프로세스가 필요하다. 그래서, 문제를 두 단계로 나누었다.

첫번째 단계에서, **where.data** 파일에 ”설문(survey)” 데이터를 받아서, 한번에 한줄씩 읽고 구글에서 지리정보를 가져오고 **geodata.sqlite** 데이터베이스에 저장한다. 각 사용자가 입력한 위치에 대한 지오코딩 API를 사용하기 전에, 특정 입력 라인에 데이터가 있는지를 알기 위해서 간단하게 확인한다. 데이터베이스는 지오 코딩 데이터의 로컬 ”캐쉬(cache)”처럼 동작해서 두번 동일한 데이터에 대해서는 구글에 요청하지 않도록 한다.

**geodata.sqlite** 파일을 제거함으로써 언제라도 프로세스를 다시 시작할 수 있다.

**geoload.py** 프로그램을 실행한다. **where.data**에 입력 라인을 읽고, 각 라인에 대해서 데이터베이스에 이미 존재하는지를 확인한다. 만약 위치에 대한 데이터가 없다면, 지오코딩 API를 호출해서 데이터를 가져오고 데이터베이스에 저장한다.

데이터베이스에 약간의 데이터가 이미 존재한 후에 샘플로 실행한 결과가 다음에 있다.

```
Found in database Northeastern University
Found in database University of Hong Kong, ...
Found in database Technion
Found in database Viswakarma Institute, Pune, India
Found in database UMD
Found in database Tufts University
```

```
Resolving Monash University
Retrieving http://maps.googleapis.com/maps/api/
    geocode/json?sensor=false&address=Monash+University
Retrieved 2063 characters {   "results" : [
{u'status': u'OK', u'results': ... }
```

```
Resolving Kokshetau Institute of Economics and Management
Retrieving http://maps.googleapis.com/maps/api/
    geocode/json?sensor=false&address=Kokshetau+Inst ...
Retrieved 1749 characters {   "results" : [
{u'status': u'OK', u'results': ... }
...
```

첫 5 지점은 이미 데이터베이스에 있으므로 건너뛴다. 프로그램은 가져오지 못한 위치 정보를 찾아 스캔하고 정보를 가져온다.

**geoload.py**는 언제라도 멈출 수 있다. 매번 실행에 대해서 지오코딩 API 호출의 횟수를 제한하기 위한 카운터가 있다. **where.data**가 수백개의 데이터 항목만 있다면, 하루 사용량 한계에 부딪치지 않을 것이지만, 만약 더 많은 데이터가 있다면, 입력데이터에 대한 모든 지리정보 데이터를 가진 데이터베이스를 만들기 위해서 몇일에 걸쳐서 여러번 실행할지도 모른다.

데이터를 **geodata.sqlite**에 적재하면, **geodump.py** 프로그램을 사용하여 데이터를 시각화할 수 있다. 프로그램이 데이터베이스를 일고 위치, 위도, 경도를 실행 가능한 자바스크립트 코드로 **where.js**에 쓴다.

**geodump.py** 프로그램 실행결과는 다음과 같다.

```
Northeastern University, ... Boston, MA 02115, USA 42.3396998 -71.08975
Bradley University, 1501 ... Peoria, IL 61625, USA 40.6963857 -89.6160811
...
Technion, Viazman 87, Kesalsaba, 32000, Israel 32.7775 35.0216667
Monash University Clayton ... VIC 3800, Australia -37.9152113 145.134682
Kokshetau, Kazakhstan 53.2833333 69.3833333
...
12 records written to where.js
Open where.html to view the data in a browser
```

**where.html** 파일은 HTML과 자바스크립트로 구성되어서 구글 맵을 시각화한다. **where.js**에 가장 최신 데이터를 읽어서 시각화할 데이터로 사용한다. 다음에 **where.js** 파일 형식이 있다.

```
myData = [
[42.3396998, -71.08975, 'Northeastern Uni ... Boston, MA 02115'],
[40.6963857, -89.6160811, 'Bradley University, ... Peoria, IL 61625, USA'],
[32.7775, 35.0216667, 'Technion, Viazman 87, Kesalsaba, 32000, Israel'],
...
];
```

리스트의 리스트를 담고 있는 자바스크립트다. 자바스크립트 리스트 상수에 대한 구분은 파이썬과 매우 유사하여, 구문이 여러분에게 매우 친숙할 것이다.

위치를 보기 위해서 브라워저에 **where.html**을 연다. 각 맵 편을 여기저기 돌아다니면서 지오코딩 API가 사용자가 입력한 것에 대해서 반환한 위치를 찾는다. **where.html** 파일을 열었을 때 어떤 데이터도 볼 수 없다면, 자바스크립트나 브라우저의 개발자 콘솔을 확인한다.

## 1.2 네트워크와 상호 연결 시각화

이번 응용프로그램에서 약간의 검색엔진 기능을 수행한다. 먼저 웹의 일부분을 스파이더링하고 어느 페이지가 가장 많이 연결되어 결정하기 위해서 간략한 구글 페이지 랭크 알고리즘을 실행하고 스파이더링한 웹의 작은 부분의 연결성과 페이지 랭크를 시각화한다. 시각화 산출물을 만들기 위해서 D3 자바스크립트 시각화 라이브러리 <http://d3js.org/>를 사용한다.

응용프로그램을 다음 url에서 다운로드 받아 압축을 풀다.

[www.py4inf.com/code/pagerank.zip](http://www.py4inf.com/code/pagerank.zip)



첫 프로그램(**spider.py**)은 웹사이트를 크롤(crawl)하고 일련의 페이지를 뽑아내서 데이터베이스(**spider.sqlite**)에 넣어서 페이지들간의 링크를 기록한다. 언제라도 **spider.sqlite** 파일을 제거하고 **spider.py**를 재실행함으로써 프로세스를 다시 시작할 수 있다.

```
Enter web url or enter: http://www.dr-chuck.com/
['http://www.dr-chuck.com']
How many pages:2
1 http://www.dr-chuck.com/ 12
2 http://www.dr-chuck.com/csev-blog/ 57
How many pages:
```

샘플 실행결과, 웹사이트를 크롤하고 두개의 페이지를 가져왔다. 프로그램을 다시 실행하고 좀더 많은 웹페이지를 크롤하게 한다면, 데이터베이스에 이미 등록된 어떤 페이지도 다시 크롤하지는 않는다. 재시작할 때 무작위로 크롤하지 않은 페이지로 가서 그곳에서 시작한다. 그래서 **spider.py**을 연속적으로 실행은 추가적이다.

```
Enter web url or enter: http://www.dr-chuck.com/
['http://www.dr-chuck.com']
How many pages:3
3 http://www.dr-chuck.com/csev-blog 57
4 http://www.dr-chuck.com/dr-chuck/resume/speaking.htm 1
5 http://www.dr-chuck.com/dr-chuck/resume/index.htm 13
How many pages:
```

같은 데이터베이스에서 다중 시작 지점을 가질 수 있다. 프로그램 내부에서 ”웹(web)”이라고 부른다. 스파이더는 다음 웹 페이지로 모든 웹사이트에서 방문하지 않는 링크 중에 무작위로 선택한다.

**spider.sqlite** 파일의 내용을 살펴보고자 한다면, 다음과 같이 **spdumpp.py**을 실행한다.

```
(5, None, 1.0, 3, u'http://www.dr-chuck.com/csev-blog')
(3, None, 1.0, 4, u'http://www.dr-chuck.com/dr-chuck/resume/speaking.htm')
(1, None, 1.0, 2, u'http://www.dr-chuck.com/csev-blog/')
(1, None, 1.0, 5, u'http://www.dr-chuck.com/dr-chuck/resume/index.htm')
4 rows.
```

인입 링크 수, 이전 페이지 랭크, 신규 페이지 랭크, 페이지 id, 페이지 url을 보여준다. **spdump.py** 프로그램은 최소한 하나의 인입 링크가 있는 페이지만을 보여준다.

데이터베이스에 몇개의 페이지가 있다면, **sprank.py** 프로그램을 실행하여 페이지의 페이지 랭크를 실행한다. 단순하게 얼마나 많은 페이지랭크 반복을 수행할지 지정한다.

```
How many iterations:2
1 0.546848992536
2 0.226714939664
[(1, 0.559), (2, 0.659), (3, 0.985), (4, 2.135), (5, 0.659)]
```

데이터베이스를 다시 열어서 페이지 랭크가 갱신되었는지 확인한다.

```
(5, 1.0, 0.985, 3, u'http://www.dr-chuck.com/csev-blog')
(3, 1.0, 2.135, 4, u'http://www.dr-chuck.com/dr-chuck/resume/speaking.htm')
(1, 1.0, 0.659, 2, u'http://www.dr-chuck.com/csev-blog/')
(1, 1.0, 0.659, 5, u'http://www.dr-chuck.com/dr-chuck/resume/index.htm')
4 rows.
```

**sprank.py**를 원하는 만큼 실행하고, 실행할 때마다 페이지 랭크를 정교화한다. **sprank.py**을 몇번 실행하고나서 **spider.py**으로 좀더 많은 페이지를 스파이더링하고 나서 **sprank.py**을 실행하여 페이지 랭크 값을 다시 수렴하여 갱신한다. 검색엔진은 일반적으로 동시에 크롤링과 랭킹 프로그램을 실행한다.

웹페이지를 다시 스파이더링하지 않고, 페이지 랭크 계산을 재시작하고자 한다면, **spreset.py**을 사용하고 나서 **sprank.py**를 다시 시작한다.

```
How many iterations:50
1 0.546848992536
2 0.226714939664
3 0.0659516187242
4 0.0244199333
5 0.0102096489546
6 0.00610244329379
...
42 0.000109076928206
43 9.91987599002e-05
44 9.02151706798e-05
45 8.20451504471e-05
46 7.46150183837e-05
47 6.7857770908e-05
48 6.17124694224e-05
49 5.61236959327e-05
50 5.10410499467e-05
[(512, 0.0296), (1, 12.79), (2, 28.93), (3, 6.808), (4, 13.46)]
```

페이지 랭크 알고리즘을 매번 반복하면, 페이지마다 페이지 랭크의 평균 변화를 출력한다. 초기에 네트워크는 매우 불균형 상태여서 각각의 페이지 랭크 값은 반복할 때마다 요동친다. 하지만, 짧은 몇번의 반복에서 페이지 랭크는 수렴한다. **prank.py**을 충분히 오랜동안 실행해서 페이지 랭크 값이 수렴하게 한다.

페이지 랭크에 대해서 현재 최상위 페이지를 시각화하고자 한다면, **spjson.py** 을 실행하여 데이터베이스를 읽고 웹브라우저에서 볼 수 있는 JSON 형식으로 가장 많이 연결된 페이지에 대해서 데이터를 쓴다.

```
Creating JSON output on spider.json...
How many nodes? 30
Open force.html in a browser to view the visualization
```

웹브라우저에 **force.html** 파일을 열어서 작업한 데이터를 볼 수 있다. 자동적인 노드와 링크의 레이아웃을 보여준다. 임의의 노드를 클릭하고 끌수 있고, 노드를 더블 클릭해서 노드로 표현된 URL을 확인할 수 있다.

만약 다른 유ти리티를 다시 실행하고 하면, **spjson.py**을 다시 실행하고 브라우저의 새로 고치기를 눌러서 **spider.json**에서 새로운 데이터를 가져온다.

### 1.3 전자우편 데이터 시각화

책을 이 지점까지 읽어오면서 **mbox-short.txt**과 **mbox.txt** 파일에 매우 친숙해졌을 것이다. 이제는 전자우편 데이터의 분석을 다음 수준으로 가져갈 때다.

실무에서, 때때로 서버에서 전자우편 데이터를 가져오는 것은 시간이 많이 걸리고, 가져온 데이터는 오류가 많고, 일관되지 못하고, 많은 보정과 정비가 필요하다. 이번 장에서, 지금까지 가장 복잡한 응용프로그램을 가지고 거의 1GB 데이터를 추출하여 시각화 작업을 한다.



응용프로그램을 다음 url에서 다운로드한다.

[www.py4inf.com/code/gmane.zip](http://www.py4inf.com/code/gmane.zip)

[www.gmane.org](http://www.gmane.org)의 무료 전자우편 보관 서비스의 데이터를 사용한다. 전자우편 활동에 대한 멋진 검색 가능한 보관 서비스를 제공하기 때문에, 오픈 소스 프로젝트에서는 매우 인기가 높다. API를 통한 데이터 접근에 대해서 매우 자유로운 정책을 유지한다. 사용량에 대해서 제한은 없지만, 서비스에 너무 많은 부하를

주지 말고 필요한 데이터만 가져가도록 요청한다. gmane의 사용조건에 대해서는 다음 페이지를 참조한다.

<http://gmane.org/export.php>

서비스 접근에 자연을 추가하거나 좀더 오랜 기간에 걸쳐서 장시간이 소요되는 작업을 분산함으로써 *gmane.org* 데이터를 책임감있게 사용하는 것은 매우 중요하다. 무료 서비스를 오용하지 말고, 다른 사용자를 위해서 서비스를 파괴하지 말아주세요.

이 프로그램을 사용하여 Sakai 전자우편 주소 데이터를 스파이더링하면, 거의 1 GB의 데이터를 생성하고 몇일에 걸쳐서 수차례 실행을 해야한다. ZIP 압축 파일 내부에 **README.txt**이 어떻게 대부분의 Sakai 전자우편 코퍼스(corpus)를 다운로드할 수 있는지에 대한 안내서를 하고 있다. 그래서 단지 프로그램을 실행하기 위해서 5일 동안 스파이더링을 할 필요는 없다. 이미 스파이더링된 콘텐츠를 다운로드하면, 좀더 최근의 메시지를 가져오기 위해서 스파이더링 프로세스를 여전히 실행해야 한다.

첫 단계는 gmane 저장소를 스파이더링하는 것이다. 기본 URL은 **gmane.py** 파일과 Sakai 개발자 리스트에 하드코딩 되었다. 기본 url을 변경함으로써 또 다른 저장소를 스파이더링 할 수 있다. 기본 url을 변경하는 경우 **content.sqlite** 파일을 필히 삭제하세요.

**gmane.py** 파일은 책임감있는 캐쉬 스파이더로서 작동하고, 천천히 실행되며 1초에 한개의 전자우편 메시지를 가져와서 gmane에 의해서 작동못하게 되는 것을 피한다. 데이터베이식 모든 데이터를 저장하고 필요하면 자주 중단하고 다시 시작할 수 있다. 모든 데이터를 가져오는데 몇 시간이 걸릴 수 있다. 그래서 여러번 재시작하는 것이 필요하다.

Sakai 개발자 리스트의 마지막 5개 메시지를 가져오는 **gmane.py**을 실행한 결과다.

```
How many messages:10
http://download.gmane.org/gmane.comp.cms.sakai.devel/51410/51411 9460
    nealcaidin@sakaifoundation.org 2013-04-05 re: [building ...
http://download.gmane.org/gmane.comp.cms.sakai.devel/51411/51412 3379
    samuelgutierrezjimenez@gmail.com 2013-04-06 re: [building ...
http://download.gmane.org/gmane.comp.cms.sakai.devel/51412/51413 9903
    da1@vt.edu 2013-04-05 [building sakai] melete 2.9 oracle ...
http://download.gmane.org/gmane.comp.cms.sakai.devel/51413/51414 349265
    m.shedid@elraed-it.com 2013-04-07 [building sakai] ...
http://download.gmane.org/gmane.comp.cms.sakai.devel/51414/51415 3481
    samuelgutierrezjimenez@gmail.com 2013-04-07 re: ...
http://download.gmane.org/gmane.comp.cms.sakai.devel/51415/51416 0
```

Does not start with From

프로그램은 1번부터 이미 스파이더링되지 않는 첫 메시지까지 **content.sqlite**을 스캔하고 그 메시지에서 스파이더링을 시작한다. 원하는 메시지 숫자를 스파이더링할 때까지 계속하거나 잘못된 형식의 메시지가 나오는 페이지에 도달할 때까지 스파이더링을 계속한다.

때때로 gmane.org이 메시지를 잃어버린다. 아마도 관리자가 메시지를 삭제하거나 아마도 분실했을 것이다. 만약 스파이더가 멈추고 잃어버린 메시지에 부딪치면 SQLite 매니저로 가서 다른 모든 행은 공백으로 구성되고 잃어버린 id를 가진 행을 추가하고 **gmane.py**을 재시작한다. 이렇게 함으로써 스파이더링 프로세스를 다시 시작하고 계속 진행한다. 빈 메시지는 다음번 프로세스에서는 무시된다.

모든 메시지를 스파이더링하고 **content.sqlite** 파일에 넣게 되면, **gmane.py**을 다시 시작해서 리스트에 보내지게 되면 새로운 메시지를 얻는 것은 멋진 기능이다.

**content.sqlite** 데이터는 원데이터 형식이고 비효율적인 데이터 모델과 압축이 되어 있지 않다. 의도적으로 그렇게 함으로써 SQLite 매니저가 스파이더링 프로세스에서 문제를 디버그할 수 있도록 **content.sqlite** 파일을 직접 살펴볼 수 있다. 매우 느린 상태에서 데이터베이스에 질의를 실행하는 것은 좋은 생각은 아니다.

두번째 프로세스는 **gmodel.py** 프로그램을 실행하는 것이다. **content.sqlite**에서 다듬지 않은 원데이터를 읽어서 **index.sqlite** 파일에 깨끗하게 잘 모델링된 형식의 데이터로 저장한다. **index.sqlite** 파일은 **content.sqlite** 보다 10배이상 크기가 적다. 왜냐하면 헤더와 본문부문 텍스트를 압축하기 때문이다.

**gmodel.py**을 실행할 때마다, **index.sqlite**을 삭제하고 다시 생성해서, 매개변수를 조정하고 데이터 정비 프로세스를 약간 바꾸기 위해서 **content.sqlite**의 매핑 테이블을 편집할 수 있게 한다. 다음은 **gmodel.py**을 샘플로 실행시킨 것이다. 매번 250개의 전자우편 메시지가 처리될 때마다 한 라인을 출력해서 거의 1GB 전자우편 데이터를 처리하는 동안 진행사항을 확인할 수 있다.

```
Loaded allsenders 1588 and mapping 28 dns mapping 1
1 2005-12-08T23:34:30-06:00 ggolden22@mac.com
251 2005-12-22T10:03:20-08:00 tpamsler@ucdavis.edu
501 2006-01-12T11:17:34-05:00 lance@indiana.edu
751 2006-01-24T11:13:28-08:00 vrajgopalan@ucmerced.edu
...

```

**gmodel.py** 프로그램이 많은 데이터 정비 작업을 처리한다.

도메인 이름이 .com, .org, .edu, .net에 대해서는 2 단계로 다른 도메인 이름은 3 단계로 잘랐다. 그래서 si.umich.edu은 umich.edu이 되고, caret.cam.ac.uk은 cam.ac.uk이 된다. 또한 전자우편 주소는 소문자가 되게 하고, 다음과 같은 @gmane.org 주소는 메시지 코퍼스 어딘가 있는 실제 전자우편 주소와 매칭이 되면 실제 주소로 변환한다.

```
arwhyte-63aXycvo3TyHXe+LvDLADg@public.gmane.org
```

**content.sqlite** 데이터베이스를 살펴보면, 도메인 이름과 전자우편 주소가 시간에 따라 변경되는 개인 전자우편 주소를 매핑하는 테이블이 두개 있다. 예를 들어, Sakai 개발자 리스트 개발기간에 걸쳐 직업을 바꿈에 따라 Steve Githens 다음 전자우편 주소를 사용했다.

```
s-githens@northwestern.edu
sgithens@cam.ac.uk
swgithen@mtu.edu
```

**content.sqlite**의 매핑(Mapping) 테이블에 두 개의 항목을 추가해서 **gmodel.py** 프로그램이 3개 주소를 하나로 매핑한다.

```
s-githens@northwestern.edu -> swgithen@mtu.edu
sgithens@cam.ac.uk -> swgithen@mtu.edu
```

다중 DNS 이름을 하나의 DNS 이름으로 매핑하려면 **DNSMapping** 테이블에 비슷한 항목을 생성할 수 있다. 다음 매핑이 Sakai 데이터에 추가된다.

```
iupui.edu -> indiana.edu
```

그래서, 다양한 인디애나 대학교 캠퍼스의 모든 계정이 함께 추적된다.

데이터를 볼 때마다, **gmodel.py**를 반복적으로 실행하고 데이터를 좀 더 깨끗하게 만들기 위해서 매핑을 추가한다. 작업이 마쳐지면, **index.sqlite**에 잘 인덱스된 버전의 전자우편 데이터가 있다. 이 파일이 자료 분석을 위해 사용되는 파일이다. 이 파일로 데이터분석은 정말 빠르게 수행된다.

첫, 가장 간단한 자료 분석은 ”누가 가장 많은 전자우편을 보냈는가?”와 ”어느 조직에서 가장 많은 전자우편을 보냈는가?”를 알아보는 것이다. **gbasic.py**를 사용해서 수행할 수 있다.

```
How many to dump? 5
Loaded messages= 51330 subjects= 25033 senders= 1584
```

```
Top 5 Email list participants
steve.swinsburg@gmail.com 2657
azeckoski@unicon.net 1742
ieb@tfd.co.uk 1591
csev@umich.edu 1304
david.horwitz@uct.ac.za 1184
```

```
Top 5 Email list organizations
gmail.com 7339
umich.edu 6243
uct.ac.za 2451
indiana.edu 2258
unicon.net 2055
```

**gmane.py**이나 혹은 **gmodel.py**과 비교하여 **gbasic.py**이 얼마나 빨리 실행되는지 주목하세요. 모두 동일한 데이터에서 작업을 수행하지만, **gbasic.py**은 **index.sqlite**에 저장된 압축되고 정규화된 데이터를 사용한다. 다를 데이터가 많다면, 이번과 같은 응용프로그램처럼 다중 단계 프로세스가 개발에는 약간 더 시간이 걸리지만, 데이터를 탐색하고 시각화할 때는 많이 시간을 절약해 준다.

**gword.py** 파일에 주제 라인의 단어 빈도를 간략히 시각화할 수 있다.

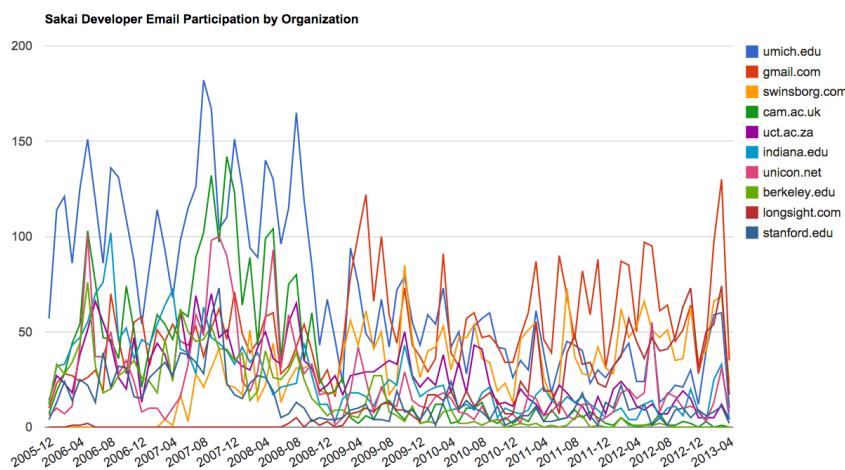
```
Range of counts: 33229 129
Output written to gword.js
```

이번 장의 처음의 것과 유사한 워드 클라우드를 생성하기 위해서 **gword.htm**를 사용하여 시각화할 **gword.js** 파일을 생성한다.

두번째 시각화는 **gline.py**으로 생성된다. 시간에 따른 조직별 전자우편 참여를 연산한다.

```
Loaded messages= 51330 subjects= 25033 senders= 1584
Top 10 Organizations
['gmail.com', 'umich.edu', 'uct.ac.za', 'indiana.edu',
'unicon.net', 'tfd.co.uk', 'berkeley.edu', 'longsight.com',
'stanford.edu', 'ox.ac.uk']
Output written to gline.js
```

출력값은 **gline.htm**을 사용하여 시각화된 **gline.js**에 쓰여진다.



상대적으로 복잡하고 정교한 응용프로그램으로 실제 데이터를 가져오고, 정비하고, 시각화하는 기능을 갖고 있다.