New York Times Article Abstract Analysis using Hadoop and NLTK

By: Lucas Nunno (lnunno@cs.unm.edu)

Part 1: Data Acquisition

Data acquisition

- Used the python **requests** module.
  - Used the offset parameter to load new pages of abstracts and slept 1/8th of a second between each request to abide by the NYT API terms of use.
- Loaded JSON response into python dictionary and then exported as a single large JSON file containing all the articles and all metadata. (**~40,000**)
- In a separate script, I export this JSON data to a CSV file with the docIDs, URLs, and abstracts.
  - This is also where I check for **duplicates**. I have a set of URLs that the exporter has seen, if this URL is in this set the program prints a warning and does not export it.

Part 2: Preprocessing and tf-idf

Preprocessing

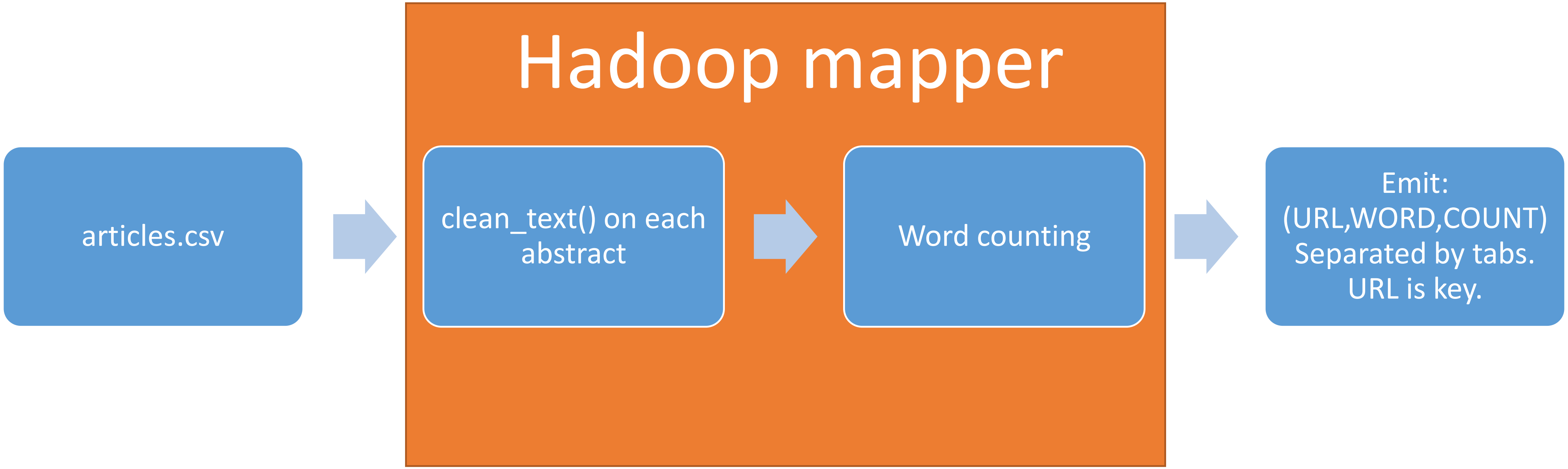
- Used the python natural language toolkit (NLTK) module for most of the preprocessing tasks. The algorithm is as follows:
  1. Convert text to lowercase.
  2. Remove punctuation and numbers.
     - Simple regex substitution:

     ```
     remove_pattern = re.compile(r'[^a-z\s]')
     ```
  3. Remove stopwords.
     - See: `nltk.corpus.stopwords`
  4. Stem all the remaining words.
     1. See: http://www.nltk.org/api/nltk.stem.html#module-nltk.stem.porter
  5. Output the cleaned abstract.

tf-idf is broken up into a couple of stages

1. Term frequency map reduce job.
2. Document frequency and calculating the total number of documents. Done as a separate map reduce job.

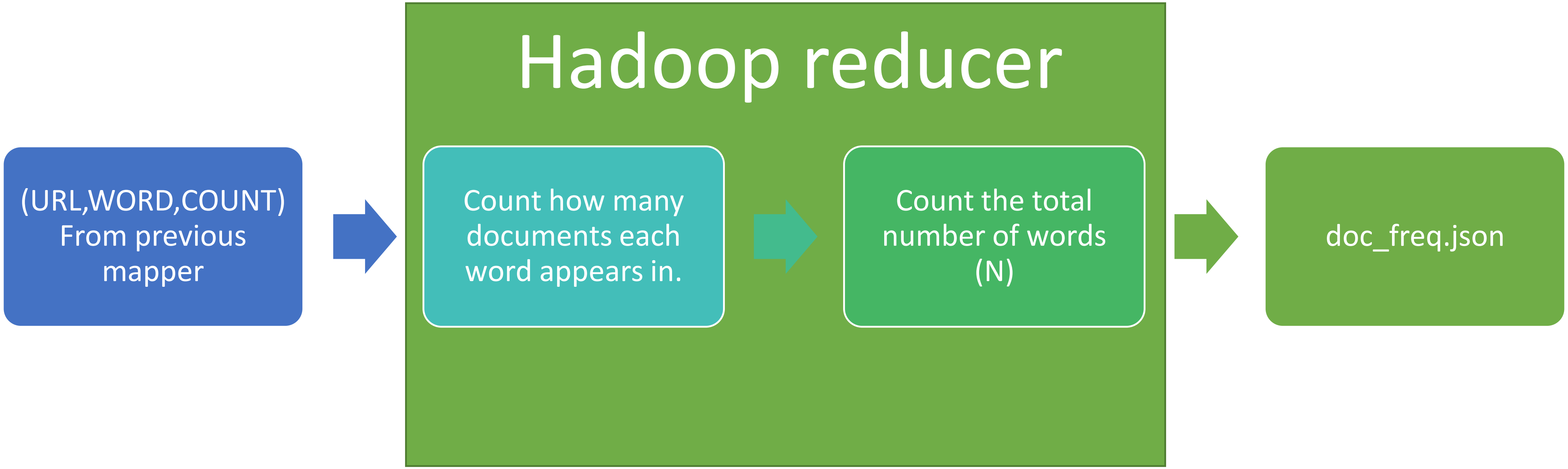
How is this parallelized in Hadoop? – Term frequency



Term frequency

- Dictionary is constructed of each document, the words it contains, and the frequencies of these words – this is all provided from the mapper.
- A pandas Series is constructed for all documents. This is sparse, because it specifies the entries which are non-zero.
- Augmented (normalized) term frequency vector is calculated.
  - $tf(t,d) = \frac{f(t,d)}{\Sigma f(t,d)\ \forall\ t \in d}$
  - This removes the bias for longer documents.

How is this parallelized in Hadoop? – Doc frequency

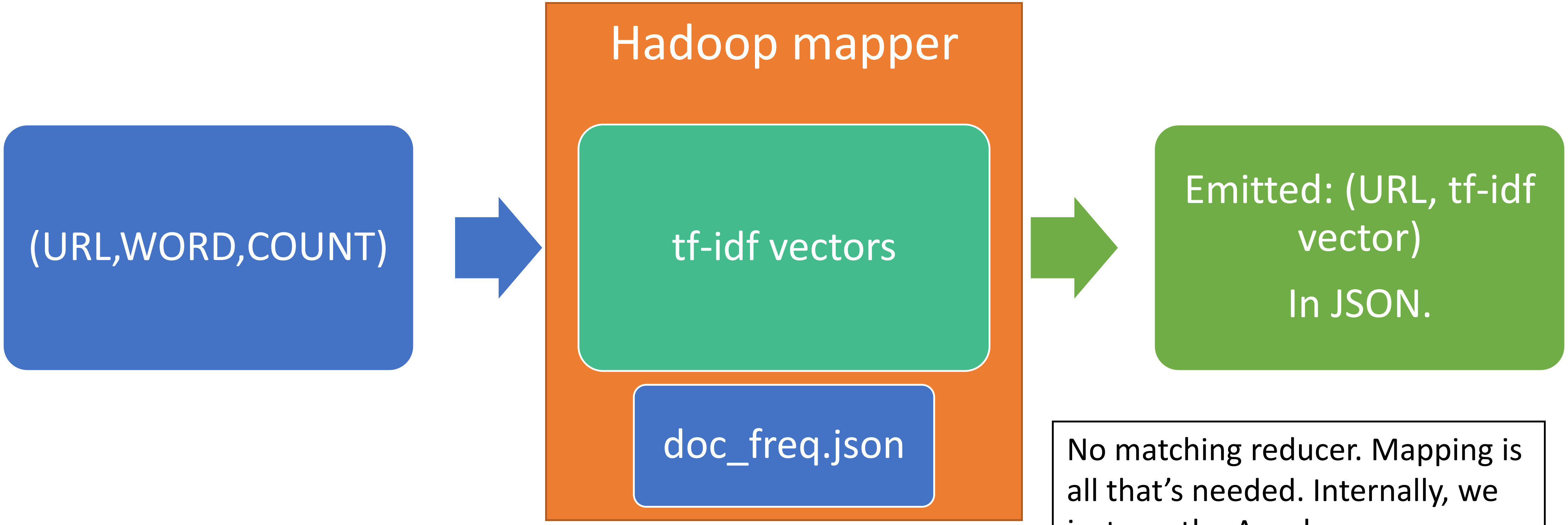


df and tf-idf

- Document frequency vector is calculated in a reducer by counting the number of unique documents that each term appears in.
- We then multiply each normalized term frequency vector with the idf vector to calculate each document’s tf-idf vector.
  - A special implementation detail is that we remove all nonzero entries and only output the information where the tf-idf score is zero. This saves a massive amount of space and time. Every tf-idf vector is very sparse.
- From these data structures we can calculate the inverse document frequency (idf) vector.

$$\mathrm{idf}(t, D) = \log \frac{N}{|\{d \in D : t \in d\}|}$$

How is this parallelized in Hadoop?



No matching reducer. Mapping is all that’s needed. Internally, we just use the Apache IdentityReducer in the Streaming API.

Part 3: Clustering and Visualization

So now that we have the tf-idf vectors, how do we cluster?

Clustering methodology

- Apache Spark makes it (relatively) easy to do distributed clustering.
- The API for Spark allows for a text file to be stored in a distributed manner.
- We use the previously calculated tf-idf vectors and transform them into Apche Spark's own SparseVectors.
- Spark's MLlib module is then used for distributed K-means clustering.
  - This uses the SparseVectors that we created and can rapidly separate the vectors into clusters.
- The centroids of the K-means model were analyzed to determine the most discriminative terms per cluster.

Why K-means?

- Aside from it being the only clustering algorithm available in Spark (which is a good reason), it is one of the most popular clustering algorithms available and used in production in many Big Data applications.
- The Mahout K-means algorithm was also considered, but Apache Spark offered an easier set-up and a Python API so it was used.

K-means pros and cons

- Can easily tweak the k value.
- Can easily tweak the number of iterations and see if this improves the solution.
- Implementation is simple and easy to understand.
- Will find a local error minimum.
- Widely used.

- Must pick the appropriate K value.
- Have to choose how many iterations to perform.
- Will probably not find the optimal solution.
- Sensitive to outliers.

How many clusters?

- If you look at the sections in the New York Times (right), there are around 20 primary sections of the paper.
- We initially started out at 20 clusters and then quantitatively analyzed how the number of clusters affected the results.

Categories

NEWS

NYT Front Page

International
- Africa
- Americas
- Asia Pacific
- Middle East
- Europe

National
- Education

New York Region

Politics
- Campaigns
- Congressional Guide
- Supreme Court Guide
- Governor Guide
- White House Guide
- Politics Navigator

Business
- Media & Advertising
- World Business
- Your Money
- Markets
- Company Research
- Mutual Funds

Technology
- E-Business
- Circuits

Science
- Earth Science
- Life Science
- Physical Science
- Social Science
- Space

Health
- Aging
- Anatomy
- Children
- Fitness
- Genetics
- Men
- Nutrition
- Policy
- Psychology
- Women

Arts
- Art & Design
- Dance
- Music
- Television
- Theater

Weather
- N.Y.C. Metro
- U.S. Regions
- International
- Travel Forecast

Sports
- Baseball
- Basketball, College
- Basketball, Pro
- Football, College
- Football, Pro
- Golf
- Hockey
- Other Sports
- Soccer
- Tennis

Obituaries

Editorials

Letters

Op-Ed

Corrections

FEATURES

Automobiles
- Auto Classifieds

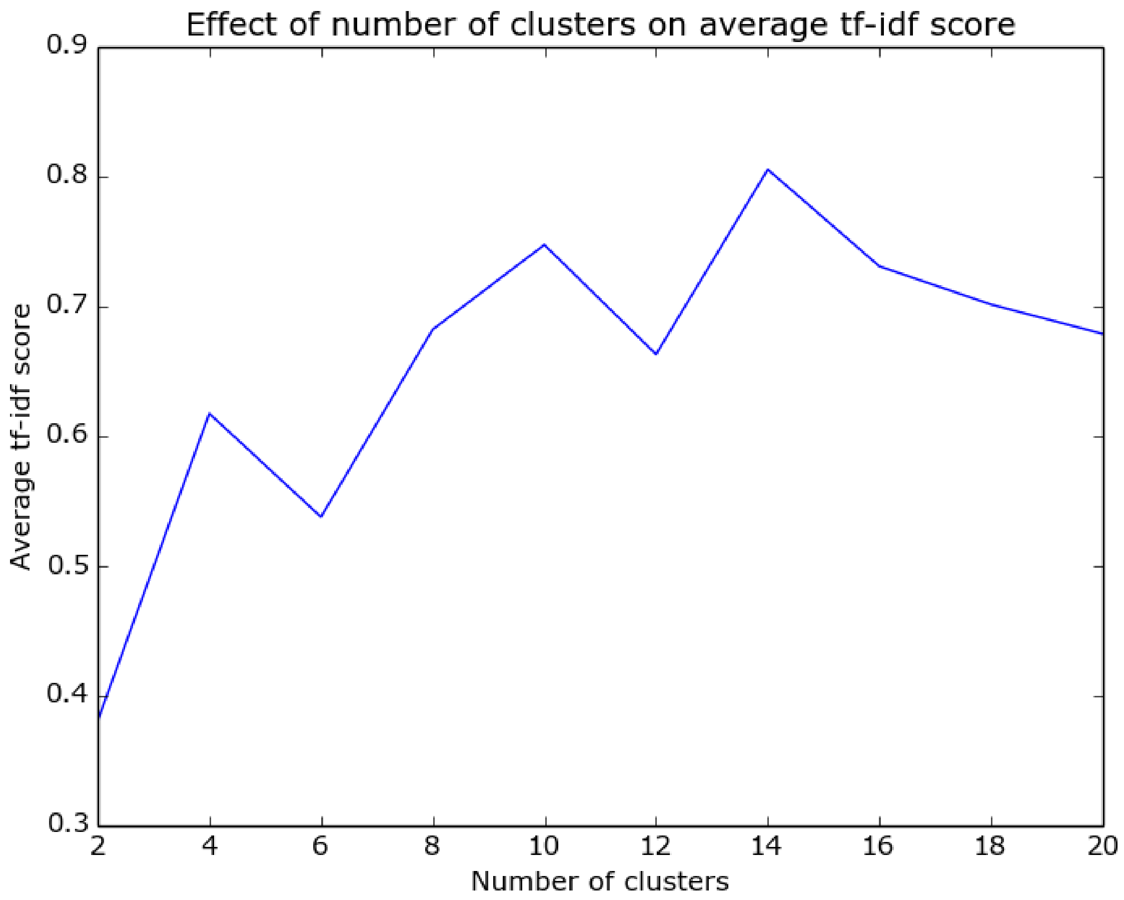
Travel

New York Today

College Times
- Students
- Faculty

Analysis on number of clusters

- Each cluster contains a vector of the 20 highest discriminative terms for that cluster.
- Notice how the average is maximized at 14 clusters.
- Due to these results, we will be using the 14 clusters found to have the highest average discriminative terms.



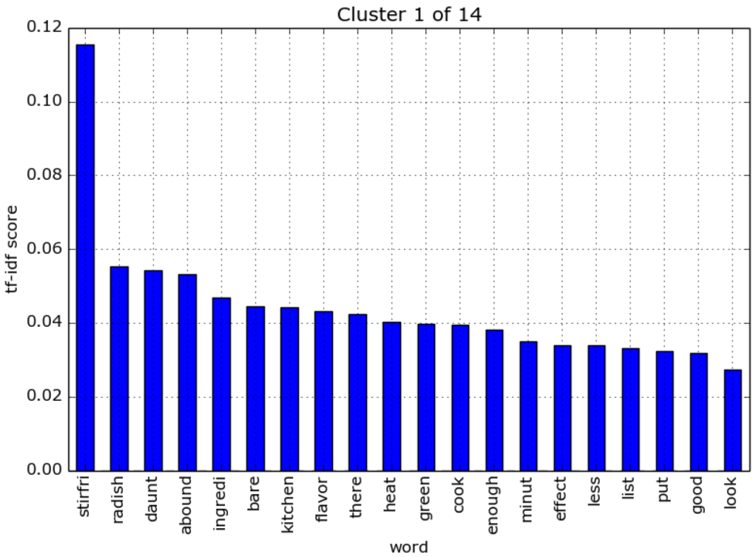
Things to consider when clustering

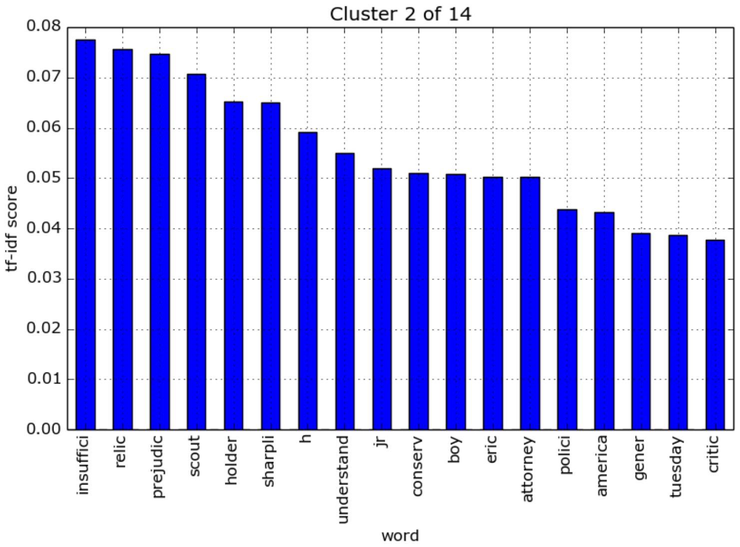
- Our sample size is relatively small (~40k) and not all sections may be represented sufficiently.
- Sampling methodology.
  - It is highly dependent on which articles were being produced the most at the time of collection. We did not discriminate on the types of articles found.
  - This is a combination of the limitations of the New York Times API and the methodology employed in this study.
- This may help to explain some of the results. Some clusters seem to be much more informative than others.

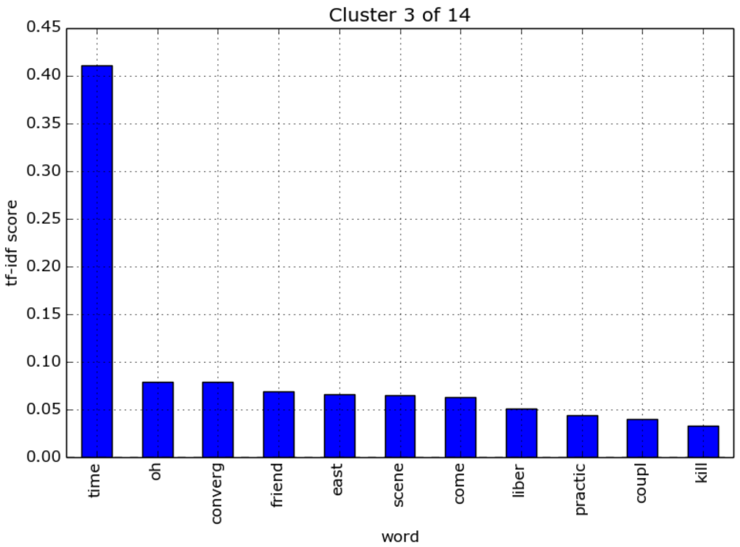
Visualization notes

- The top terms were selected based on their tf-idf value for that cluster. This was simply `row.nlargest(NUM_TERMS)` for each `DataFrame` row.
- Some clusters had less than 20 significant terms, the terms with zero tf-idf scores were eliminated, therefore some of the plots show less than 20 terms per cluster.
- This is an effect of the clustering algorithm used.

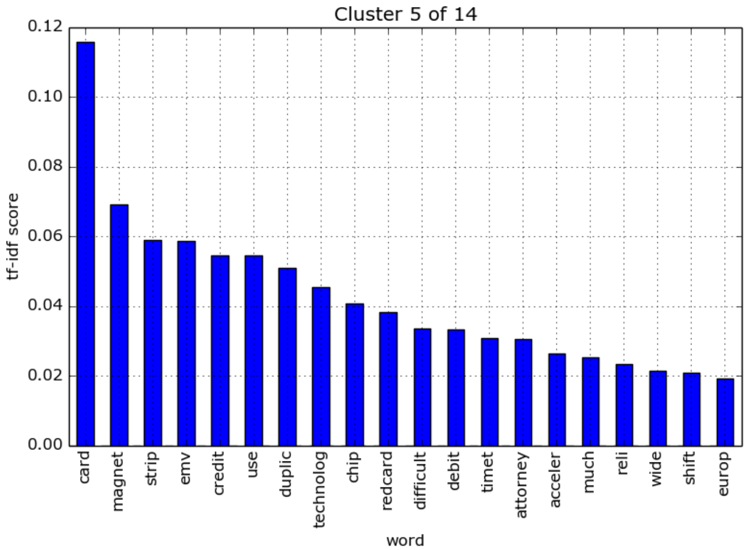
14 Cluster Plots

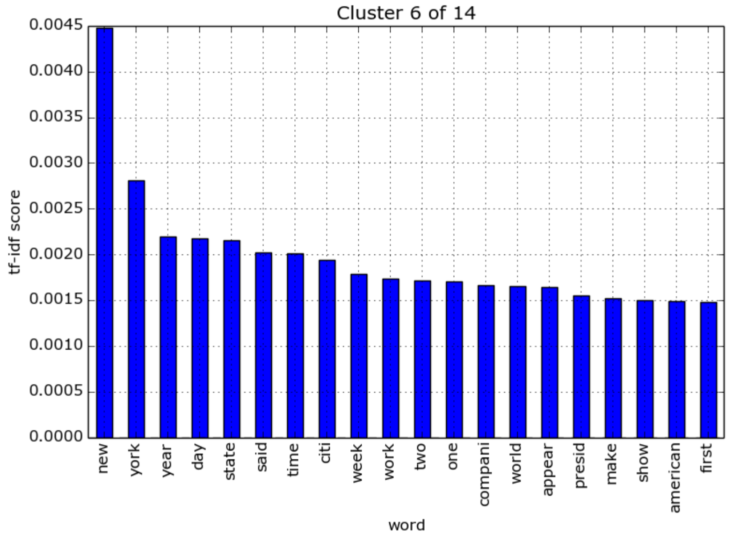

Cluster 1 of 14
tf-idf score
stirfri
radish
daunt
abound
ingredi
bare
kitchen
flavor
there
heat
green
cook
enough
minut
effect
less
list
put
good
look
word

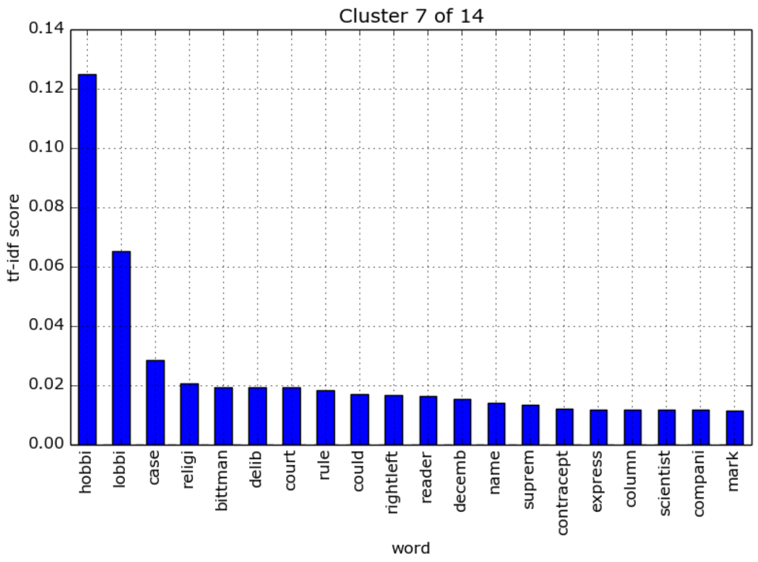

Cluster 2 of 14
tf-idf score
insuffici
relic
prejudic
scout
holder
sharpli
h
understand
jr
conserv
boy
eric
attorney
polici
america
gener
tuesday
critic
word

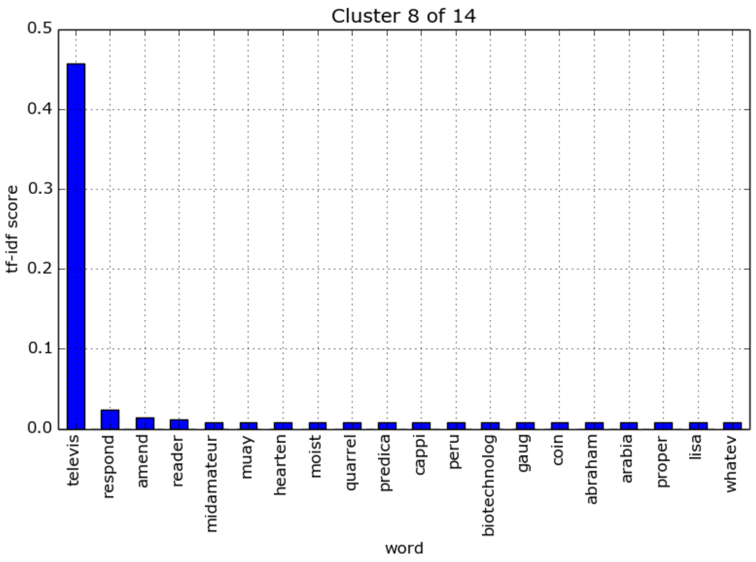

Cluster 3 of 14
tf-idf score
time
oh
converg
friend
east
scene
come
liber
practic
coupl
kill
word

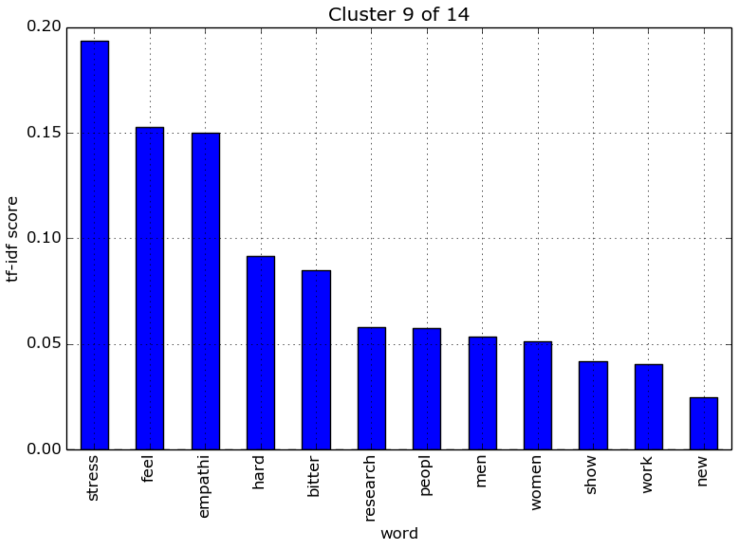

Cluster 4 of 14
tf-idf score
citi
word

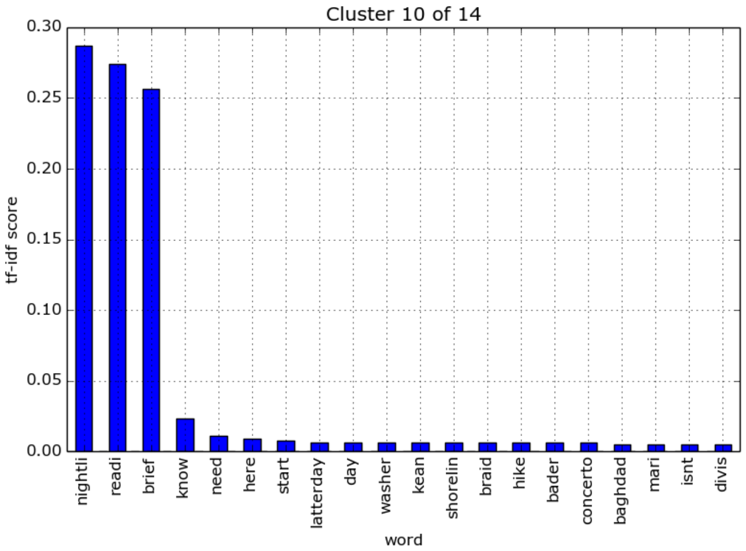

Cluster 5 of 14
tf-idf score
card
magnet
strip
emv
credit
use
duplic
technolog
chip
redcard
difficult
debit
timet
attorney
acceler
much
reli
wide
shift
europ
word

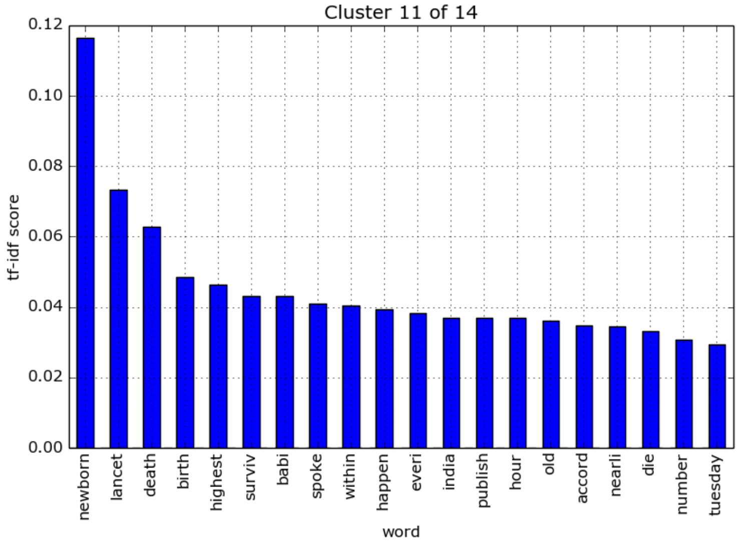

Cluster 6 of 14
tf-idf score
new
york
year
day
state
said
time
citi
week
work
two
one
compani
world
appear
presid
make
show
american
first
word

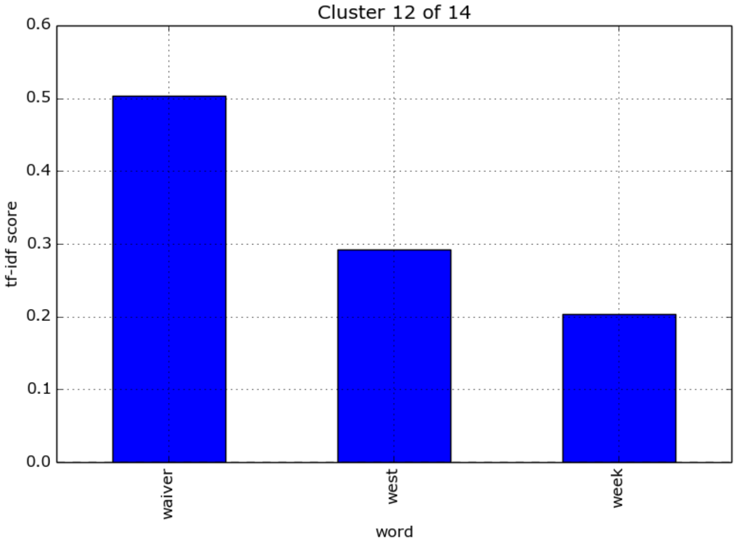

Cluster 7 of 14
tf-idf score
0.14
0.12
0.10
0.08
0.06
0.04
0.02
0.00
hobbi
lobbi
case
religi
bittman
delib
court
rule
could
rightleft
reader
decemb
name
suprem
contracept
express
column
scientist
compani
mark
word

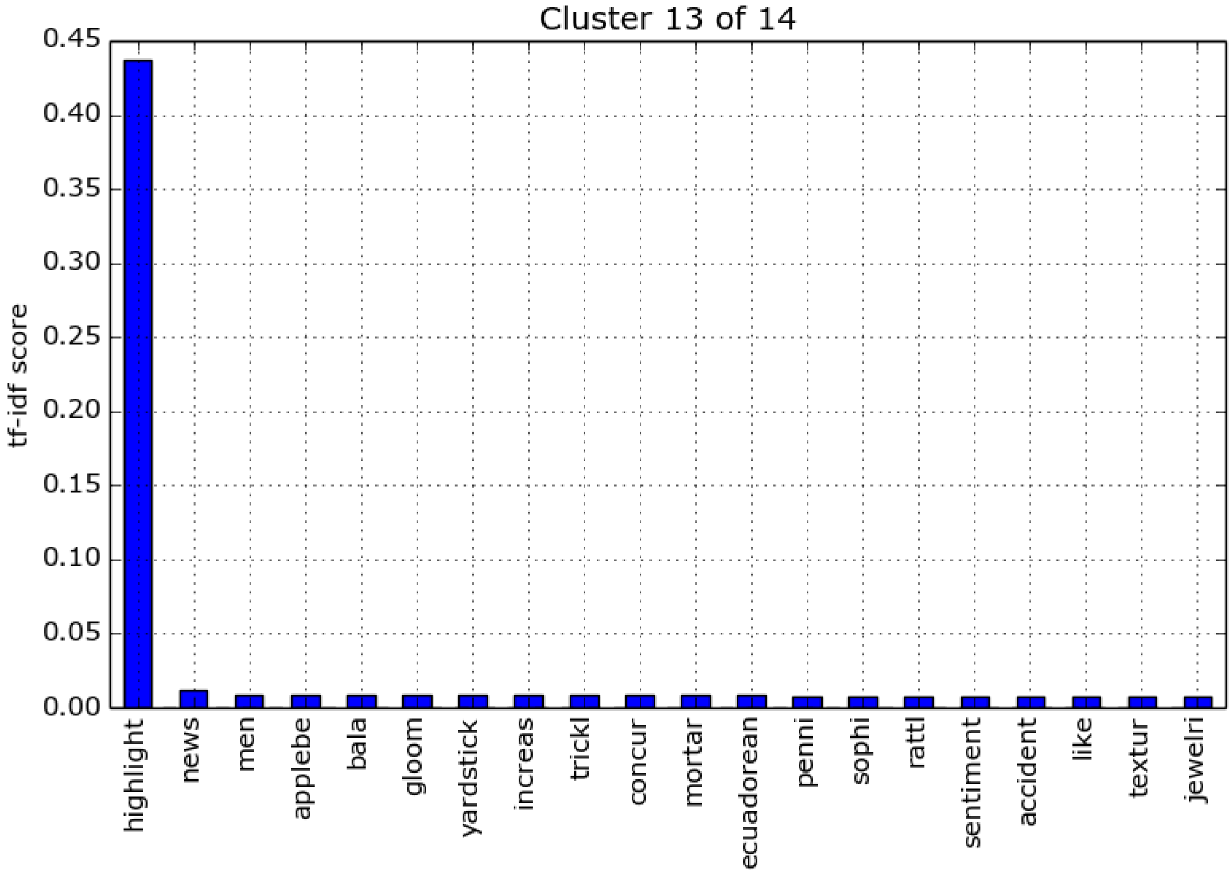

Cluster 8 of 14
tf-idf score
0.5
0.4
0.3
0.2
0.1
0.0
televis
respond
amend
reader
midamateur
muay
hearten
moist
quarrel
predica
cappi
peru
biotechnolog
gaug
coin
abraham
arabia
proper
lisa
whatev
word

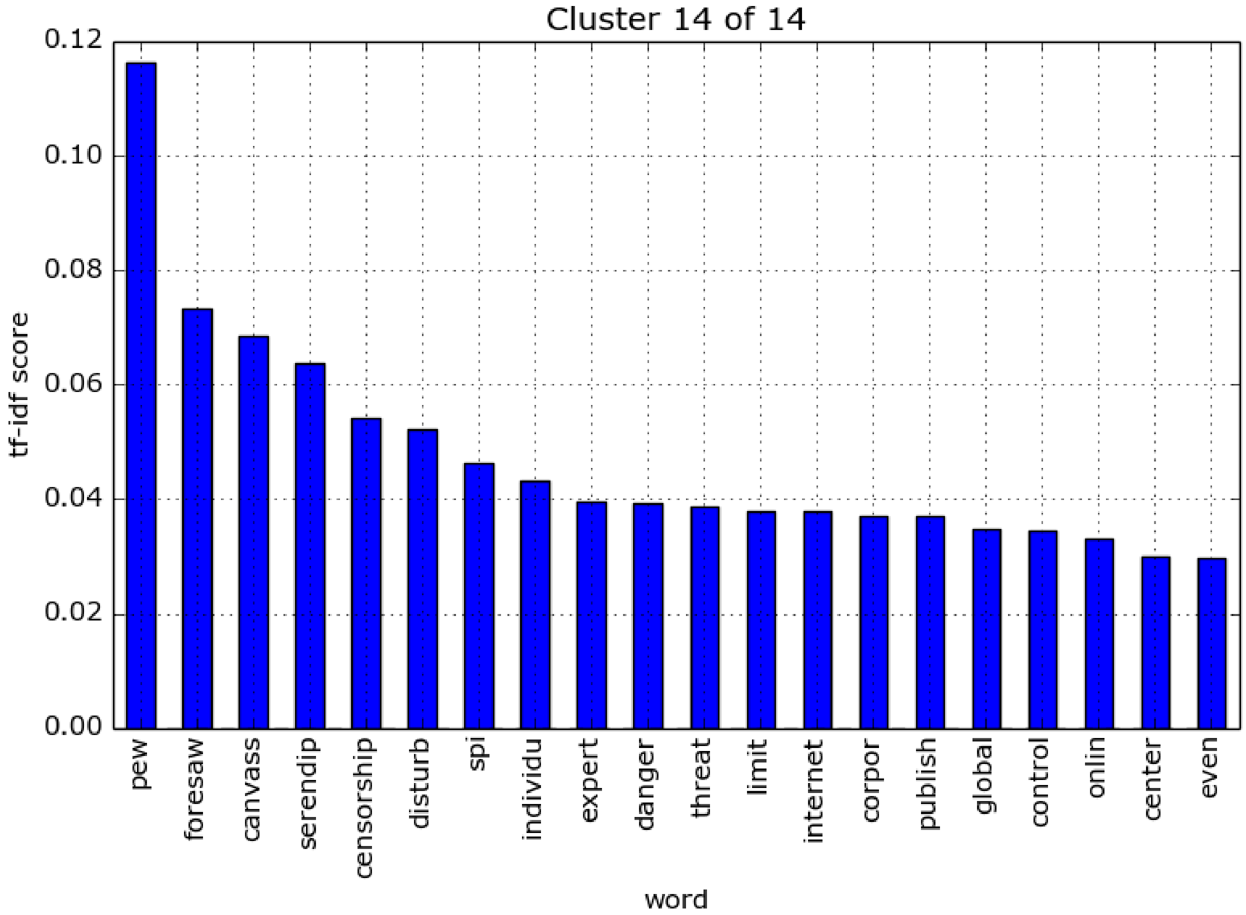

Cluster 9 of 14
tf-idf score
0.20
0.15
0.10
0.05
0.00
stress
feel
empathi
hard
bitter
research
peopl
men
women
show
work
new
word


Cluster 10 of 14
tf-idf score
0.30
0.25
0.20
0.15
0.10
0.05
0.00
nightli
readi
brief
know
need
here
start
latterday
day
washer
kean
shorelin
braid
hike
bader
concerto
baghdad
mari
isnt
divis
word


Cluster 11 of 14
tf-idf score
0.12
0.10
0.08
0.06
0.04
0.02
0.00
newborn
lancet
death
birth
highest
surviv
babi
spoke
within
happen
everi
india
publish
hour
old
accord
nearli
die
number
tuesday
word


Cluster 12 of 14
tf-idf score
0.6
0.5
0.4
0.3
0.2
0.1
0.0
waiver
west
week
word


Cluster 13 of 14
0.45
0.40
0.35
0.30
0.25
0.20
0.15
0.10
0.05
0.00
tf-idf score
highlight
news
men
applebe
bala
gloom
yardstick
increas
trickl
concur
mortar
ecuadorean
penni
sophi
rattl
sentiment
accident
like
textur
jewelri
word


Cluster 14 of 14
0.12
0.10
0.08
0.06
0.04
0.02
0.00
tf-idf score
pew
foresaw
canvass
serendip
censorship
disturb
spi
individu
expert
danger
threat
limit
internet
corpor
publish
global
control
onlin
center
even
word

14 Clusters Selected Word Clouds

These were created by hand at http://www.wordle.net/compose using the tf-idf scores as weights. Words with higher weights are larger.

Most interesting included, some left out for brevity.

Cooking cluster?


critic
boy
america
prejudic
conserv
scout
gener
eric
h
sharpli
relic
polici
attorney
jr
understand
insuffici
tuesday
holder


time
converg
oh
practic
coupl
come
friend
scene
east
liber
kill

card
magnet
strip
credit
duplic
emv
technolog
use
chip
redcard
attorney
acceler
debit
timet
difficult
reli
shift
much
europ
wide

Local news?


new
york
said
make
time
work
citi
week
show
compani
presidtwo
appear
year
state
american
world
one
day
first


hobbi
lobbi
case
decemb
column
rightleft
delib
could
bittman
mark
express
scientist
suprem
reader
religi
contracept
court
compani
name
rule


televis
respond
reader
amend
cappi
cole

Lifestyle & health cluster?


readi
brief
nightli
know


number
nearli
birth
everi
babi
spoke
tuesday
within
lancet
happen
die
death
hour
accord
publish
newborn
highest
surviv
old
india


limit
disturb
onlin
expert
center
serendip
censorship
even
individu
pew
corpor
control
global
spi
canvass
publish
danger
foresaw
threat
internet

Thank you.

Check out the source on Github: https://github.com/lnunno/big-data-nyt-tf-idf