The Illusion of Privacy on Reddit

Renzo Lucioni and R.J. Aquino

Table of Contents

Abstract

The goal of this project was to explore privacy on Reddit, the popular social news and entertainment website. On Reddit, users register under a pseudonym, submit content primarily in the form of links, and make text comments on content submitted by others. We set out to harvest personally identifying information from a collection of 15 million comments posted to Reddit between August 28 and September 12. We succeeded in finding hundreds of Reddit users' Facebook profiles, Twitter handles, email addresses, phone numbers, and ZIP codes. This information allowed us to identify dozens of Redditors with a high degree of confidence. These results demonstrate that the notion of privacy on Reddit is an illusion for many users. Creating a truly anonymous environment on Reddit would require measures preventing users from posting this kind of information. However, such changes would be in direct opposition to the way the community on Reddit behaves.

Introduction

For many, Reddit is a website that users browse anonymously, often employing their usernames as pseudonyms. In their Privacy Policy, Reddit states that "we will only share your personal data with your consent." For these reasons, it is reasonable for Reddit users to expect that their personal information is indeed private, and that their "real world" data (e.g., name, date of birth, phone number) is not connected to their “digital” data (i.e., their Reddit username and content). We intend to demonstrate that a number of Reddit users have, either on purpose or accidentally, revealed identifying information, and that the information they have revealed can be tied back to a real identity. Many Reddit users publish phone numbers, email addresses, Twitter handles, and Facebook profiles. This information can sometimes be used to automatically generate an identity, and can often be combined with some additional sleuthing to reveal more personal information.

Our project consists of two parts. In the first part, we search a dump of 15 million Reddit comments searching for personally identifying information. In the second part, we attempt to identify specific users based on the personal information we found in their comments.

Methods

We used a dump of 15 million comments posted to Reddit between August 28 and September 12, cited in the References. Using the Python code we wrote below, we combed the comments for keywords and different kinds of personally identifying information. First, we had to perform imports and other setup.

In [1]:
```
# special IPython command to prepare the notebook for matplotlib
%matplotlib inline

# we might not use all of these
from fnmatch import fnmatch

import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import requests
import pickle
from pattern import web
import json
import re
import sys
import datetime as dt
import operator
import collections
import sys
import linecache

# set some nicer defaults for matplotlib
from matplotlib import rcParams

#these colors come from colorbrewer2.org. Each is an RGB triplet
dark2_colors = [(0.10588235294117647, 0.6196078431372549, 0.4666666666666667),
                (0.8509803921568627, 0.37254901960784315, 0.00784313725490196),
                (0.4588235294117647, 0.4392156862745098, 0.7019607843137254),
                (0.9058823529411765, 0.1607843137254902, 0.5411764705882353),
                (0.4, 0.6509803921568628, 0.11764705882352941),
                (0.9019607843137255, 0.6705882352941176, 0.00784313725490196),
                (0.6509803921568628, 0.4627450980392157, 0.11372549019607843),
                (0.4, 0.4, 0.4)]

rcParams['figure.figsize'] = (10, 6)
rcParams['figure.dpi'] = 150
rcParams['axes.color_cycle'] = dark2_colors
```

```
rcParams['lines.linewidth'] = 2
rcParams['axes.grid'] = False
rcParams['axes.facecolor'] = 'white'
rcParams['font.size'] = 14
rcParams['patch.edgecolor'] = 'none'

def remove_border(axes=None, top=False, right=False, left=True, bottom=True):
    """
    Minimize chartjunk by stripping out unnecessary plot borders and axis ticks

    The top/right/left/bottom keywords toggle whether the corresponding plot border is draw
n
    """
    ax = axes or plt.gca()
    ax.spines['top'].set_visible(top)
    ax.spines['right'].set_visible(right)
    ax.spines['left'].set_visible(left)
    ax.spines['bottom'].set_visible(bottom)

    #turn off all ticks
    ax.yaxis.set_ticks_position('none')
    ax.xaxis.set_ticks_position('none')

    #now re-enable visibles
    if top:
        ax.xaxis.tick_top()
    if bottom:
        ax.xaxis.tick_bottom()
    if left:
        ax.yaxis.tick_left()
    if right:
        ax.yaxis.tick_right()
```

This list of common English stop words defines words which should be ignored when performing a word count.

In [2]:
```
# common English stop words
stop_words = """
a
about
above
after
again
against
all
am
an
and
any
are
aren't
as
at
be
because
been
before
```

```
being
below
between
both
but
by
can't
cannot
could
couldn't
did
didn't
do
does
doesn't
doing
don't
down
during
each
few
for
from
further
had
hadn't
has
hasn't
have
haven't
having
he
he'd
he'll
he's
her
here
here's
hers
herself
him
himself
his
how
how's
i
i'd
i'll
i'm
i've
if
in
into
is
isn't
it
```

```
it's
its
itself
let's
me
more
most
mustn't
my
myself
no
nor
not
of
off
on
once
only
or
other
ought
our
ours
ourselves
out
over
own
same
shan't
she
she'd
she'll
she's
should
shouldn't
so
some
such
than
that
that's
the
their
theirs
them
themselves
then
there
there's
these
they
they'd
they'll
they're
they've
this
```

```
those
through
to
too
under
until
up
very
was
wasn't
we
we'd
we'll
we're
we've
were
weren't
what
what's
when
when's
where
where's
which
while
who
who's
whom
why
why's
with
won't
would
wouldn't
you
you'd
you'll
you're
you've
your
yours
yourself
yourselves
"""

stop_words = stop_words.split()
print stop_words
```

```
['a', 'about', 'above', 'after', 'again', 'against', 'all', 'am', 'an', 'and', 'any', 'are'
, "aren't", 'as', 'at', 'be', 'because', 'been', 'before', 'being', 'below', 'between', 'bo
th', 'but', 'by', "can't", 'cannot', 'could', "couldn't", 'did', "didn't", 'do', 'does', "d
oesn't", 'doing', "don't", 'down', 'during', 'each', 'few', 'for', 'from', 'further', 'had'
, "hadn't", 'has', "hasn't", 'have', "haven't", 'having', 'he', "he'd", "he'll", "he's", 'h

er', 'here', "here's", 'hers', 'herself', 'him', 'himself', 'his', 'how', "how's", 'i', "i'
d", "i'll", "i'm", "i've", 'if', 'in', 'into', 'is', "isn't", 'it', "it's", 'its', 'itself'
, "let's", 'me', 'more', 'most', "mustn't", 'my', 'myself', 'no', 'nor', 'not', 'of', 'off'
, 'on', 'once', 'only', 'or', 'other', 'ought', 'our', 'ours', 'ourselves', 'out', 'over',
'own', 'same', "shan't", 'she', "she'd", "she'll", "she's", 'should', "shouldn't", 'so', 's
ome', 'such', 'than', 'that', "that's", 'the', 'their', 'theirs', 'them', 'themselves', 'th
en', 'there', "there's", 'these', 'they', "they'd", "they'll", "they're", "they've", 'this'
, 'those', 'through', 'to', 'too', 'under', 'until', 'up', 'very', 'was', "wasn't", 'we', "
we'd", "we'll", "we're", "we've", 'were', "weren't", 'what', "what's", 'when', "when's", 'w
here', "where's", 'which', 'while', 'who', "who's", 'whom', 'why', "why's", 'with', "won't"
, 'would', "wouldn't", 'you', "you'd", "you'll", "you're", "you've", 'your', 'yours', 'your
self', 'yourselves']
```

We double-checked that we were in fact working with 15 million comments.

In [44]:
```python
# how many comments are we actually dealing with?
f = open('dedupped-1-comment-per-line.json')
count = 0
# dump is too big to read into memory, so we're using an iterator
for line in f:
    count += 1

print count
```

```
15505576
```

We are dealing with exactly 15,505,576 unique comments.

The comment dump extracts to a file more than 40 GB in size. This is too large to be read into memory, so we needed a way to work with subsets of the data. For this purpose, we wrote the function `sample`.

In [3]:
```python
def sample(comment_count, start=0, progress=None):
    "Sample the comment dump. Passed an int, returns that many comments as a list of dicts.
"
    # open uncompressed September 2013 Reddit comment dump
    f = open('dedupped-1-comment-per-line.json')
    count = 0
    comments = []
    idx = 0
    # dump is too big to read into memory, so we're using an iterator
    for line in f:
        if idx < start:
            idx += 1
            if progress is not None and idx % progress == 0:
                print "idx: {}".format(idx)
            continue

        if count < comment_count:
            comments.append(json.loads(line))
            count += 1
            if progress is not None and count % progress == 0:
                print count
                sys.stdout.flush()
        else:
            break

    return comments
```

Below is a single sample comment, representative of the data we searched through.

In [21]:
```
sample(1)
```

Out[21]:
```
[{u'approved_by': None,
  u'author': u'Hasaan5',
  u'author_flair_css_class': u'other-quest',
  u'author_flair_text': u'Do you even quest, bro? [Scaper since 2004] (I still hate suomi)'
,
  u'banned_by': None,
  u'body': u"And then people will complain that it isn't exactly like the old one. They wil
l never win over this.",
  u'body_html': u'&lt;div class="md"&gt;&lt;p&gt;And then people will complain that it isn&
amp;#39;t exactly like the old one. They will never win over this.&lt;/p&gt;\n&lt;/div&gt;'
,
  u'created': 1377723538.0,
  u'created_utc': 1377694738.0,
  u'distinguished': None,
  u'downs': 0,
  u'edited': False,
  u'gilded': 0,
  u'id': u'cbwxtsz',
  u'likes': None,
  u'link_id': u't3_1l8m7k',
  u'link_title': u'Old-school overrides (suggestion)',
  u'name': u't1_cbwxtsz',
  u'num_reports': None,
  u'parent_id': u't1_cbwx93k',
  u'replies': None,
  u'score_hidden': True,
  u'subreddit': u'runescape',
  u'subreddit_id': u't5_2qwxl',
  u'ups': 1}]
```

We wanted to know which subreddits were best represented in these comments. The code below serves this purpose.

In [4]:
```
def profile_comments(comments, metric):
    "Returns a dictionary with counts given a metric and a sample of comments."
    items = {}
    for comment in comments:
        item = comment[metric]
        if item in items.keys():
            items[item] += 1
        else:
            items[item] = 1

    return items
```

In [13]:
```
def visualize(title, item_count_dict, bar_color=dark2_colors[0], subplot=111,set_size=True,
 offX = 1000, offY = .5):
    "Visualize sorted item and count data with a horizontal bar graph. Takes an item/count
dict."

    # sort
    items_counts_tuples = zip(item_count_dict.keys(), item_count_dict.values())
    get_count = operator.itemgetter(1)
    items_counts_tuples = sorted(items_counts_tuples, key=get_count, reverse=True)

    items, counts = [], []
    for item, count in items_counts_tuples:
        items.append(item)
        counts.append(count)

    # plot
    pos = np.arange(len(items))

    if str(subplot)[2] == '1' and set_size:
        plt.figure(figsize=(10, len(items)/3))

    plt.subplot(subplot)

    plt.title(title)
    plt.barh(pos, counts, color=bar_color)

    # add the numbers to the side of each bar
    for p, item, count in zip(pos, items, counts):
            plt.annotate(str(count), xy=(count + offX, p + offY), va='center')

    # customize ticks
    ticks = plt.yticks(pos + .5, items)
    xt = plt.xticks()[0]
    plt.xticks(xt, [' '] * len(xt))

    # minimize chartjunk
    remove_border(left=False, bottom=False)
    plt.grid(axis = 'x', color ='white', linestyle='-')

    # set plot limits
    if set_size:
        plt.ylim(pos.max() + 1, pos.min())
#     plt.xlim(0, xmax)

        plt.tight_layout()
    plt.show()
```

We assessed the quality and quantity of personally identifying information in these 15 million comments as follows. We first searched for personally identifying information in these 15 million comments at various levels of specificity. First, we searched for four keywords in the comment bodies: "email", "birthday", "zip", and "address." With the help of regular expressions, we then searched for actual information, in particular email addresses and phone numbers. We also looked for social networking information, using regular expressions to search for comments containing general Facebook URLs and Facebook profile URLs of the form facebook.com/profile.php?id=.

For the second part of our project, we considered only those comments returned by the aformentioned searches and refined our methods, searching for personally identifying words like “my,” “personal,” “work,” “private,” “brother," and “sister." Using this filtered list of comments, we tried to identify particular users by aggregating all the information we had gathered on this user from their comments. The Facebook profile URLs alone were likely to identify a user, but we also considered phone numbers and email addresses.

We used to code below to perform the procedures described above.

In [6]:
```
# comment keyword search
import time

def keyword_search(sample_size, keywords, start=0, progress=None):
    "Keywords is a list!"
    keyword_dict = dict((k,[]) for k in keywords)
    field = 'body'
    sample_start = time.time()
    comments = sample(sample_size)
    sample_end = time.time()
    build_start = time.time()
    for comment in comments:
        words = re.findall(r'\w+\S*\w+|\w', comment['body'].lower())
        for word in words:
            if word in keywords:
                keyword_dict[word].append(comment[field])
    build_end = time.time()

    print "Sampling time: {}; run time: {}".format(sample_end - sample_start, build_end - b
uild_start)

    for k,v in keyword_dict.iteritems():
        print "{}: {}".format(k, len(v))

    return keyword_dict
```

In [8]:
```
# search for older Facebook profile URLs; see below function for preferred search
def facebook_profile_search(sample_size, start=0, progress=None):

    comments = sample(sample_size, start=start, progress=progress)
    profile_comments = []
    for comment in comments:
        if "facebook.com/profile.php" in comment['body'].lower():
            profile_comments.append(comment)
    print "Found {} comments containing a Facebook Profile URL".format(len(fb_comments))

    return profile_comments
```

In [9]:
```
def facebook_search(sample_size, start=0, progress=None):
    comments = sample(sample_size, start=start, progress=progress)

    fb_comments = []
    for comment in comments:
        if "facebook.com/" in comment['body'].lower():
            fb_comments.append(comment)

    print "Found {} comments containing a Facebook URL".format(len(fb_comments))

    return fb_comments
```

In [3]:
```
# heuristic for checking if a comment has the phrase zip code
# for text in keyword_dict['zip']:
#     if 'code' in re.findall(r'\w+\S*\w+|\w', text.lower()):
#         print text
```

In [10]:
```
def email_twitter_search(sample_size, start=0, progress=None):
    comments = sample(sample_size, start=start, progress=progress)

    at_comments = []
    print "Searching for email addresses and Twitter handles in {} comments, beginning at i
ndex {}.".format(sample_size, start)
    for comment in comments:
        words = re.findall(r'\S*@\S+', comment['body'].lower())
        if len(words) > 0:
            at_comments.append(comment)
#             print words

    print "Found {} comments containing an email or Twitter handle".format(len(at_comments)
)

    return at_comments
```

In [11]:
```
def phone_search(sample_size, start=0, progress=None):
    comments = sample(sample_size, start=start, progress=progress)

    phone_comments = []
    print "Searching for phone numbers in {} comments, beginning at index {}.".format(sampl
e_size, start)
    for comment in comments:
        # phone number regex from http://stackoverflow.com/questions/123559/a-comprehensive
-regex-for-phone-number-validation
        words = re.findall(r'^(?:(?:\+?1\s*(?:[.-]\s*)?)?(?:\(\s*([2-9]1[02-9]|[2-9][02-8]1
|[2-9][02-8][02-9])\s*\)|([2-9]1[02-9]|[2-9][02-8]1|[2-9][02-8][02-9]))\s*(?:[.-]\s*)?)?([2
-9]1[02-9]|[2-9][02-9]1|[2-9][02-9]{2})\s*(?:[.-]\s*)?([0-9]{4})(?:\s*(?:#|x\.?|ext\.?|exte
nsion)\s*(\d+))?$', comment['body'].lower())
        if len(words) > 0:
            phone_comments.append(comment)
#             print words

    print "Found {} comments containing a phone number".format(len(phone_comments))

    return phone_comments
```

In [ ]:
```
sample_size = 1000000
start=0
progress=100000
email_and_twitter = email_twitter_search(sample_size, start=start, progress=progress)
phone = phone_search(sample_size, start=start, progress=progress)
profile = facebook_profile_search(sample_size, start=start, progress=progress)
fb_comments = facebook_search(sample_size, start=start, progress=progress)
```

The iterative search above is an extraordinarily slow way to search through all 15 million comments, since it requires multiple passes of all 15 million comments. A faster way of searching for the information we want is to conduct a parallelized search, performing all searches in a single pass.

In [32]:
```
def search(sample_size, conds, keywords, logpath=None, start=0, progress=None):
    print "Sampling..."
    comments = sample(sample_size, start=start, progress=progress)
    print "Searching in {} comments, beginning at index {}.".format(sample_size, start)
    # build dictionary for results
#     res_comments = dict((cond["name"],[]) for cond in conds)

    # backup dict for ids for writing (speeds up writes)
    res_ids = dict((cond["name"],[]) for cond in conds)
    for keyword in keywords:
        res_ids[keyword] = []

    # seperate dict for keyword results
#     keyword_dict = dict((k,[]) for k in keywords)

    # open file for logging
    if logpath is not None:
        logfile = open(logpath, 'a+')

    # go over each comment (and each function for each comment)
    for idx,comment in enumerate(comments):
        for cond in conds:
            if cond["f"](comment):
#                 res_comments[cond["name"]].append(comment)
                # add to res_ids for fast writing
                res_ids[cond["name"]].append(comment['id'])

        # keyword search (kinda hacked in)
        words = re.findall(r'\w+\S*\w+|\w', comment['body'].lower())
        for word in words:
            if word in keywords:
#                 keyword_dict[word].append(comment)

                # add to res_ids for fast writing (keyword and condition name cannot overla
p)
                res_ids[word].append(comment['id'])

        # optional progress print
        if progress is not None and idx % progress == 0:
            print idx
            sys.stdout.flush()

            # log ids
            if logfile is not None:
                logfile.write(json.dumps(res_ids)+'\n')

    # print results
    for name in res_ids:
        if name in keywords:
            print "Found {} comments containing \"{}\"".format(len(res_ids[name]), name)
        else:
            print "Found {} comments containing a {}".format(len(res_ids[name]), name)

#     res_comments["keywords"] = keyword_dict

    return res_ids
```

In [9]:

```
def contains_phone(comment):
    # phone number regex from http://stackoverflow.com/questions/3868753/find-phone-numbers
```

```
-in-python-script
    words = re.findall(r'\W(\d{3}-)?\d{3}-\d{4}\W', comment['body'].lower())
    return len(words) > 0

def contains_email_twitter(comment):
    words = re.findall(r'\S*@\S+', comment['body'].lower())
    return len(words) > 0

def contains_facebook(comment):
    return "facebook.com/" in comment['body'].lower()

def contains_facebook_profile(comment):
    return "facebook.com/profile.php" in comment['body'].lower()
```

In [14]:
```
test_text = "this is a phone number 123-719 yeah it is"
test_comment = {"body":test_text}
contains_phone(test_comment)
```

Out[14]:
```
False
```

Now that we have collected this trove of information, we want to use it to manually identify Reddit users and their personal information. Unfortunately, as we will discuss in our results section, the above methods paint too broad a picture. Too many comments, for instance, mention the word "birthday" without revealing personally identifying information. In order to locate actual identifying information, we need to refine our results to maximize the chance that a comment we produce contains information we care about. We can do this by searching through the filtered comments for certain keywords like the word "my." For instance, in the case of birthdays, we can search for the word "today," only producing comments that contain the word "today" AND the word "birthday."

Finally, viewing the output of these results can be a manual process, looking for comments that definitively reveal personal information. Looking at Facebook URLs in particular, we may be able to learn a user's full identity.

In [5]:
```
def refine(comments, type, keywords):
    result_set = set()
    for comment in comments[type]:
        for keyword in keywords:
            if keyword in comment['body'].lower():
                result_set.add(comment['body'])
    return result_set
```

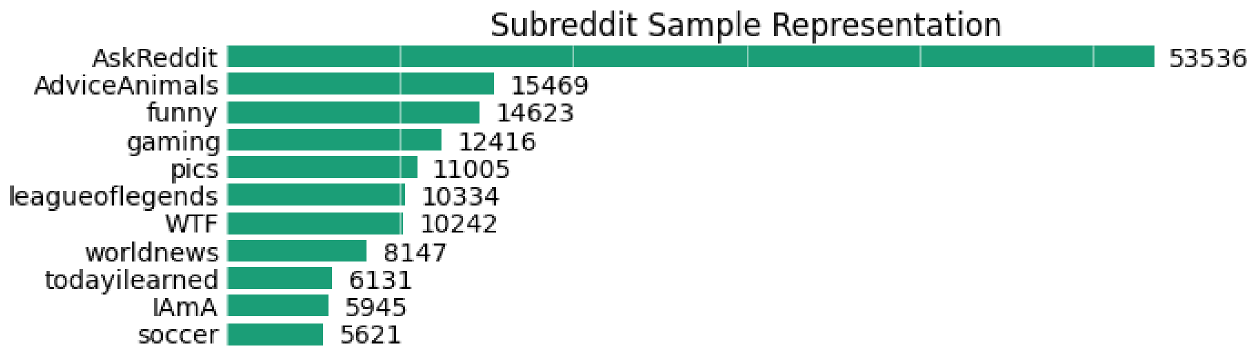
Results

The following is output from the code defined above.

In [54]:
```
# toggle to find most highly represented subreddits (500k comment sample)
# comments = sample(500000)
# subreddit_counts = profile_comments(comments, 'subreddit')
# truncated_subreddit_counts = {}
# for subreddit, count in subreddit_counts.iteritems():
#     if count >= 5000:
#         truncated_subreddit_counts[subreddit] = count
# print truncated_subreddit_counts
```

Out[54]:
```
{u'AdviceAnimals': 15469,
 u'AskReddit': 53536,
 u'IAmA': 5945,
 u'WTF': 10242,
 u'funny': 14623,
 u'gaming': 12416,
 u'leagueoflegends': 10334,
 u'pics': 11005,
 u'soccer': 5621,
 u'todayilearned': 6131,
 u'worldnews': 8147}
```

In [61]:
```
# toggle to visualize results from above
# visualize("Subreddit Sample Representation", truncated_subreddit_counts)
```



As we can see, the comments primarily come from r/AskReddit, a question-and-answer community known for its high comment volume. Comments from AskReddit account for about 11% of comments in our sample. Many comments also come from r/AdviceAnimals and r/funny, each accounting for about 3% of comments.

We also wanted to know which words were most commonly used in these comments. The code below serves this purpose.

In [11]:
```
# toggle to find most common words in comments
# comments = sample(2000)
# all_comment_text = [comment['body'] for comment in comments]
# words = [re.findall(r'\w+\S*\w+|\w', text.lower()) for text in all_comment_text]
# words = [item for sublist in words for item in sublist]

# scrubbed_words = []
# for word in words:
#     if word in stop_words:
#         continue
#     else:
#         scrubbed_words.append(word)

# common_words = collections.Counter(scrubbed_words).most_common(25)
# print common_words
```

```
[(u'just', 294), (u'like', 283), (u'can', 255), (u'one', 206), (u'get', 204), (u'people', 1
97), (u'think', 173), (u'will', 172), (u'really', 138), (u'good', 128), (u'time', 118), (u'
make', 113), (u'see', 112), (u'know', 110), (u'still', 105), (u'well', 104), (u'much', 100)
, (u'go', 96), (u'even', 91), (u'now', 90), (u'also', 88), (u'way', 88), (u'want', 86), (u'
going', 86), (u'right', 83)]
```

To our disappointment, this exploration was not particularly enlightening. None of these words reflect privacy concerns or other interesting facets of the discourse on Reddit.

The iterative Facebook URL search successfully identified one user after searching through one million comments. The Facebook URL returned redirected to a Facebook Page, but this user inadvertently embedded his Facebook user ID in the URL. We can visit https://facebook.com/profile.php?id=100000427775740 to view the Facebook Profile of a likely high school student named Austin Jenkins.

The parallelized search was highly successful.

In [10]:
```
conds = [{"name": "phone number", "f": contains_phone}, {"name": "email/twitter", "f": cont
ains_email_twitter}, {"name": "facebook url", "f": contains_facebook}, {"name": "facebook p
rofile", "f": contains_facebook_profile}]
keywords = ["email", "birthday", "zip", "address"]

# toggle to run our search on 15 1M comment samples
# results_0 = search(1000000, conds, keywords, logpath="gov-0.log", start=0, progress=25000
0)
# results_1 = search(1000000, conds, keywords, logpath="gov-1.log", start=1000000, progress
=250000)
# results_2 = search(1000000, conds, keywords, logpath="gov-2.log", start=2000000, progress
=250000)
# results_3 = search(1000000, conds, keywords, logpath="gov-3.log", start=3000000, progress
=250000)
# results_4 = search(1000000, conds, keywords, logpath="gov-4.log", start=4000000, progress
=250000)
# results_5 = search(1000000, conds, keywords, logpath="gov-5.log", start=5000000, progress
=250000)
# results_6 = search(1000000, conds, keywords, logpath="gov-6.log", start=6000000, progress
=250000)
# results_7 = search(1000000, conds, keywords, logpath="gov-7.log", start=7000000, progress
=250000)
# results_8 = search(1000000, conds, keywords, logpath="gov-8.log", start=8000000, progress
=250000)
# results_9 = search(1000000, conds, keywords, logpath="gov-9.log", start=9000000, progress
=250000)
# results_10 = search(1000000, conds, keywords, logpath="gov-10.log", start=10000000, progr
ess=250000)
# results_11 = search(1000000, conds, keywords, logpath="gov-11.log", start=11000000, progr
ess=250000)
# results_12 = search(1000000, conds, keywords, logpath="gov-12.log", start=12000000, progr
ess=250000)
# results_13 = search(1000000, conds, keywords, logpath="gov-13.log", start=13000000, progr
ess=250000)
# results_14 = search(1000000, conds, keywords, logpath="gov-14.log", start=14000000, progr
ess=250000)


# cleaned output
# Sample 0
# Found 1 comments containing a facebook profile
# Found 244 comments containing "zip"
# Found 146 comments containing a phone number
# Found 1068 comments containing a email/twitter
# Found 1258 comments containing "birthday"
# Found 1704 comments containing "address"
# Found 331 comments containing a facebook url
# Found 1927 comments containing "email"
# Sample 1
# Found 1 comments containing a facebook profile
# Found 258 comments containing "zip"
# Found 135 comments containing a phone number
# Found 1027 comments containing a email/twitter
# Found 1366 comments containing "birthday"
# Found 1813 comments containing "address"
# Found 310 comments containing a facebook url
# Found 1583 comments containing "email"
# Sample 2
# Found 1 comments containing a facebook profile
# Found 191 comments containing "zip"
# Found 138 comments containing a phone number
# Found 1003 comments containing a email/twitter
# Found 1360 comments containing "birthday"
# Found 1691 comments containing "address"
# Found 301 comments containing a facebook url
# Found 1739 comments containing "email"
# Sample 3
# Found 1 comments containing a facebook profile
# Found 204 comments containing "zip"
# Found 104 comments containing a phone number
# Found 1030 comments containing a email/twitter
# Found 1503 comments containing "birthday"
# Found 1530 comments containing "address"
# Found 253 comments containing a facebook url
# Found 1327 comments containing "email"
# Sample 4
# Found 0 comments containing a facebook profile
# Found 310 comments containing "zip"
# Found 104 comments containing a phone number
# Found 1095 comments containing a email/twitter
# Found 1395 comments containing "birthday"
# Found 1439 comments containing "address"
# Found 257 comments containing a facebook url
# Found 1473 comments containing "email"
# Sample 5
# Found 0 comments containing a facebook profile
# Found 237 comments containing "zip"
# Found 135 comments containing a phone number
# Found 1167 comments containing a email/twitter
# Found 1352 comments containing "birthday"
# Found 1550 comments containing "address"
# Found 283 comments containing a facebook url
# Found 1447 comments containing "email"
# Sample 6
# Found 0 comments containing a facebook profile
# Found 202 comments containing "zip"
# Found 107 comments containing a phone number
# Found 1006 comments containing a email/twitter
# Found 1586 comments containing "birthday"
# Found 1635 comments containing "address"
# Found 322 comments containing a facebook url
# Found 1714 comments containing "email"
# Sample 7
# Found 2 comments containing a facebook profile
# Found 229 comments containing "zip"
# Found 142 comments containing a phone number
# Found 1074 comments containing a email/twitter
# Found 1519 comments containing "birthday"
# Found 1641 comments containing "address"
# Found 377 comments containing a facebook url
# Found 1948 comments containing "email"
# Sample 8
# Found 2 comments containing a facebook profile
# Found 220 comments containing "zip"
# Found 123 comments containing a phone number
# Found 1011 comments containing a email/twitter
# Found 1339 comments containing "birthday"
# Found 1616 comments containing "address"
# Found 351 comments containing a facebook url
# Found 1976 comments containing "email"
# Sample 9
# Found 2 comments containing a facebook profile
# Found 204 comments containing "zip"
# Found 131 comments containing a phone number
# Found 982 comments containing a email/twitter
# Found 1464 comments containing "birthday"
# Found 1592 comments containing "address"
# Found 364 comments containing a facebook url
# Found 1811 comments containing "email"
# Sample 10
# Found 4 comments containing a facebook profile
# Found 212 comments containing "zip"
# Found 109 comments containing a phone number
# Found 984 comments containing a email/twitter
# Found 1598 comments containing "birthday"
# Found 1364 comments containing "address"
# Found 363 comments containing a facebook url
# Found 1465 comments containing "email"
# Sample 11
# Found 3 comments containing a facebook profile
# Found 217 comments containing "zip"
# Found 116 comments containing a phone number
# Found 892 comments containing a email/twitter
# Found 1451 comments containing "birthday"
# Found 1392 comments containing "address"
# Found 286 comments containing a facebook url
# Found 1341 comments containing "email"
# Sample 12
# Found 0 comments containing a facebook profile
# Found 198 comments containing "zip"
# Found 157 comments containing a phone number
# Found 1011 comments containing a email/twitter
# Found 1424 comments containing "birthday"
# Found 1642 comments containing "address"
# Found 341 comments containing a facebook url
# Found 1582 comments containing "email"
# Sample 13
# Found 2 comments containing a facebook profile
# Found 219 comments containing "zip"
# Found 117 comments containing a phone number
# Found 983 comments containing a email/twitter
# Found 1494 comments containing "birthday"
# Found 1790 comments containing "address"
# Found 325 comments containing a facebook url
# Found 1669 comments containing "email"
# Sample 14
# Found 0 comments containing a facebook profile
# Found 230 comments containing "zip"
# Found 133 comments containing a phone number
# Found 971 comments containing a email/twitter
# Found 1836 comments containing "birthday"
# Found 1686 comments containing "address"
# Found 360 comments containing a facebook url
# Found 1851 comments containing "email"



# summed
facebook_profile_ct = 1+1+1+1+0+0+0+2+2+2+4+3+0+2+0
zip_keyword_ct = 244+258+191+204+310+237+202+229+220+204+212+217+198+219+230
phone_number_ct = 146+135+138+104+104+135+107+142+123+131+109+116+157+117+133
email_twitter_ct = 1068+1027+1003+1030+1095+1167+1006+1074+1011+982+984+892+1011+983+971
birthday_keyword_ct = 1258+1366+1360+1503+1395+1352+1586+1519+1339+1464+1598+1451+1424+1494
+1836
address_keyword_ct = 1704+1813+1691+1530+1439+1550+1635+1641+1616+1592+1364+1392+1642+1790+
1686
facebook_url_ct = 331+310+301+253+257+283+322+377+351+364+363+286+341+325+360
email_keyword_ct = 1927+1583+1739+1327+1473+1447+1714+1948+1976+1811+1465+1341+1582+1669+18
51

print "We found:"
print "{} hits for 'zip', {} hits for 'birthday', {} hits for 'address', and {} hits for 'e
mail'".format(zip_keyword_ct,

              birthday_keyword_ct,

              address_keyword_ct,

              email_keyword_ct)
print "{} Facebook URLs".format(facebook_url_ct)
print "{} likely Facebook profiles".format(facebook_profile_ct)
print "{} likely phone numbers".format(phone_number_ct)
print "{} likely Twitter handles or email addresses".format(email_twitter_ct)
```

```
We found:
3375 hits for 'zip', 21945 hits for 'birthday', 24085 hits for 'address', and 24853 hits fo
r 'email'
4824 Facebook URLs
19 likely Facebook profiles
1897 likely phone numbers

15304 likely Twitter handles or email addresses
```

In [40]:

```
# toggle to write the results from above to their own files
# open('results-0.json','w').write(json.dumps(results_0))
# open('results-1.json','w').write(json.dumps(results_1))
# open('results-2.json','w').write(json.dumps(results_2))
# open('results-3.json','w').write(json.dumps(results_3))
# open('results-4.json','w').write(json.dumps(results_4))
# open('results-5.json','w').write(json.dumps(results_5))
# open('results-6.json','w').write(json.dumps(results_6))
# open('results-7.json','w').write(json.dumps(results_7))
# open('results-8.json','w').write(json.dumps(results_8))
# open('results-9.json','w').write(json.dumps(results_9))
# open('results-10.json','w').write(json.dumps(results_10))
# open('results-11.json','w').write(json.dumps(results_11))
# open('results-12.json','w').write(json.dumps(results_12))
# open('results-13.json','w').write(json.dumps(results_13))
# open('results-14.json','w').write(json.dumps(results_14))
```

```
In [9]: # toggle to read in results data from above
results_0 = json.loads(open('data/results-0.json').read())
results_1 = json.loads(open('data/results-1.json').read())
results_2 = json.loads(open('data/results-2.json').read())
results_3 = json.loads(open('data/results-3.json').read())
results_4 = json.loads(open('data/results-4.json').read())
results_5 = json.loads(open('data/results-5.json').read())
results_6 = json.loads(open('data/results-6.json').read())
results_7 = json.loads(open('data/results-7.json').read())
results_8 = json.loads(open('data/results-8.json').read())
results_9 = json.loads(open('data/results-9.json').read())
results_10 = json.loads(open('data/results-10.json').read())
results_11 = json.loads(open('data/results-11.json').read())
results_12 = json.loads(open('data/results-12.json').read())
results_13 = json.loads(open('data/results-13.json').read())
results_14 = json.loads(open('data/results-14.json').read())
```

To reduce the size of our log files, our parallelized search only saved comment IDs. The below is our way of retrieving comment bodies given a comment ID.

```
In [6]: def comment_lookup(results_obj, sample_size, sample_start, filename):
    temp = {}
    for key in results_obj:
        temp[key] = []

    comments = sample(sample_size, start=sample_start, progress=250000)
    count = 0
    for comment in comments:
        if count % 250000 == 0:
            print "Inspecting comment #{}".format(count)
            sys.stdout.flush()
        comment_id = comment['id']
        for key in results_obj:
            if comment_id in results_obj[key]:
                temp[key].append(comment)
        count += 1

    open(filename+'.json','w').write(json.dumps(temp))
```

```
In [1]: comment_lookup(results_0, 1000000, 0, 'comments_0')
comment_lookup(results_1, 1000000, 1000000, 'comments_1')
comment_lookup(results_2, 1000000, 2000000, 'comments_2')
comment_lookup(results_3, 1000000, 3000000, 'comments_3')
comment_lookup(results_4, 1000000, 4000000, 'comments_4')
comment_lookup(results_5, 1000000, 5000000, 'comments_5')
comment_lookup(results_6, 1000000, 6000000, 'comments_6')
comment_lookup(results_7, 1000000, 7000000, 'comments_7')
comment_lookup(results_8, 1000000, 8000000, 'comments_8')
comment_lookup(results_9, 1000000, 9000000, 'comments_9')
comment_lookup(results_10, 1000000, 10000000, 'comments_10')
comment_lookup(results_11, 1000000, 11000000, 'comments_11')
comment_lookup(results_12, 1000000, 12000000, 'comments_12')
comment_lookup(results_13, 1000000, 13000000, 'comments_13')
comment_lookup(results_14, 1000000, 14000000, 'comments_14')
```

In [28]:

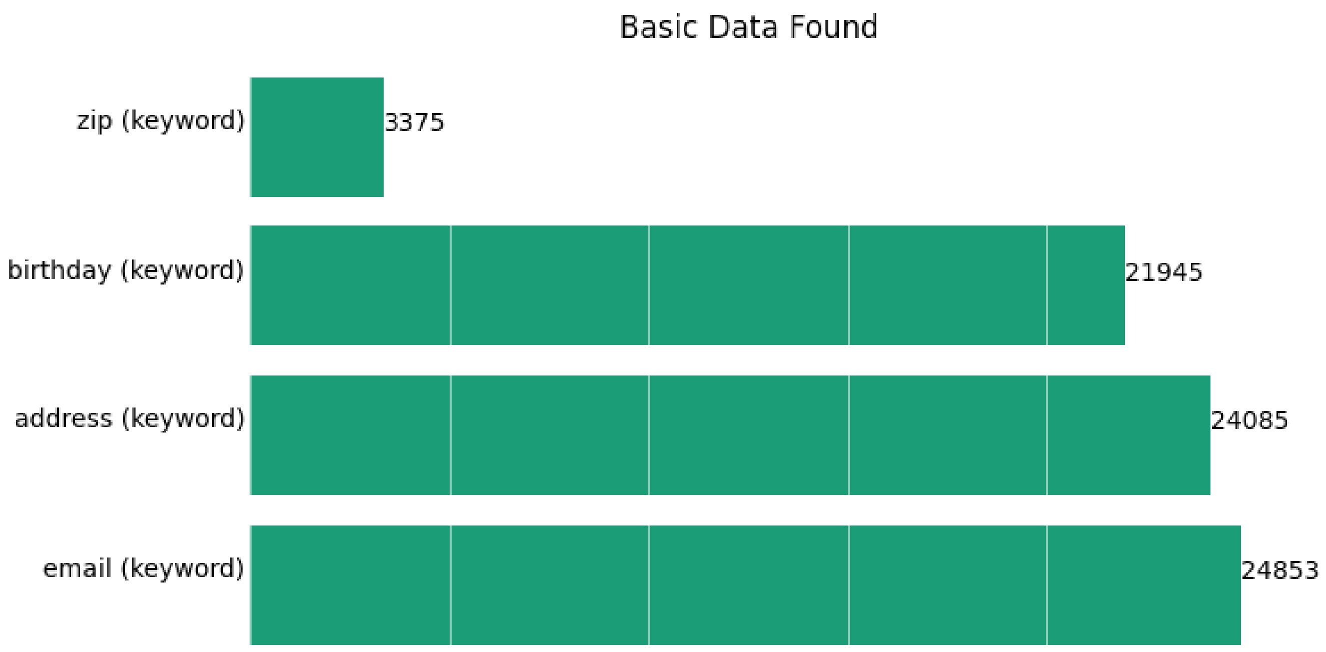
```
# an example of how to look up bodies from the retrieved comments
x = json.loads(open('comments_12.json').read())
print x['email'][0]['body']
```

```
Why would you send the full amount if you don't agree?  I've battled with landlords before.
 You usually just have to confront him.  Before you get all crazy, check the laws of your s
tate first.  This should be very quick since landlord laws are usually very rigid.  Then kn
ow your rights, and confront him if you can.  If you can't, send him a certified letter exp
laining that he changed the rent, and copy the email.  At the worst, all you have to do is
go to court.  Dispute or not pay the rent that you think you owe every month.  And sign a n
ew lease if you can.
```

The keyword search revealed that Reddit users are at least discussing personal information. In our data set of 15 million comments, the keywords email, address, and birthday occurred over 20,000 times each.

In [14]:

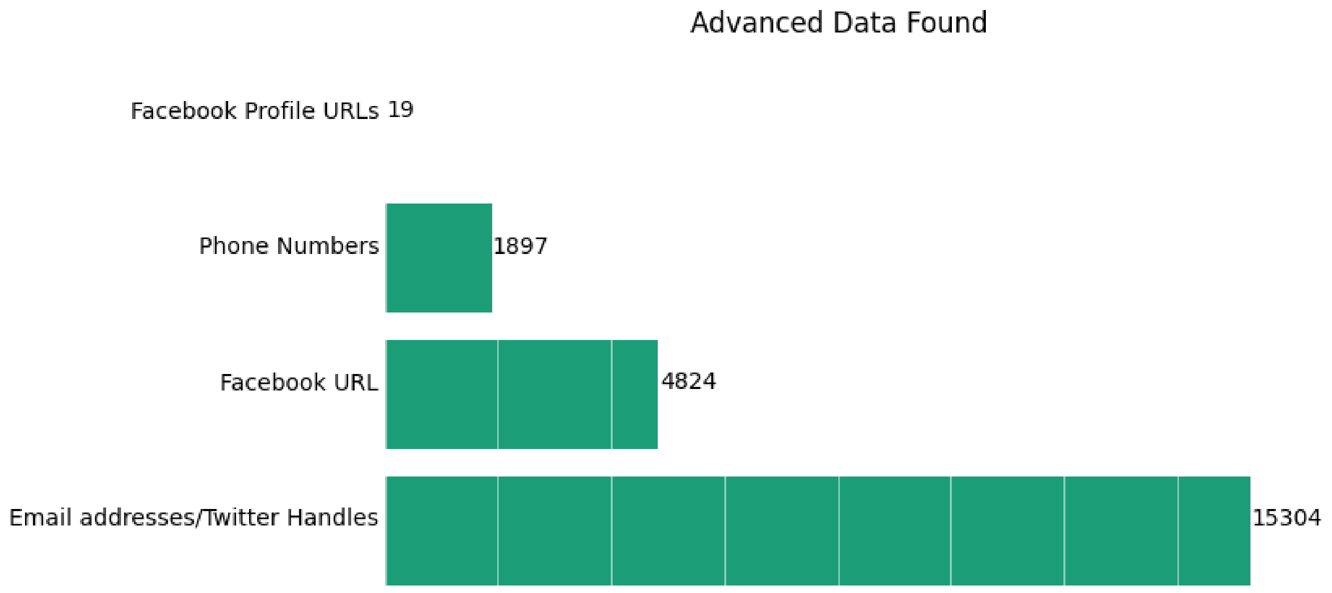
```
keyword_data = {"zip (keyword)": zip_keyword_ct, "birthday (keyword)": birthday_keyword_ct,
 "address (keyword)": address_keyword_ct, "email (keyword)": email_keyword_ct}
visualize("Basic Data Found", keyword_data, set_size=False, offX=50)
```



Looking at the comments returned by this search revealed that the signal to noise ratio is pretty low – most of the comments do not contain identifying information, just the keyword in question. For instance, discussion of NSA surveillance often contained the word "email" without the user actually posting a real email address. We determined that this data would not allow us to easily and automatically identify users.

Searches for more specific information – actual email addresses, phone numbers, and Facebook URLs - were fruitful.

In [16]:
```
other_data = {"Facebook URL": facebook_url_ct, "Phone Numbers": phone_number_ct, "Email add
resses/Twitter Handles": email_twitter_ct, "Facebook Profile URLs": facebook_profile_ct}
visualize("Advanced Data Found", other_data, set_size=False, offX=50)
```



We found almost 5,000 Facebook URLs, and 19 Facebook profile URLs. Looking at these, they were much more identifying, often tying a Reddit account to that user's band's Facebook page, or even to their personal Facebook account. Manual interaction with the results also paid significant dividends. Filtering our results by keywords such as "my" and "today" left us with a much smaller dataset which could be surveyed manually. Specifically, 929 of the comments containing "birthday" also contained "today," and 814 of the comments containing a Facebook URL contained the word "my." We discovered that most of these comments were personal; comments like "today is my birthday" and "like my page" were common, leading us to believe that this dataset would be much more useful when it came to identifying users.

In [17]:
```
""" Builds Master dict from comments_0 to comments_14 """
x = json.loads(open('comments_0.json').read())

keys = x.keys()
result_dict = dict((k, []) for k in keys)
for i in xrange(15):
    x = json.loads(open('comments_{}.json'.format(i)).read())
    for k in keys:
        result_dict[k] += x[k]
```

In [24]:
```
# look at facebook profiles
for comment in result_dict['facebook profile']:
    print comment['body']
    print "########"
```

```
Here you go! https://facebook.com/profile.php?id=330659473723743&amp;__user=100000427775740
########
Thank you, citizen! I will remove you from our NSA observation list, instead we will focus
on our next target, this guy:
http://facebook.com/profile.php?=73322363
########
https://m.facebook.com/profile.php?id=119985464691108
```

```
########
http://facebook.com/profile.php?=73322363
########
At least you weren't [THIS person](http://www.facebook.com/profile.php?=743264506)
########
[Here](https://facebook.com/profile.php?id=129439940472702) is the non-mobile version of th
is site.
########
I might.  [Is this you Kismet7?](http://facebook.com/profile.php?=73322363)
########
https://www.facebook.com/profile.php?id=100000152574438&amp;directed_target_id=0
########
http://www.facebook.com/profile.php?id=621834263

I actually don't use my FB at all (I might when school starts) but I have email notificatio
ns sent so if you send me a message I will get it even though I never logon :) Of course I
will accept your friend request
########
[This one?](https://www.facebook.com/profile.php?=87912848)
########
Please mesage send riot and arcade hecarim https://www.facebook.com/profile.php?id=10000542
2148710

########
i need blitz code https://www.facebook.com/profile.php?id=100006331893271&amp;ref=tn_tnmn
########
Arcade Heca Riot Blitz Olan varsa faceden atabilirmi ya lütfen 2 gündür arıyorum :(( https:
//www.facebook.com/profile.php?id=100000267590563
########
Fag reposter OP's facebook
http://facebook.com/profile.php?=73322363
########
https://www.facebook.com/profile.php?id=100006696840159
########
Hey guys can you like my fb page:
http://m.facebook.com/profile.php?id=584026538326111&amp;v=feed&amp;_rdr
########
FAST DONT MISS UR CHANCE FOR FREE SKIN EVERY THIRD WINS - ADD ME AS A FRIEND MESSAGE ME AND
 IF UR ARE THIRD PERSON U WIN SKIN ONLY TODAY https://www.facebook.com/profile.php?id=10000
6637927471 !!!
########
Cattoos in Dunkirk  ny. I've seen some wonderful portraits done by her although I've never
been to her studio before https://m.facebook.com/profile.php?id=125060270882401&amp;refsrc=
http%3A%2F%2Fwww.google.com%2F&amp;refid=9&amp;_rdr#_=_
########
[He does.](https://www.facebook.com/profile.php?id=9369714&amp;fref=ts)

########
```

Above are comments that contained FB profiles. In particular, we see one URL that contains a "user=" parameter, which is the poster's account, accidentally included. Additionally, we see a user reference "my fb page" (https://m.facebook.com/profile.php?id=584026538326111), which goes to a page with 2 listed owners, one of which is likely the poster. Another few appear to be giving out their personal page as a means of communication.

In [2]:
```
today_birthdays = refine(result_dict, 'birthday', ['today'])
```

```
print "Found {} birthdays containing 'today'".format(len(today_birthdays))
for bday_comment in today_birthdays:
    print bday_comment
    print "########"
```

Found 929 birthdays containing 'today'

SAMPLE OUTPUT:

I have been wanting some different colored jeans, so I bought some red ones last night at a discount store called Dirt Cheap (for $4.80!), and I am totally rocking them today! And it's also my birthday.

I was at home celebrating with it as the main event. We had a huge party and still celebrate it today. Just came home from sushi dinner!

9/11 is my birthday :P

It's my birthday today :'(

Its my birthday today. Practise here!

Yesterday was my birthday and I was just feeling really shitty. Like oh another year passed, still not presenting, still not happy, etc etc. Wasn't able to concentrate at work just dwelling on my situation all day on the brink of breaking down. And my mom is taking me to dinner tonight to celebrate, but I'm not out publicly, and it annoys me to celebrate as a guy.

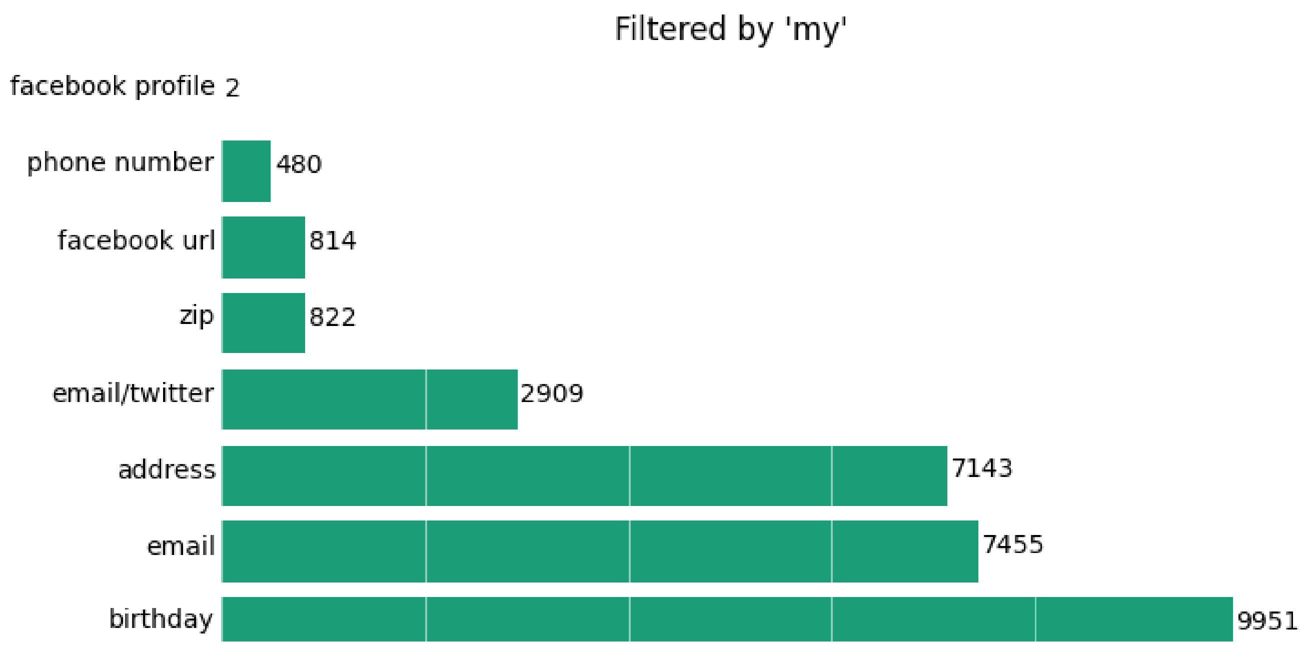
Though the day got slightly better after work. Viewed an apartment and everything seemed to go really well, and signing the lease for it today(my current place I'm in got foreclosed on). Then put on some nice pajamas and roasted a birthday chicken and watched Dexter all night.

Friends birthday today.

Today is my birthday

I'm feeling happy with my life

In [21]:
```
my_counts = {}
for k in result_dict:
    my_refined = refine(result_dict, k, ['my'])
    my_counts[k] = len(my_refined)
visualize("Filtered by 'my'", my_counts, set_size=False, offX=50)
```



Now the overwhelming 5,000 FB URLs is a much more manageable 800. Let's take a look at some of the comment containing these URLs.

In [3]:
```
my_refined = list(refine(result_dict, "facebook url", ['my']))
for fb_comment in my_refined:
    print fb_comment
    print "############"
```

SAMPLE OUTPUT:

Please help me reach My Goal of 1000 Likes on my Business page! https://www.facebook.com/CoachTaraWoodruff

Huf Lurkers Tee - DSWOT

Palace Tee - 8.5/10 Condition

Vans Tee - 8/10 Condition

Koto Aztec Long sleeve - 8/10 Condition

Gold Coast Complete Cruiser - Won it in a contest, rode it about 10 times.

All size large besides Koto Aztec Long Sleeve which is a medium.

Tag is my name on Facebook (https://www.facebook.com/aleddis) and if you don't believe me already feel free to send me a message through Facebook instead of Reddit.

actual bike: I couldn't take a pic with my phone but someone took this one the other night: https://www.facebook.com/photo.php?fbid=10201900198764533&set=at.10201514909812550.1073741825.1170750049.100003338799680&type=1&theater

Since I don't want to spam Reddit with photos, I'll be uploading them as I get them to my Facebook page (https://www.facebook.com/ViverraCosplay).

We have starting to go through and characterize comments as containing links to pages belonging to the poster, or not. We marked each identifying comment with a "1", and non-identifying comments with a "0" and comments that identify someone else (a relative, etc) with "2".

In [39]:
```
found_str = "11001110101000101221021021211100"
print "In the first {} comments, {} contain a page belonging to the user and {} contain a p
age belonging to an acquaintance of the user.".format(len(found_str), found_str.count('1'),
 found_str.count('2'))
print "The last comment checked begins with \"{}\"".format(my_refined[len(found_str)-1][:50
])
```

```
In the first 32 comments, 15 contain a page belonging to the user and 5 contain a page belo
nging to an acquaintance of the user.
The last comment checked begins with "[Mons Montis](https://www.facebook.com/pages/Mons-"
```

Discussion

As we expected, there is a lot of personally identifying information on Reddit. However, it was not as easy as we had hoped to separate identifying, personal information from useless comments. By searching for specific patterns of information and by filtering broad searches with “personal” words, we were able to connect personal information, like birthdays, to Reddit users. Furthermore, we were able to identify 20 Reddit users, specifically through their Facebook profiles, after only looking at 50 comments (with over 700 more to look at). We did not contact the users we tentatively identified to confirm their suspected identities.

Moving forward, we would like to run more complex analyses on the data in hopes of finding a better way to identify "personal" comments. One way of doing this is through a combination of natural language processing and machine learning techniques. If our methods are able to reliably identify comments containing personal information, we could write a program which continuously polls the stream of new comments on Reddit and alerts us, and potentially the user, to new personal posts. We could host this program online as a counterpart to Reddit Investigator (http://www.redditinvestigator.com/), a tool which allows a user to look up information associated with a Reddit username.

Reddit cannot prevent personally identifying information from appearing in comments and posts on the site. Censorship of this kind of content, even if it was done for the safety of the site's users, would not be received well by the Reddit community. A smarter way of alleviating this problem would be to provide users with a tool which alerts them when they post personal information, giving them the option to edit or delete their post or comment to redact the sensitive information.

References

The Reddit comment data set was collected and cleaned by Reddit user `w2m3d` and posted in this thread (http://www.reddit.com/r/datasets/comments/1mbsa2/155m_reddit_comments_over_15_days/).

The Reddit logo ("Snoo") was taken from here (http://www.redditstatic.com/about/assets/reddit-alien.png).

---

css tweaks in this cell