Explorations in Named Entity Recognition, and was Eleanor Roosevelt right?

tds towardsdatascience.com/explorations-in-named-entity-recognition-and-was-eleanor-roosevelt-right-671271117218

Mikael Davidsson July 17, 2019

Image credit: unsplash

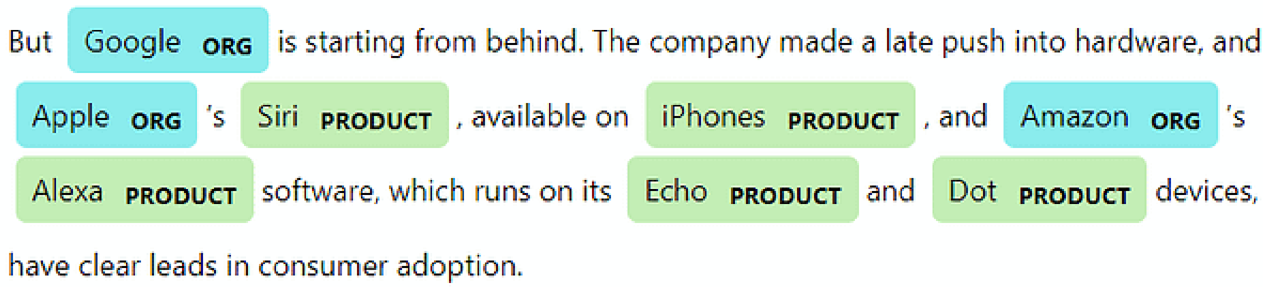
Eleanor Roosevelt is alleged to have said:

> *Great minds discuss ideas; average minds discuss events; small minds discuss people.*

And although this might be a misattribution, the statement as such seems to resonate with a lot of people's intuition, but how true is it? Does it stand up to scrutiny?

There are many ways in which this could be investigated, one fun approach might be to look through a bunch of newspapers for **ideas**, **events** and **people** and see if the fraction in which they appear can be correlated to the "mind size" (great, average, small) of its readers.

To mine the newspaper articles for information, I decided to use a natural language processing technique called Named Entity Recognition (NER), which is used to identify something called "named entities" in a sentence. Named entities are things such as products, countries, companies, numbers. I will use the spaCy natural language processing lib for this. Here's an example from their documentation of how NER-tagging can look:



spaCy recognizes the following entities:

As can be seen we have PERSON and EVENT, but IDEA is sorely lacking. To rectify this we will need to choose one of the others to take the role as a proxy for ideas. For this I have chosen PERCENT. The reason for this is that percent usually is a way to describe abstract ideas, one uses it when talking about for example humanity as a whole, instead of this or that person. It is not a perfect map from ideas but we have to work with what we got.

As for the "mind size" of the readers, I'm going to go with the Coleman-Liau readability index. This is a way to quantify at what education level the reader must be to understand the text and is calculated using the formula:

```
Coleman_Liau = 0.0588*L–0.296*S-15.8L = Average number of letters per 100 charactersS
= Average number of sentences per 100 characters
```

Again, the analogy is not perfect, but hopefully good enough.

Looks like we have our methodology all set up, lets start working with the data!

Acquiring and cleaning the data

We are going to use the news feed provided by a free subscription to newsapi.org. This means that we will get a **Title**, a **Description** (a summary of the article) and the first 260 characters of the **Content** of the article. I decided to pick a few popular English language newspapers mostly from USA and England:

```
sources = ['abc-news', 'cnn', 'fox-news', 'cbs-news', 'the-new-york-times',
'reuters', 'the-wall-street-journal', 'the-washington-post', 'bloomberg', 'buzzfeed',
'bbc-news', 'daily-mail']
```

After getting the data (13,368 articles from June 2019) I inspected it and found that there were a few articles in Chinese and Arabic that will cause problems for spaCy. I cleaned it using a function I found on StackOverflow:

```
latin_letters= {}def is_latin(uchr):    try:         return latin_letters[uchr]
except KeyError:        try:             return latin_letters.setdefault(
uchr, 'LATIN' in ud.name(uchr))        except:            print(uchr)
raise Exception()def only_roman_chars(unistr):    return all(is_latin(uchr) for uchr
in unistr if uchr.isalpha())
```

After cleaning we have 11458 posts left, distributed over the different sources:

```
df.groupby('source').count()['title']abc-news                  1563bbc-news
1076bloomberg                    56buzzfeed                  295cbs-news
780cnn                         809daily-mail                1306fox-news
1366reuters                    916the-new-york-times        1467the-wall-street-
journal     590the-washington-post        1234
```

I decided to use **Description** as basis for the NER tagging since we want to tag the article based on what it's about, and Description seemed to be the best fit for that.

For Coleman-Liau we will use **Content** since that better reflect the overall writing style of the article.

Now when that is done we can start extracting our entities:

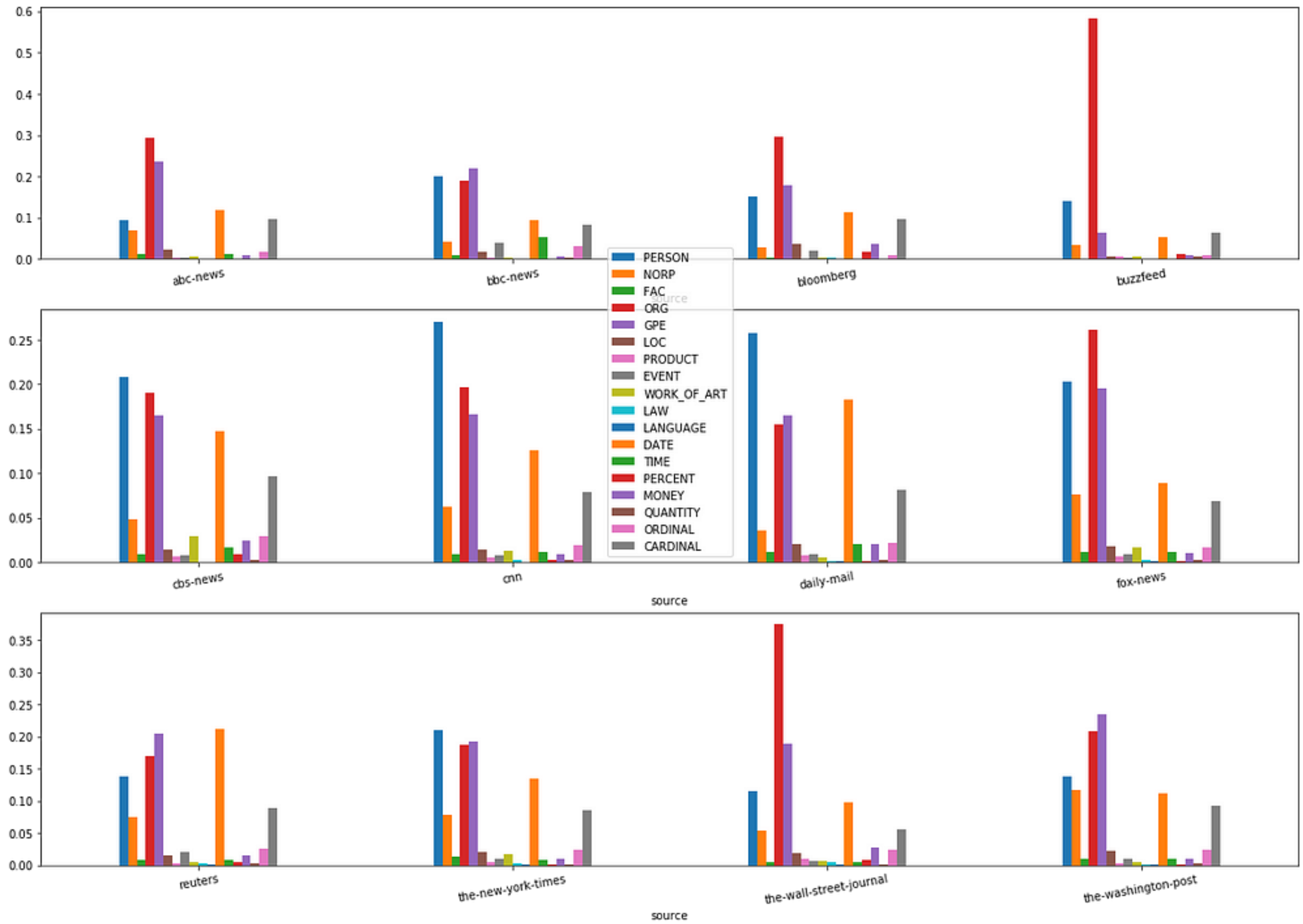
```
ners =
['PERSON','NORP','FAC','ORG','GPE','LOC','PRODUCT','EVENT','WORK_OF_ART','LAW','LANGUA
 The ners we are most interested inners_small = ['PERSON', 'EVENT', 'PERCENT']nlp =
spacy.load("en_core_web_sm")df['ner'] = df['Description'].apply(lambda desc:
dict(Counter([ent.label_ for ent in nlp(desc).ents])))for ner in ners:    df[ner] =
df['ner'].apply(lambda n: n[ner] if ner in n else 0)
```

We group them by source and normalize them:

```
df_grouped_mean = df.groupby('source').mean()# Normalize df_grouped =
df_grouped_mean[ners].div(    df_grouped_mean[ners].sum(axis=1),
axis=0)df_grouped['coleman_content'] = df_grouped_mean['coleman_content']# Do the
same for the smaller ners-setdf_grouped_small = df_grouped_mean[ners_small].div(
df_grouped_mean[ners_small].sum(axis=1), axis=0)df_grouped_small['coleman_content'] =
df_grouped_mean['coleman_content']
```

Looking at the result

```
fig, axes = plt.subplots(nrows=3, ncols=1)df_grouped[ners].iloc[:4].plot(kind='bar',
figsize=(20,14), rot=10, ax=axes[0],
legend=False);df_grouped[ners].iloc[4:8].plot(kind='bar', figsize=(20,14), rot=10,
ax=axes[1]);df_grouped[ners].iloc[8:].plot(kind='bar', figsize=(20,14), rot=10,
ax=axes[2], legend=False);
```



This bar plot might be a bit hard to interpret so let's look at the different news sources focus areas, or the entities that they have the most of:

```
focus = []

for source in df_grouped[ners].values:
    focus.append(sorted([(ners[i],x) for i,x in enumerate(source)], key=lambda x:
x[1], reverse=True)[:3])

        df_grouped['focus'] = [' '.join([y[0] for y in x]) for x in
focus]df_grouped['focus']abc-news                          ORG GPE DATEbbc-news
GPE PERSON ORGbloomberg                      ORG GPE PERSONbuzzfeed
ORG PERSON CARDINALcbs-news                       PERSON ORG GPEcnn
PERSON ORG GPEdaily-mail                    PERSON DATE GPEfox-news
ORG PERSON GPEreuters                          DATE GPE ORGthe-new-york-times
PERSON GPE ORGthe-wall-street-journal        ORG GPE PERSONthe-washington-post
GPE ORG PERSON
```

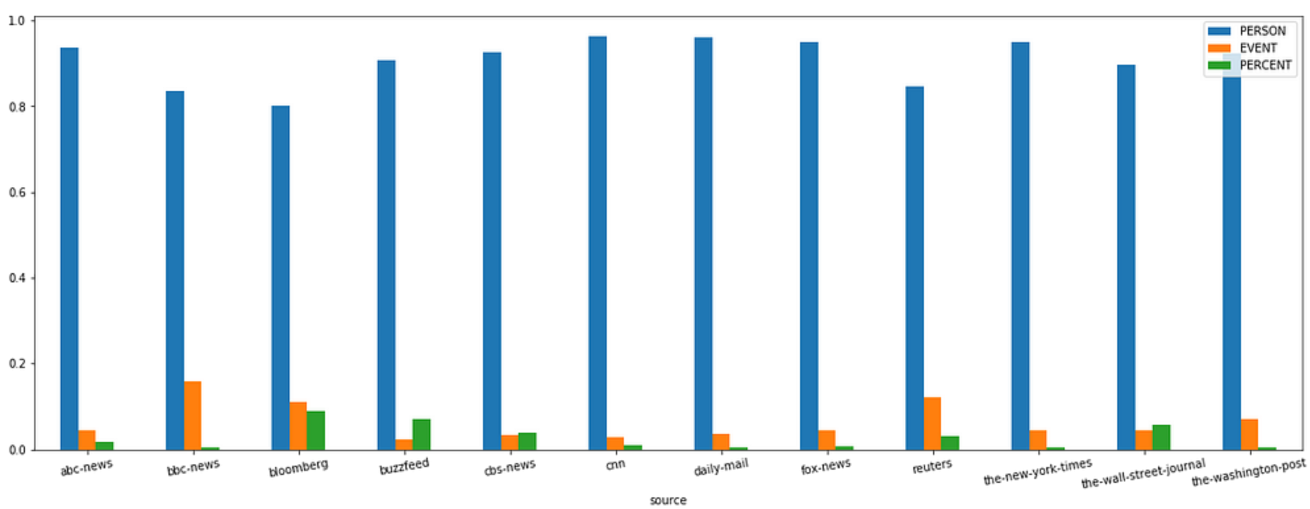
And also, let's list the news sources that have the largest fraction in a certain topic:

```
largest_in_topic = {}for n in ners:    largest_in_topic[n] = list(df_grouped.sort_values(n,ascending=False).index[:3])largest_in_topic{'PERSON': ['cnn', 'daily-mail', 'the-new-york-times'], 'NORP': ['the-washington-post', 'the-new-york-times', 'fox-news'], 'FAC': ['the-new-york-times', 'abc-news', 'fox-news'], 'ORG': ['buzzfeed', 'the-wall-street-journal', 'bloomberg'], 'GPE': ['abc-news', 'the-washington-post', 'bbc-news'], 'LOC': ['bloomberg', 'abc-news', 'the-washington-post'], 'PRODUCT': ['the-wall-street-journal', 'daily-mail', 'buzzfeed'], 'EVENT': ['bbc-news', 'bloomberg', 'reuters'], 'WORK_OF_ART': ['cbs-news', 'fox-news', 'the-new-york-times'], 'LAW': ['bloomberg', 'the-wall-street-journal', 'cnn'], 'LANGUAGE': ['bbc-news', 'fox-news', 'the-new-york-times'], 'DATE': ['reuters', 'daily-mail', 'cbs-news'], 'TIME': ['bbc-news', 'daily-mail', 'cbs-news'], 'PERCENT': ['bloomberg', 'buzzfeed', 'cbs-news'], 'MONEY': ['bloomberg', 'the-wall-street-journal', 'cbs-news'], 'QUANTITY': ['buzzfeed', 'bbc-news', 'cnn'], 'ORDINAL': ['bbc-news', 'cbs-news', 'reuters'], 'CARDINAL': ['bloomberg', 'cbs-news', 'abc-news']}
```

There are a few interesting things to notice here:

- Almost everyone likes to talk about countries, companies and persons.
- The Wall Street Journal and Bloomberg likes money and organizations, just as expected.
- Reuters likes to be precise about dates.

If we only look at the smaller NER-set we get:



Ok, so that looks good. It’s time to calculate the Coleman-Liau index. For this we need to be able to split into sentences, which is a harder task than one might suspect. I will use a function from StackOverflow:

```
import re
alphabets= "([A-Za-z])"
prefixes = "(Mr|St|Mrs|Ms|Dr)[.]"
suffixes = "(Inc|Ltd|Jr|Sr|Co)"
starters = "
(Mr|Mrs|Ms|Dr|He\s|She\s|It\s|They\s|Their\s|Our\s|We\s|But\s|However\s|That\s|This\s|
acronyms = "([A-Z][.][A-Z][.](?:[A-Z][.])?)"
websites = "[.](com|net|org|io|gov)"

def split_into_sentences(text):    text = " " + text + "  "    text =
text.replace("\n"," ")    text = re.sub(prefixes,"\\1<prd>",text)    text =
re.sub(websites,"<prd>\\1",text)    if "Ph.D" in text: text =
text.replace("Ph.D.","Ph<prd>D<prd>")    text = re.sub("\s" + alphabets + "[.] ","
\\1<prd> ",text)    text = re.sub(acronyms+" "+starters,"\\1<stop> \\2",text)    text
= re.sub(alphabets + "[.]" + alphabets + "[.]" + alphabets + "
[.]","\\1<prd>\\2<prd>\\3<prd>",text)    text = re.sub(alphabets + "[.]" + alphabets
+ "[.]","\\1<prd>\\2<prd>",text)    text = re.sub(" "+suffixes+"[.] "+starters,"
\\1<stop> \\2",text)    text = re.sub(" "+suffixes+"[.]"," \\1<prd>",text)    text =
re.sub(" " + alphabets + "[.]"," \\1<prd>",text)    if "”" in text: text =
text.replace(".”","”.")    if "\"" in text: text = text.replace(".\"","\".")    if
"!" in text: text = text.replace("!\"","\"!")    if "?" in text: text =
text.replace("?\"","\"?")    text = text.replace(".",".<stop>")    text =
text.replace("?","?<stop>")    text = text.replace("!","!<stop>")    text =
text.replace("<prd>",".")    sentences = text.split("<stop>")    sentences =
sentences[:-1]    sentences = [s.strip() for s in sentences]    return sentences
```

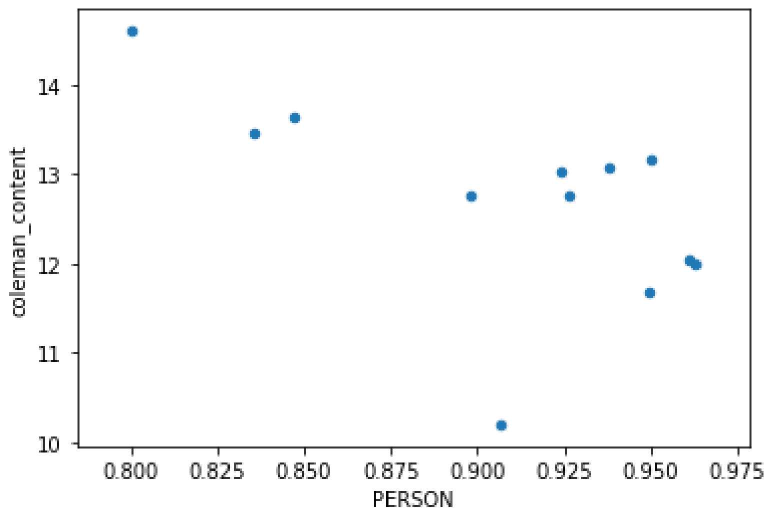
Do the calculation:

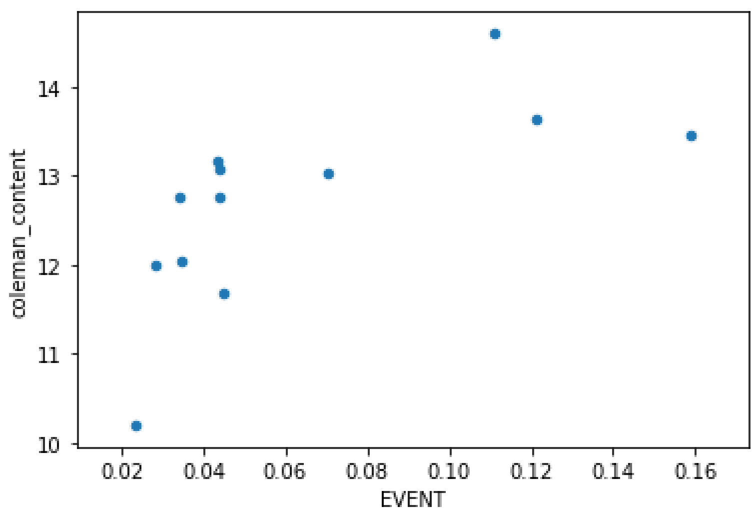
```
def calculate_coleman(letter_count, word_count, sentence_count):    return 0.0588 *
letter_count*100/word_count - 0.296 *              sentence_count*100/word_count -
15.8df['coleman'] = df['split_content'].apply(lambda x: calculate_coleman(    len('
'.join(x).replace(' ', '').replace('.', '')),    len([y for y in '
'.join(x).replace('’', '').split() if not y.isnumeric()]),
len(x)))df_grouped['coleman'].sort_values(ascending=False)bloomberg
14.606977reuters                   13.641115bbc-news                  13.453002fox-
news                 13.167492abc-news                  13.076667the-washington-
post        13.025180the-wall-street-journal    12.762103cbs-news
12.753429daily-mail                12.030524cnn                       11.988568the-
new-york-times         11.682979buzzfeed                  10.184662
```

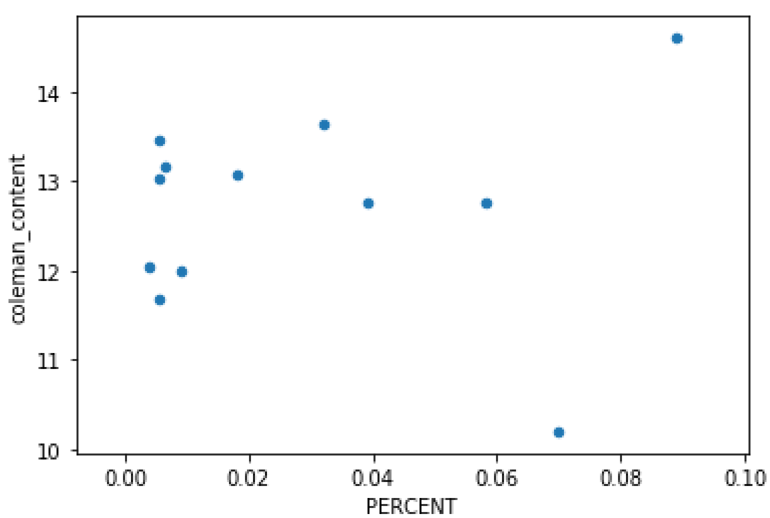
This is a bit surprising; I would have expected The New York Times to be higher up for example, but then on the other hand it just might be right. It would probably be more accurate if I had more than 260 chars of content, but the next tier of newsapi is $449/month. Just to be sure I will double check against an external source for readability score later on.

Looking for a correlation

Let’s plot the readability against people, event and percent:







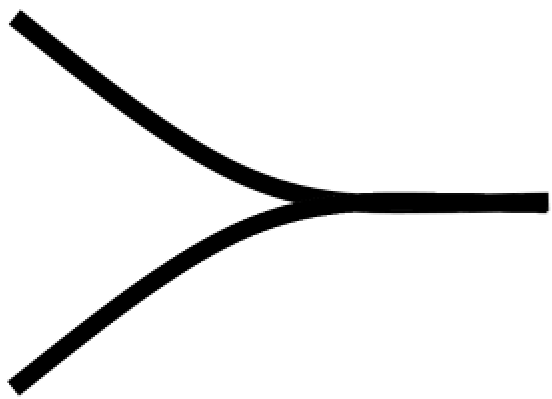
Interestingly there actually seems to be a bit of correlation, at least on PERSON and EVENT. Let’s calculate the correlation score:

```
df_grouped_small.corr()
```

	PERSON	EVENT	PERCENT	coleman_content
PERSON	1.000000	-0.835785	-0.586153	-0.598271
EVENT	-0.835785	1.000000	0.045052	0.677805
PERCENT	-0.586153	0.045052	1.000000	0.088344
coleman_content	-0.598271	0.677805	0.088344	1.000000

Looking at the coleman_content column there might actually be something to the Eleanor Roosevelt quote! At least insofar that there is a negative correlation between Coleman-Liau and PERSON and a positive one between Coleman-Liau and EVENT.

Since EVENT is supposed to be for “average” minds we would expect the scatter plot to move to the middle for high EVENT values, like so:



This is not really what we see though, but the negative/positive correlation for PERSON/EVENT still lend some credibility to the quote.

Of course, this is to be taken with a bucket load of salt. Apart from all the concession we have made so far we don’t have nearly enough samples to reach statistical significance. In fact, let’s look at the p value (function from StackOverflow):

```
from scipy.stats import pearsonrdef calculate_pvalues(df):    df =
df.dropna()._get_numeric_data()    dfcols = pd.DataFrame(columns=df.columns)
pvalues = dfcols.transpose().join(dfcols, how='outer')    for r in df.columns:
for c in df.columns:            pvalues[r][c] = round(pearsonr(df[r], df[c])[1], 4)
return pvaluescalculate_pvalues(df_grouped_small)
```

	PERSON	EVENT	PERCENT	coleman_content
PERSON	0	0.0007	0.0452	0.0399
EVENT	0.0007	0	0.8894	0.0154
PERCENT	0.0452	0.8894	0	0.7848
coleman_content	0.0399	0.0154	0.7848	0

As expected, the p-values are low, except for PERCENT.

Since the calculated Coleman-Liau levels seemed to be a bit off I decided to test with the following readability levels, taken from http://www.adamsherk.com/publishing/news-sites-google-reading-level/

```
reading_level = {'abc-news': (41,57,1),'cnn': (27,69,2),   'fox-news':
(23,73,2),'cbs-news': (28,70,0),   'the-new-york-times': (7,85,7),'reuters':
(6,85,7),   'the-wall-street-journal': (9,88,2),   'the-washington-post':
(24,72,2),'bloomberg': (6,81,11)}
```

They give 3 values (Basic, Intermediate, Advanced) which I gave different weights (-1,0,1) to calculate a singe value.

```
df_grouped_small['external_reading_level'] = df_grouped_small.index.map(    lambda x:
reading_level[x][2]-reading_level[x][0] if x in reading_level else 0)
```

Looking at the correlation

```
df_grouped_small[df_grouped_small['external_reading_level'] != 0][ners_small +
['external_reading_level']].corr()
```

	PERSON	EVENT	PERCENT	external_reading_level
PERSON	1.000000	-0.883805	-0.823875	-0.857877
EVENT	-0.883805	1.000000	0.462978	0.775646
PERCENT	-0.823875	0.462978	1.000000	0.685644
external_reading_level	-0.857877	0.775646	0.685644	1.000000

We find that the correlation is similar to what we got before, except that we actually have an even higher positive correlation with PERCENT.

Conclusion

Our results indicate that there actually might be some truth to the quote, but the statistical significance is so low that further research is needed. Also, it turns out that no matter the size of the mind, people love to talk about other people, a lot. Even the brainiest news (Bloomberg, with a whooping 14.6 Coleman-Liau level) talks about people 7 times more than it talks about events or percent.

Another thing that stands out looking at the bar plots is how similar the newspapers are in their choice of content. So even though there are differences in peoples interests, ultimately we are more similar than we are different.