Examining Google Ngrams with Apache Pig on Amazon Elastic MapReduce

Michael Tiernay*

June 10, 2013

Question

Google has digitized 6% of all books ever published, covering five centuries and eight languages.[1] The resulting data (called Ngrams) allow the analyst to examine the usage and frequency of words and phrases over time. To this end, Google provides an interface for those interested to see how word usage has changed over time.[2] Take, for example, the word "gay". According to Wikipedia, "The word gay arrived in English during the 12th century from Old French 'gai', most likely deriving ultimately from a Germanic source. For most of its life in English, the word's primary meaning was 'joyful', 'carefree', 'bright and showy', and the word was very commonly used with this meaning in speech and literature. For example, the optimistic 1890s are still often referred to as the Gay Nineties."[3]

However, "By the mid-20th century, gay was well established in reference to hedonistic and uninhibited lifestyles and its antonym straight, which had long had connotations of seriousness, respectability, and conventionality, had now acquired specific connotations of heterosexuality. In the case of gay, other connotations of frivolousness and showiness in dress ('gay apparel') led to association with camp and effeminacy. This association no doubt helped the gradual narrowing in scope of the term towards its current dominant meaning, which was at first confined to subcultures. Gay was the preferred term since other terms, such as queer, were felt to be derogatory."

*michael.tiernay@nyu.edu

[1] http://aclweb.org/anthology-new/P/P12/P12-3029.pdf

[2] http://books.google.com/ngrams

[3] http://en.wikipedia.org/wiki/Gay

Figure 1: Use of 'gay' Over Time

Figure 1 shows how the usage rate of the word 'gay' changes over time. Between 1800 and 1900 the word is used at a rate between .000015% and .00002% of the time. This rate drops substantially over the next 70 years before exploding in the later half of the 1980's.

While usage rate is interesting in its own right, I found myself wanting to know the answer to the question: do the words that appear near the word 'gay' change over time? You can't answer this question with the Google Ngram viewer, so I turned directly to the Google Ngram data itself.

What is an N-gram?

To answer my question I selected the Google 5-gram data which is described as follows:[4]

> N-grams are fixed size tuples of items. In this case the items are words extracted from the Google Books corpus. The n specifies the number of elements in the tuple, so a 5-gram contains five words or characters.
>
> The n grams in this dataset were produced by passing a sliding window of the text of books and outputting a record for each new token. For example, the following sentence:
>
> The yellow dog played fetch.
>
> Would produce the following 2-grams:
>
> ['The', 'yellow'] ['yellow', 'dog'] ['dog', 'played'] ['played', 'fetch'] ['fetch', '.']
>
> Or the following 3-grams:
>
> ['The', 'yellow', 'dog'] ['yellow', 'dog', 'played'] ['dog', 'played', 'fetch'] ['played', 'fetch', '.']

[4] http://aws.amazon.com/datasets/8172056142375670

Working with the 5-gram data allows me to examine the four words that preceded and followed the word 'gay'. The 5-gram data contains 10,175,161,944 rows and is 90.2 GB compressed. Amazon hosts the Ngram data as a Public Dataset that is accessible for free on S3 and Google has the entire dataset available for download[5].

Workflow

I read the data onto 20 distributed computers and created two smaller datasets, one filtered the last element of the 5-gram to match the word 'gay' and the other filtered the first element of the 5-gram to match the word 'gay'. Then, I added up the usage of each word per year. The majority of the computing time dealt with reading in, decompressing, and filtering the data (9-10 hours). Once filtered, the computation time was minimal (10-15 minutes).

The dataset consisted of (word near gay, year, number of times used that year). I loaded the data into R and added up the number of words used in each year and divide each particular word with that number to provide the percentage for that year (that is: (number of times a particular word is used in a year)/ (number of words in that year)). Now I can see how certain words become more or less likely to appear near 'gay' over time.

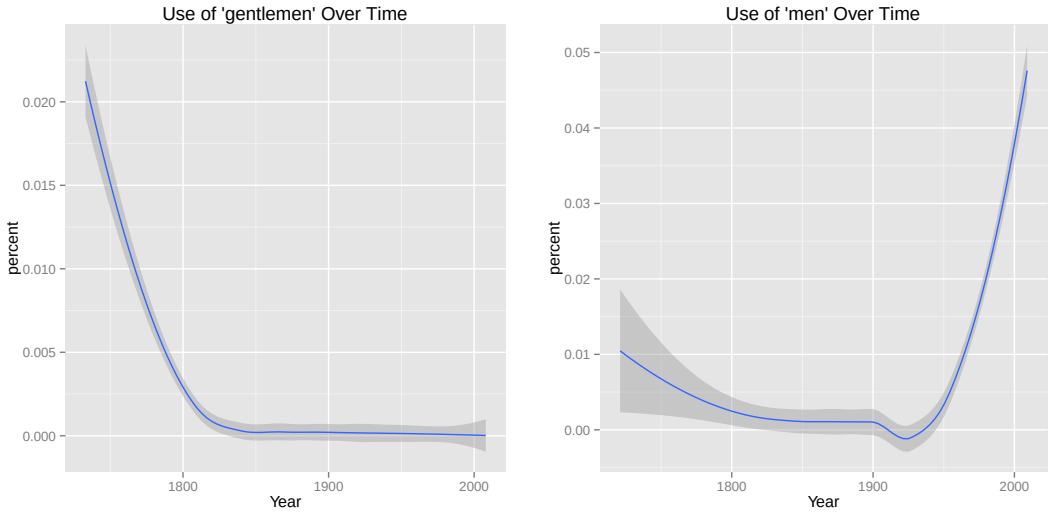
Results

To start, take the words 'gentlemen' and 'men'. It turns out that prior to 1825 the word 'gentlemen' was used in the same phrase as the word 'gay' quite frequently. Contrast this with the use of the word 'men'. In the 1950's it became much more common for the word 'men' to be in the same phrase as the word 'gay'.[6]

[5] http://storage.googleapis.com/books/ngrams/books/datasetsv2.html

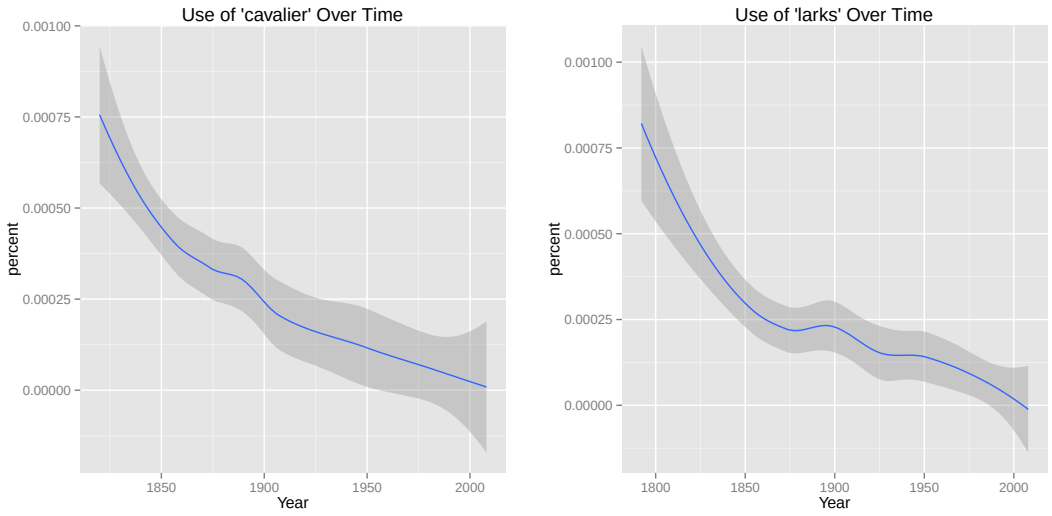
[6] All plots produced in R using the loess option in the geom_smooth function in ggplot2.

Figure 2: Words Associated with 'gay' Over Time



Historically, some words used to be used more commonly with 'gay' in the past. A lark, for example, is "a merry, carefree adventure; frolic; escapade." Additionally, cavalier is "A gallant or chivalrous man, especially one serving as escort to a woman of high social position." These words appear less frequently over time next to 'gay'.

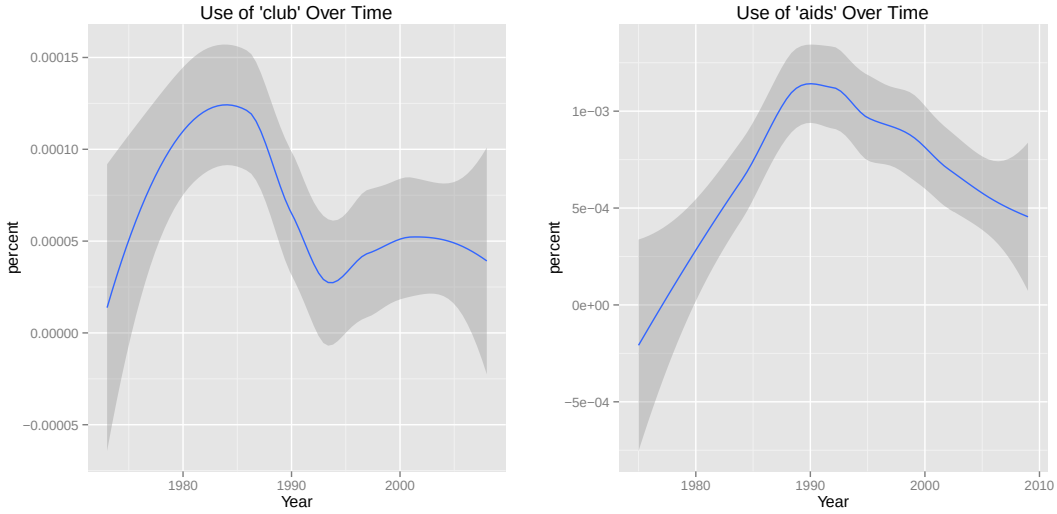
Figure 3: Words Associated with 'gay' Over Time



On the other hand, some words only appear near gay recently. For example, 'aids' first appeared in the 1970's, peaked in the early 1990's, and has decreased ever since. While several references to the word 'club' date back to the 1800's, there was a sharp increase in the use of the word through

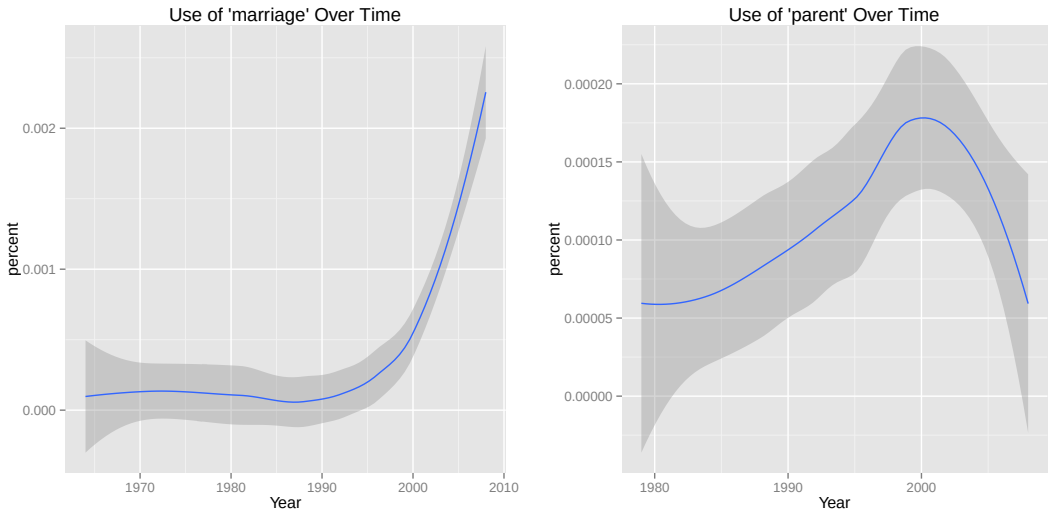
the 1970's until its peak in the mid 1980's. Interestingly, it seems that literature is most concerned with gay 'clubs' and 'aids' in the 1980's and early 1990's, but the level of interest has fallen since then.

Figure 4: Words Associated with 'gay' Over Time



More recently, gay 'marriage' and 'parents' seem to have become popular topics. Around the mid-1990's the discussion of 'marriage' took off, while the discussion of 'parents' peaked around 2000, but has dropped off since then.

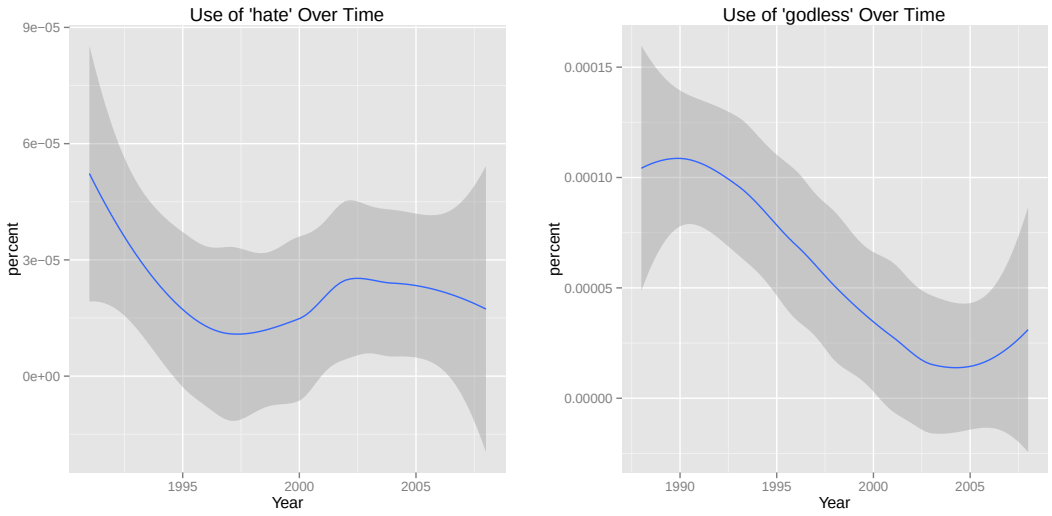
Figure 5: Words Associated with 'gay' Over Time



Interestingly, the words 'hate' and 'godless' did not appear near 'gay' until the late 1980's/early

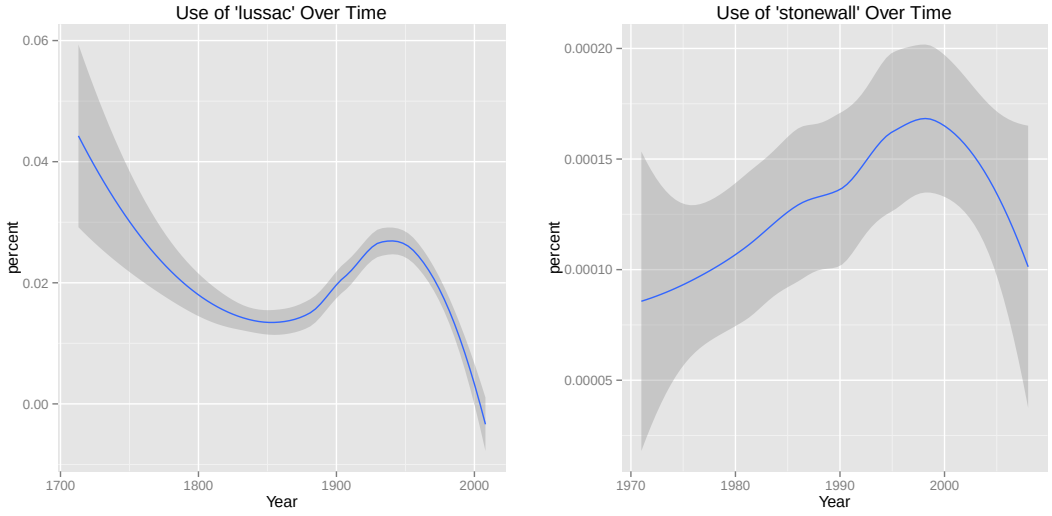
1990's. Nevertheless, their usage has decreased from their initial levels.

Figure 6: Words Associated with 'gay' Over Time



Finally, some proper nouns appear in the dataset. The word 'lussac' initially puzzled me, until I discovered French chemist Joseph Louis Gay-Lussac. Additionally, 'stonewall' presumably refers to the Stonewall Inn/Stonewall riots, which, according to Wikipedia, "is widely considered to be the single most important event leading to the gay liberation movement and the modern fight for gay and lesbian rights in the United States".[7]

Figure 7: Words Associated with 'gay' Over Time



[7] http://en.wikipedia.org/wiki/Stonewall_Inn

Conclusion

The Google Ngram viewer is fun to play with. This extension allows to analyst to observe the frequency of words associated with words of interest over time. There are many other potential interesting words to examine. Next, I'm most interested in seeing what words are associated with the word 'terrorist'. Will the 'IRA' or 'Libya' dominate the 1970's? When will 'Al-Qaeda' or 'Osama' appear?

Code

Pig Code

Below is the Pig code used to grab the 4 words before 'gay' used on EMR.

```
-- This program counts the 4 words before any word used in Google's Ngrams
-- Note that there one place below where you must place the word you want to study

-- If you want to do this as an interactive session, SSH into amazon
--ssh -i '/key_location.pem' hadoop@ec2...rest of key


-- Load in the necessary SequenceFileLoader UDF for the compressed data
REGISTER file:/home/hadoop/lib/pig/piggybank.jar;
DEFINE SequenceFileLoader org.apache.pig.piggybank.storage.SequenceFileLoader();
```

The Ngram data are stored on Amazon's S3 servers as block level LZO compressed data. Pig doesn't have a built-in function to read this type of data. Luckily, Twitter has made a UDF, SequenceFileLoader, available.[8] SequenceFileLoader is stored in Piggy Bank, which is automati-

[8] https://github.com/kevinweil/elephant-bird

cally loaded when Amazon loads Pig on Elastic MapReduce. All that needs to be done is use the 'REGISTER' command to tell Pig where the Piggy Bank is, and DEFINE SequenceFileLoader as the UDF we want to use.

```
-- Load Data
data = LOAD '$INPUT' USING SequenceFileLoader AS (noidea:int, ngram:chararray);
-- If you are using an interactive session
--data = LOAD 's3://datasets.elasticmapreduce/ngrams/books/20090715/eng-us-all/5gram/data'
USING SequenceFileLoader AS (noidea:int, ngram:chararray);
```

The data are stored as (counter, ngram).

```
data1 = foreach data generate FLATTEN(STRSPLIT(ngram, '\t'));

data2 = foreach data1 generate FLATTEN(STRSPLIT($0, ' '))
AS (f1:chararray, f2:chararray, f3:chararray, f4:chararray, f5:chararray) ,
$1 AS year:int, $2 AS count:int;

data2a = foreach data2 generate LOWER(f1)
AS f1, LOWER(f2) AS f2, LOWER(f3) AS f3, LOWER(f4) AS f4, LOWER(f5) AS f5, year, count;
```

The ngram data are tab separated, so data1 separates the ngram from the year and word counts. data2 splits the 5gram into 5 different parts. data2a takes each field in the ngram and makes it lowercase

```
-- Put the word of interest in the ' ' - this gives 4 words before
data3 = filter data2a by f5 matches 'gay';

--  Count all the words after our word of interest
grouped = GROUP data3 BY (f2, year);
counts2 = FOREACH grouped GENERATE group, SUM(data3.count) AS count2;

grouped = GROUP data3 BY (f3, year);
counts3 = FOREACH grouped GENERATE group, SUM(data3.count) AS count3;

grouped = GROUP data3 BY (f4, year);
counts4 = FOREACH grouped GENERATE group, SUM(data3.count) AS count4;

grouped = GROUP data3 BY (f1, year);
counts5 = FOREACH grouped GENERATE group, SUM(data3.count) AS count5;
```

data3 only keeps the grams that have the word 'gay' as the 5th element. counts2 groups data3 by the second element in the ngram and stores each time a word was used in that element in that year. counts3-5 does the same for other elements.

```
-------------------------------------------------------------------------------------------
-- Merge all the words together - This generates they word as a key, with
--  each column as a bag and the total number of counts at that position
all_data = cogroup counts2 BY $0, counts3 BY $0, counts4 BY$0, counts5 BY$0;
```

all_data merges the previously grouped counts. This stores each time a word was used in any position in the ngram in a specific year.

```
-------------------------------------------------------------------------------------------
-- Remove the bags (and make sure missing values don't screw everything up)
noempty = foreach all_data generate group,
 flatten(((counts2.count2 is null or IsEmpty(counts2.count2)) ? null : counts2.count2)),
 flatten(((counts3.count3 is null or IsEmpty(counts3.count3)) ? null : counts3.count3)),
 flatten(((counts4.count4 is null or IsEmpty(counts4.count4)) ? null : counts4.count4)),
 flatten(((counts5.count5 is null or IsEmpty(counts5.count5)) ? null : counts5.count5));
```

```
-- Turn the nulls into zeros so that pig can add them
noempty2 = foreach noempty generate group, (($1 is null) ? 0 : $1), (($2 is null) ? 0 : $2),
(($3 is null) ? 0 : $3), (($4 is null) ? 0 : $4);
```

the COGROUP command used to produce all_data creates bags. The bags must be flattened into tuples, but many bags have no values (because a words wasn't used in that particular year), so the syntax is a little ugly, telling Pig not to flatten empty bags.

noempty2 places a zero for any word that was not used in a particular year. This is necessary because in Pig null + 1 = null.

```
-----------------------------------------------------------------------------------------
-- This gives ((word,year),number of times word directly after,
-- number of times 2nd after, 3rd after, 4th after)
summed = foreach noempty2 generate flatten(group)
AS (z1,z2), $1+$2+$3+$4+$5+$6+$7+$8 AS summed;
```

summed gives the number of times a word was used in given year.

```
-----------------------------------------------------------------------------------------
-- Sort by number of summed references
grouped = GROUP summed by z1;
total_ordered = FOREACH grouped GENERATE group AS names, SUM(summed.summed) AS totals;


-- Merge in totals summed
```

```
merge1 = JOIN total_ordered BY names, summed BY z1;
merge2 = FOREACH merge1 GENERATE names, totals, z2, summed;

-----------------------------------------------------------------------------------------------
-- This gives us (word, total number of times used, year, number of times used that year)
final_output = ORDER merge2 BY totals DESC, names, z2;
```

total_ordered provides the total number of times a word was used over the entire time frame under study. final_output orders the words by the total number of times they were used.

```
-----------------------------------------------------------------------------------------------
store final_output INTO '$OUTPUT';
-- If using an interactive session
--store final_output INTO 's3://...';
```

The data are stored as (word, total number of times used, year, number of times used that year).

R Code

Once the data are merged in R, I counted the number of total words per year. This was used as a denominator to calculate the y-axis of the plots, which is (number of times a specific word is used in Year Y/number of total words in year Y).

```
# Generate a variable for the number of total words in a given year
count.data <- aggregate(data$sum, by=list(Category=data$v3), FUN=sum)
names(count.data)[names(count.data)=="Category"] <- "v3"
data2 <- merge(data,count.data, by="v3")
data2$percent <- data2$sum/data2$x
names(data2)[names(data2)=="Category"] <- "v3"



############################
# Plots

plot.data <- subset(data2, data2$v1 == "men")
ggplot(data=plot.data, aes(x=v3, y=percent)) +
geom_smooth(method="loess")+ labs(title = "Use of 'men' Over Time", x = "Year")
```