NLP Final Project

Matthew Dunne

The Assignment

Illinois is famous for being one of the very few states in the country with negative population growth. The objective of your final project is to:

1. Identify the key reasons for the declining population by extracting meaningful insights from unstructured text
2. Provide actionable recommendations on what can be done to reverse this trend

Loading the Data

In [2]:

```
import pandas as pd
import nltk as nltk
import numpy as np
import re
import pickle
porter = nltk.PorterStemmer()
lancaster = nltk.LancasterStemmer()
import sklearn
from sklearn.model_selection import train_test_split
from sklearn.feature_extraction.text import CountVectorizer, HashingVectorizer, TfidfTransformer, TfidfVectorizer
from sklearn.naive_bayes import MultinomialNB
from sklearn.metrics import classification_report, confusion_matrix, accuracy_score
from sklearn.linear_model import LogisticRegression, SGDClassifier
from sklearn import metrics
```

In [3]:

```
df = pd.read_pickle("news_chicago_il.pkl")
```

In [5]:

```
df.head()
```

Out[5]:

	crawled	language	text	title
0	2018-08-24T00:00:46.000+03:00	english	Illinois to Offer New Merit-Based College Scho...	Illinois to Offer New Merit-Based College Scho...
1	2018-08-24T00:01:09.008+03:00	english	GX_GR110 Springfield, IL Thu, Aug 23, 2018 USD...	USDA Chicago Terminal Grain - Aug 23
2	2018-08-24T00:01:09.041+03:00	english	Duncan James to replace Martin Kemp in West En...	Duncan James to replace Martin Kemp in West En...
3	2018-08-24T00:01:09.060+03:00	english	Nick Givas, DCNF\nChicago Democratic Mayor Rah...	Embattled Chicago mayor threatens Trump over i...
4	2018-08-24T00:02:44.002+03:00	english	Second man convicted in killing of Chicago hon...	Prosecutors: Overwhelming evidence man killed ...

In [259]:

```
alt_df = pd.read_pickle("news_chicago_il.pkl")
alt_df['duplicate']=alt_df.duplicated(subset='title', keep='first').reset_index(drop=True)
duplicates=alt_df[alt_df['duplicate']==True]
duplicates.head()
```

Out[259]:

	crawled	language	text	title
22	2018-08-24T00:21:44.009+03:00	english	Illinois Sen. Dick Durbin Discusses His Meeting With Brett Kavanaugh August 23, 2018 -- Copyright 2018 NPR. To see more, visit NPR . Thanks to our Sponsors	Illinois Sen. Dick Durbin Discusses His Meeting With Brett Kavanaugh
23	2018-08-24T00:22:53.027+03:00	english	Illinois Sen. Dick Durbin Discusses His Meeting With Brett Kavanaugh By editor • 38 minutes ago Listen	Illinois Sen. Dick Durbin Discusses His Meeting With Brett Kavanaugh
26	2018-08-24T00:24:04.015+03:00	english	Chicago 'anti-bait truck' event to offer free shoes The Canadian Press Published August 23, 2018 - 5:08pm Last Updated August 23, 2018 - 5:25pm\nCHICAGO — A Chicago rapper's foundation will be giv...	Chicago 'anti-bait truck' event to offer free shoes
27	2018-08-24T00:26:17.001+03:00	english	Illinois Sen. Dick Durbin Discusses His Meeting With Brett Kavanaugh By editor • 27 minutes ago Listen	Illinois Sen. Dick Durbin Discusses His Meeting With Brett Kavanaugh
28	2018-08-24T00:28:05.045+03:00	english	Illinois Sen. Dick Durbin Discusses His Meeting With Brett Kavanaugh By editor • 46 minutes ago Listen	Illinois Sen. Dick Durbin Discusses His Meeting With Brett Kavanaugh

Drop the columns we don't need.

In [260]:

```
df=df.drop(['crawled', 'language'], axis=1)
```

We will assume that articles with the same title have substantially the same text. We will drop the duplicates of title. Also, articles on population decline are not likely to be duplicated as they are not especially sensational, and so there is little benefit from knowing how many such articles there are relative to other subjects.

In [261]:

```
df=df.drop_duplicates(subset='title').reset_index(drop=True)
len(df)
```

Out[261]:

```
58520
```

But set aside the duplicates of articles (by looking for duplication of title).

In [262]:

```
alt_df = pd.read_pickle("news_chicago_il.pkl")
alt_df['duplicate']=alt_df.duplicated(subset='title', keep='first').reset_index(drop=True)
duplicates=alt_df[alt_df['duplicate']==True]
len(duplicates)
```

Out[262]:

```
21954
```

Create a sample of 100 rows that you can use to see if your code is working.

In [263]:

```
sample=df.sample(n=100, random_state=634)
```

Filtering the Data

Many of these articles are not useful to our analysis. In addition to taking out duplicates, we will filter out articles where we can guess that underlying article does not relate to the issue at hand, but is about something else. These would include:
-Chicago sports teams (Cubs, Bears, Bulls, Blackhawks, Sox)
-Famous people associated with Chicago/Illinois
-Things we see in the title where we know it the article falls out of scope, e.g. job posting

In [20]:

```
#the stemmers you will try
porter = nltk.PorterStemmer()
lancaster = nltk.LancasterStemmer()
```

In [265]:

```
key_words=['cubs', 'bears', 'bulls', 'blackhawks', 'football', 'baseball', 'basketball', 'hockey', 'sox', 'Obama', 'Kanye', 'Kardashian', 'Durbin', 'Trump'
, '(USA-IL-Chicago)', 'vs', 'vs.', 'Illinois)', 'Chicago)']
key_p_stems=[porter.stem(t) for t in key_words]
key_l_stems=[lancaster.stem(t) for t in key_words]
print(key_p_stems)
print(key_l_stems)
```

```
['cub', 'bear', 'bull', 'blackhawk', 'footbal', 'basebal', 'basketbal', 'hockey', 'sox', 'obama', 'kany', 'kardashian', 'durbin', 'trump', '(usa-il-chicago)', 'vs'
'illinois)', 'chicago)']
['cub', 'bear', 'bul', 'blackhawk', 'footbal', 'basebal', 'basketbal', 'hockey', 'sox', 'obam', 'kany', 'kardash', 'durbin', 'trump', '(usa-il-chicago)', 'vs', 'vs
inois)', 'chicago)']
```

In [266]:

```
porter_stems=[]
for a in range(0, len(df)):
    words=df.iloc[a].title
    stems=[porter.stem(t) for t in words.split()]
    #if any of the keywords are in the title (stems), put those stems in the list
    if any(s in stems for s in key_p_stems):
        porter_stems.append(stems)
    #if none of the keywords are present insert a Null value
    else:
        porter_stems.append(None)
```

Do the same for the duplicates:

In [267]:

```
dup_porter_stems=[]
for a in range(0, len(duplicates)):
    words=duplicates.iloc[a].title
    stems=[porter.stem(t) for t in words.split()]
    #if any of the keywords are in the title (stems), put those stems in the list
    if any(s in stems for s in key_p_stems):
        dup_porter_stems.append(stems)
    #if none of the keywords are present insert a Null value
    else:
        dup_porter_stems.append(None)
```

How many articles have titles with these words in them?

In [268]:

```
pop_articles=[el for el in porter_stems if el is not None]
len(pop_articles)
```

Out[268]:

```
13550
```

Take these articles out of the data. They almost certainly have nothing to do with Illinois' negative population growth. Do it for the duplicates too.

In [269]:

```
df['Off_Topic']=pd.Series(porter_stems)
```

In [270]:

```
df=df[df['Off_Topic'].isnull()]
df=df.reset_index(drop=True)
len(df)
```

Out[270]:

```
44970
```

In [271]:

```
duplicates['Off_Topic']=pd.Series(dup_porter_stems)
duplicates=duplicates[duplicates['Off_Topic'].isnull()]
len(duplicates)
```

```
C:\Users\mjdun\Anaconda\lib\site-packages\ipykernel_launcher.py:1: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/pandas-docs/stable/indexing.html#indexing-view-versus-copy
  """Entry point for launching an IPython kernel.
```

Out[271]:

```
20820
```

So we have narrowed down the articles we are looking at (not the duplicates) to about 45,000 articles. How many of these have to do with negative population growth. We shall look at which articles have: 'economy', 'shrink', 'amazon', 'outmigration', 'move' in their **text**.

To do this, we will "clean up" the text of each article and ask whether certain words are present.

We will need to clean the data further.

- replace \n (newlines) with spaces
- cut off the footer that many of these articles have (signalled by the terms 'MORE COVERAGE' which lists other articles)

In [272]:

```
df['text_clean'] = df['text'].map(lambda x: re.sub('\n', ' ', str(x)))
df['text_clean'] = df['text_clean'].map(lambda x: str(x).split('MORE COVERAGE')[0])
```

Cast a wide net and see which articles discuss population or migration *anywhere in the text of the article*.

In [275]:

```
key_words=['population', 'migration']
in_stems=[porter.stem(c) for c in key_words]
new_stems=[]

for a in range(0, len(df)):
    words=df.iloc[a].text
    stems=[porter.stem(t) for t in words.split()]
    #if any of the keywords are in the title (stems), put those stems in the list
    if any(s in stems for s in in_stems):
        new_stems.append(a)
    #if none of the keywords are present insert a Null value
    else:
        continue
```

In [278]:

```
df=df.iloc[new_stems]
```

In [279]:

```
df=df.reset_index(drop=True)
len(df)
```

Out[279]:

```
898
```

This gets us down to about 900 articles.

Then we will look at the text of the article to see if it has some other keywords (created by calibration).

In [280]:

```
in_terms=['economy', 'shrink', 'amazon', 'outmigration', 'revitalize', 'move']
in_stems=[porter.stem(c) for c in in_terms]
in_index=[]

for a in range(0, len(df)):
    words=df.iloc[a].text
    stems=[porter.stem(t) for t in words.split()]
    #if any of the keywords are in the title (stems), put those stems in the list
    if any(s in stems for s in in_stems):
        in_index.append(a)
    #if none of the keywords are present insert a Null value
    else:
        continue
```

In [281]:

```
relevant_articles=df.iloc[in_index]
relevant_articles=relevant_articles.reset_index(drop=True)
len(relevant_articles)
```

Out[281]:

```
346
```

There are about 350 articles in this new, filtered dataframe. Let us try to take away some more by looking at words in the title. If we review the titles of these articles, which is easier when there are only 230 as opposed to 80,000, we notice a few areas that should not be included. One can identify these just on the title. Take out the articles with these particular words in the title.

In [285]:

```
out_terms=['Trump', 'VoteCast', '$', 'Illinois House', 'Illinois Senate', 'Republican', 'Democrat', '#jobs', 'Homes for Sale', 'Home for Sale', 'bird', 'ca
t ', 'snake', 'Flight', 'flight', 'Air', 'deal', 'offer']
out_index=[]
for r in range(0, len(relevant_articles)):
    if any(p in relevant_articles.iloc[r].title for p in out_terms):
        out_index.append(r)
    else:
        continue
relevant_articles=relevant_articles.drop(out_index)
len(relevant_articles)
```

Out[285]:

```
285
```

In [347]:

```
out_terms=['World War I', 'WWI', 'History', 'Medicaid', 'illegal', 'Migrant', 'refugee', 'Muslim', 'Immigrant', 'immigrant', 'GOP', 'Military', 'North', 'B
lack', 'Laquan', 'fish', 'CD Duplication', 'Festival', 'Zoo', 'River', 'Lake', 'Mayor', 'mayor']
out_index=[]
for r in range(0, len(relevant_articles)):
    if any(p in relevant_articles.iloc[r].title for p in out_terms):
        out_index.append(r)
    else:
        continue
relevant_articles=relevant_articles.reset_index(drop=True)
in_articles=relevant_articles.drop(out_index).reset_index(drop=True)
out_articles=relevant_articles.iloc[out_index].reset_index(drop=True)
len(in_articles)
```

Out[347]:

```
209
```

There are a little over 200 articles left. At this point we will just eyeball each title and classify it as in our analysis or out. We can review the titles of each of these articles and say with great confidence they are about Illinois' negative population growth. Also, this is necessary for our Topic Modeling, because if we do not further limit here the resulting topics are not especially helpful.

In [348]:

```
onpoint=[0,3, 4, 9, 11, 16, 19, 27, 29, 31, 33, 35, 36, 37, 38, 42, 43, 44, 46, 51, 57, 60, 61, 62, 65, 66, 67, 72, 75, 80, 85, \
         89, 90, 91, 92, 93, 97, 103, 104, 109, 110, 111, 112, 118, 119, 120, 127, 131, 132, 133, 134, 136, 141, 142, 143, 145, \
         147, 148, 151, 158, 161, 162, 165, 166, 171, 174, 185, 186, 187, 191, 193, 194, 196, 202, 205, 208]
on_point=in_articles.iloc[onpoint].reset_index(drop=True)
on_point.head()
```

Out[348]:

	text	title	Off_Topic	
0	Proviso Township, Illinois\n Located just nine miles west of the city of Chicago, Proviso Township is one of 29 townships in Cook County, Illinois. An urban community, the township spans just 30 ...	Spotlight Stories: Proviso Township, Illinois	None	Proviso Township, Illinois Located just nine miles west of the city of C Proviso Township is one of 29 townships in Cook County, Illinois. An community, the township spans just 30 s...
1	By Mary Hansen • 37 minutes ago Photo by Tom Zittergruen on Unsplash\nFarm towns in Illinois have been shrinking for decades, and the trend doesn't show signs of reversing.\nBy 2025, rural countie...	For Shrinking Illinois Towns, A 'Smarter' Approach	None	By Mary Hansen • 37 minutes ago Photo by Tom Zittergruen on Unsp towns in Illinois have been shrinking for decades, and the trend doesr signs of reversing. By 2025, rural counties ...
2	By Cole Lauterbach Illinois News Network (ILLINOIS NEWS NETWORK)			By Cole Lauterbach Illinois News Network (ILLINOIS NEWS NETWO

	text	title	Off_Topic	
2	By Cole Lauterbach Illinois News Network (ILLINOIS NEWS NETWORK) – Wisconsin's top economist is using Illinois as an example to highlight how well Wisconsin's economy is doing. John Koskinen is th...	Official compares Wisconsin economy to Illinois	None	By Cole Lauterbach Illinois News Network (ILLINOIS NEWS NETWORK) – Wisconsin's top economist is using Illinois as an example to highlight how well Wisconsin's economy is doing. John Koskinen is th...
3	Chicago, Illinois Travel Guide - Must-See Attractions published: 26 Apr 2013 Chicago, Illinois Travel Guide - Must-See Attractions Chicago, Illinois Travel Guide - Must-See Attractions published: ...	Dome Energy gives production update for July 2018 and commences third round of drilling in the Illinois Basin	None	Chicago, Illinois Travel Guide - Must-See Attractions published: 26 Ap Chicago, Illinois Travel Guide - Must-See Attractions Chicago, Illinois Guide - Must-See Attractions published: ...
4	(202) 734-6543 | "Illinois Commitment" a Critical Step Towards Affordable College for Low-Income Illinois Students Chicago – Earlier today, the University of Illinois announced a new program aimed...	"Illinois Commitment" a Critical Step Towards Affordable College for Low-Income Illinois Students	None	(202) 734-6543 | "Illinois Commitment" a Critical Step Towards Afford College for Low-Income Illinois Students Chicago – Earlier today, the of Illinois announced a new program aimed...

Now we have under 200 articles, much more manageable for our purposes. Furthermore, we can be confident that a much greater percentage of these articles deal with population decline than from the full 80,000.

Put to pickle for Later Use

In [304]:

```
in_articles.to_pickle('my_df.pickle')
on_point.to_pickle('on_point.pickle')
```

In [305]:

```
del in_articles
del on_point
```

In [326]:

```
#articles=pd.read_pickle('my_df.pickle')
#articles=articles.reset_index(drop=True)
articles=pd.read_pickle('on_point.pickle')
```

In [327]:

```
len(articles)
```

Out[327]:

```
76
```

Topic Modeling

In [53]:

```
import time
import math
import re
from textblob import TextBlob
import pandas as pd

import nltk as nltk
from nltk.corpus import stopwords
from nltk.stem.wordnet import WordNetLemmatizer

import string

import warnings
warnings.filterwarnings(action='ignore', category=UserWarning, module='gensim')

import gensim
from gensim import corpora, models
from gensim.models.ldamulticore import LdaMulticore
import pyLDAvis.gensim
```

Cleaning and Preparing the Data

We will need to clean the data further.

- replace \n (newlines) with spaces
- cut off the footer that many of these articles have (signalled by the terms 'MORE COVERAGE' which lists other articles)

Then we will get the data ready for the LDA model by:

- taking out the standard English stopwords plus some others that are essentially stopwords in this context (Chicago, Illinois, etc.)
- taking out punctuation
- lemmatizing

In [331]:

```
doc_complete=articles['text_clean'].tolist()
```

In [332]:

```
extra_stops=['chicago', 'illinois', 'city', 'said', 'say', 'year', 'trump']
```

In [333]:

```
stop = set(stopwords.words('english'))
stop.update(extra_stops)
```

In [334]:

```
stop = set(stopwords.words('english'))
stop.update(extra_stops)
exclude = set(string.punctuation)
lemma = WordNetLemmatizer()
def clean(doc):
    stop_free = " ".join([i for i in doc.lower().split() if i not in stop])
    punc_free = ''.join(ch for ch in stop_free if ch not in exclude)
    normalized = " ".join(lemma.lemmatize(word) for word in punc_free.split())
    return normalized

doc_clean = [clean(doc).split() for doc in doc_complete]
```

Then we create a dictionary from our corpus, and then convert each record into a record in a Document Term Matrix.

In [335]:

```
dictionary = corpora.Dictionary(doc_clean)
doc_term_matrix = [dictionary.doc2bow(doc) for doc in doc_clean]
```

The LDA Models

In [336]:

In [336]:

```
# Creating the object for LDA model using gensim library
Lda = gensim.models.ldamodel.LdaModel
```

In [337]:

```
ldamodel = Lda(doc_term_matrix, num_topics=5, id2word = dictionary, passes=50)
print(*ldamodel.print_topics(num_topics=5, num_words=3), sep='\n')
```

```
(0, '0.007*"caterpillar" + 0.007*"school" + 0.006*"student"')
(1, '0.010*"population" + 0.008*"home" + 0.007*"new"')
(2, '0.010*"rauner" + 0.010*"pritzker" + 0.008*"state"')
(3, '0.014*"school" + 0.011*"student" + 0.008*"state"')
(4, '0.010*"tax" + 0.010*"home" + 0.007*"nursing"')
```

In [338]:

```
print(*ldamodel.print_topics(num_topics=10, num_words=10), sep='\n\n')
```

```
(0, '0.007*"caterpillar" + 0.007*"school" + 0.006*"student" + 0.006*"year" + 0.005*"one" + 0.005*"—" + 0.005*"enrollment" + 0.004*"university" + 0.004*"police" +
0.004*"also"')

(1, '0.010*"population" + 0.008*"home" + 0.007*"new" + 0.007*"city" + 0.007*"people" + 0.007*"tax" + 0.007*"housing" + 0.006*"cost" + 0.005*"state" + 0.005*"change

(2, '0.010*"rauner" + 0.010*"pritzker" + 0.008*"state" + 0.006*"tax" + 0.006*"-" + 0.005*"campaign" + 0.005*"million" + 0.005*"governor" + 0.004*"county" + 0.004*"

(3, '0.014*"school" + 0.011*"student" + 0.008*"state" + 0.008*"community" + 0.005*"education" + 0.005*"teacher" + 0.005*"would" + 0.004*"college" + 0.004*"universi
004*"also"')

(4, '0.010*"tax" + 0.010*"home" + 0.007*"nursing" + 0.006*"state" + 0.005*"property" + 0.005*"teacher" + 0.005*"school" + 0.005*"said" + 0.004*"percent" + 0.004*"c
```

In [339]:

```
lda_display = pyLDAvis.gensim.prepare(ldamodel, doc_term_matrix, dictionary, sort_topics=False)
pyLDAvis.display(lda_display)
```

Out[339]:

TF-IDF Models

First we have to define the functions we will use.

In [340]:

```
def tf(word, blob):
    return blob.words.count(word) / len(blob.words)
# tf(word, blob) computes "term frequency" which is the number of times a word appears in a document blob,
# normalized by dividing by the total number of words in blob (corpus). We use TextBlob for breaking up the text into words
# and getting the word counts.


def n_containing(word, bloblist):
    return sum(1 for blob in bloblist if word in blob.words)
# n_containing(word, bloblist) returns the number of documents containing word.
# A generator expression is passed to the sum() function.


def idf(word, bloblist):
    return math.log(len(bloblist) / (1 + n_containing(word, bloblist)))
# idf(word, bloblist) computes "inverse document frequency" which measures how common a word is
# among all documents in bloblist. The more common a word is, the lower its idf.
# We take the ratio of the total number of documents to the number of documents containing word,
# then take the log of that. Add 1 to the divisor to prevent division by zero


def tfidf(word, blob, bloblist):
    return tf(word, blob) * idf(word, bloblist)
# tfidf(word, blob, bloblist) computes the TF-IDF score. It is simply the product of tf and idf.
```

Then make some changes to our data: (1) make a column where all the text is lower, (2) create a "bloblist".

In [341]:

```
articles['text_lower']=articles['text_clean'].str.lower()
```

In [342]:

```
bloblist = []
del bloblist[:]

for i  in range(0,len(articles)):
    bloblist.append(TextBlob(articles['text_lower'].iloc[i]))
len(bloblist)
```

Out[342]:

```
76
```

In [343]:

```
for i, blob in enumerate(bloblist):
# Print top 5 values
    if i == 5:
        break
    print("Top words in news article {}".format(i + 1))
    scores = {word: tfidf(word, blob, bloblist) for word in blob.words}
    sorted_words = sorted(scores.items(), key=lambda x: x[1], reverse=True)
    for word, score in sorted_words[:10]:
        print("\tWord: {}, TF-IDF: {}".format(word, round(score, 5)))
```

```
Top words in news article 1
 Word: proviso, TF-IDF: 0.04605
 Word: township, TF-IDF: 0.03581
 Word: community, TF-IDF: 0.03256
 Word: pp4h, TF-IDF: 0.0307
 Word: organizations, TF-IDF: 0.01914
 Word: health, TF-IDF: 0.01826
 Word: obesity, TF-IDF: 0.01818
 Word: academy, TF-IDF: 0.01818
 Word: prevention, TF-IDF: 0.01535
 Word: partners, TF-IDF: 0.01531
Top words in news article 2
 Word: towns, TF-IDF: 0.02391
 Word: zarecor, TF-IDF: 0.02208
 Word: smart, TF-IDF: 0.01787
 Word: iowa, TF-IDF: 0.01652
 Word: shrinking, TF-IDF: 0.01539
 Word: says, TF-IDF: 0.01477
 Word: amenities, TF-IDF: 0.0134
 Word: factors, TF-IDF: 0.01231
 Word: merrett, TF-IDF: 0.01104
 Word: rural, TF-IDF: 0.01086
```

```
Top words in news article 3
 Word: wisconsin, TF-IDF: 0.0716
 Word: koskinen, TF-IDF: 0.0252
 Word: businesses, TF-IDF: 0.02447
 Word: economist, TF-IDF: 0.0168
 Word: using, TF-IDF: 0.01405
 Word: network, TF-IDF: 0.0136
 Word: draw, TF-IDF: 0.01257
 Word: doing, TF-IDF: 0.01102
 Word: he, TF-IDF: 0.0101
 Word: labor, TF-IDF: 0.00985
Top words in news article 4
 Word: span, TF-IDF: 0.08287
 Word: div, TF-IDF: 0.06446
 Word: class, TF-IDF: 0.04147
 Word: travel, TF-IDF: 0.03978
 Word: title, TF-IDF: 0.03775
 Word: must-see, TF-IDF: 0.01842
 Word: attractions, TF-IDF: 0.01842
 Word: bookinghunter.com, TF-IDF: 0.01842
 Word: pier, TF-IDF: 0.01842
 Word: li, TF-IDF: 0.01842
Top words in news article 5
 Word: commitment, TF-IDF: 0.0398
 Word: students, TF-IDF: 0.03487
 Word: college, TF-IDF: 0.03027
 Word: low-income, TF-IDF: 0.03027
 Word: tuition, TF-IDF: 0.0252
 Word: gaps, TF-IDF: 0.02412
 Word: towards, TF-IDF: 0.02031
 Word: affordable, TF-IDF: 0.01816
 Word: degree, TF-IDF: 0.01592
 Word: step, TF-IDF: 0.01514
```

In [344]:

```
doc_topics=[]
for i, blob in enumerate(bloblist):
    scores = {word: tfidf(word, blob, bloblist) for word in blob.words}
    sorted_words = sorted(scores.items(), key=lambda x: x[1], reverse=True)
    list=[]
    for word, score in sorted_words[:10]:
        list.append(word)
    doc_topics.append(list)
```

In [345]:

```
top_words=pd.DataFrame(columns=['Word 1', 'Word 2', 'Word 3', 'Word 4', 'Word 5', 'Word 6', 'Word 7', 'Word 8' ,'Word 9', 'Word 10'], data=doc_topics)
top_words
```

Out[345]:

	Word 1	Word 2	Word 3	Word 4	Word 5	Word 6	Word 7	Word 8	Word 9	Word 10
0	proviso	township	community	pp4h	organizations	health	obesity	academy	prevention	partners
1	towns	zarecor	smart	iowa	shrinking	says	amenities	factors	merrett	rural
2	wisconsin	koskinen	businesses	economist	using	network	draw	doing	he	labor
3	span	div	class	travel	title	must-see	attractions	bookinghunter.com	pier	li
4	commitment	students	college	low-income	tuition	gaps	towards	affordable	degree	step
5	rochelle	nippon	sharyo	ogle	manufacturer	–	plant	taxpayers	boom	hormel
6	span	div	class	travel	title	must-see	attractions	bookinghunter.com	pier	li
7	south	tax	assessed	homeowners	property	value	exemptions	homes	businesses	harvey
8	south	tax	assessed	homeowners	property	value	exemptions	homes	businesses	harvey
9	your	prenominal	field	p	you	clients	boodle	clams	whitethorn	non
10	employers	amazon	locate	foxconn	landing	dallas	austin	location	headquarters	hearing
11	emanuel	police	third	rahm	he	re-election	bid	seek	term	announcement
12	minors	shooting	killed	lab	shot	shootings	victims	gunshots	davis	arrests
13	emanuel	he	his	dyke	announcement	bowen	van	race	jason	mayoral
14	foxconn	amazon	employers	wisconsin	locate	burn	sorry	landing	gift	dallas
15	nursing	sepsis	dorsey	homes	care	infection	bedsores	jackson	medicare	pressure
16	emanuel	he	his	dyke	announcement	bowen	van	race	jason	mayoral
17	nursing	sepsis	dorsey	homes	care	infection	bedsores	jackson	medicare	pressure
18	's	na	n't	you	gon	know	i	'll	're	mean
19	inner-ring	niles	suburbs	suburb	dolton	park	central	attention	calumet	renn
20	joachim	employers	criminal	records	banking	allen	licensed	society	said	counselor
21	redfin	searching	houston	austin	houstonians	users	homes	looking	francisco	images
22	enrollment	students	edwardsville	universities	university	carbondale	freshman	first-time	fall	eastern
23	enrollment	students	edwardsville	universities	university	carbondale	freshman	first-time	fall	eastern
24	teaching	smith	profession	teacher	teachers	develop	recommendations	districts	obtain	helfer
25	income	personal	growth	nation	2007	hike	policy	failures	rating	tax
26	letters	friends	editor	contention	vacate	flexibility	threatening	trivial	competed	headcount
27	•	teaching	teacher	smith	profession	teachers	recommendations	shortage	preparation	districts
28	2010-2017	decrease	migration	due	net	change	0.2	population	ca	ny
29	becky	niu	principal	principals	students	adds	internship	grado	says	special
...	...	...	...	...	...	...	...	...	...	...
46	enlarge	florida	click	missouri	louis	kansas	chart	oregon	st	's
47	median	rent	cheaper	income	jose	droves		san	housing	leaving
48	median	rent	cheaper	income	jose	droves	san	housing	leaving	metro
49	teacher	teachers	test	basic	skills	education	latham	banchero	prospective	warned
50	median	rent	cheaper	income	jose	droves	san	housing	leaving	metro
51	gaming	sports	betting	gambling	rita	hearing	ryan	targets	fantasy	subcommittees
52	median	rent	cheaper	income	jose	droves	san	la	housing	leaving
53	rural	towns	communities	counties	loss	champaign-urbana	dayton	decline	incubator	small
54	amazon	bezos	company	cities	visits	hq2	virginia	executives	washington	location
55	median	rent	cheaper	income	jose	droves	san	la	housing	leaving

	Word 1	Word 2	Word 3	Word 4	Word 5	Word 6	Word 7	Word 8	Word 9	Word 10
56	amazon	bezos	company	cities	visits	hq2	virginia	executives	washington	location
57	amazon	bezos	company	cities	visits	hq2	virginia	executives	washington	location
58	note	depaul	overview	socioeconomic	1950	sprawl	proper	40,000	low-cost	moderate-cost
59	students	pritzker	teachers	rauner	centralia	teacher	grants	map	education	lane
60	rural	baker	health	report	summit	areas	hospitals	urban	2003	vohra
61	pritzker	education	rauner	students	funding	colleges	governor	teachers	centralia	teacher
62	oberweis	dairy	sen	company	company-owned	taxes	property	neither	owner	his
63	pekin	trampoline	wage	hour	sellers	institution	peoria	operational	stuff-a-bus	minimum
64	cps	schools	enrollment	students	kids	miami-dade	dropped	charter	count	jackson
65	fellows	our	conflict	edgar	2045	pension	thinking	process	infectious	hammered
66	luxury	conde	nast	traveler	realtors	newman	categories	sampah	hampton	realty
67	ewing	mourning	court	schools	she	book	closed	institution	school	ghosts
68	infectious	minnesota	problems	our	np	columnist	enemy—and	strengths—location	degrees—have	login
69	judges	employment	students	schools	prior	letters	student	followed	rather	union
70	pritzker	rauner	advertising	's	governor	democratic	campaign	race	–	ad
71	pritzker	rauner	advertising	democratic	governor	's	file	campaign	race	–
72	filipino	ethnically	suburb	'd	i	area	planning	downtown	sacramento	asians
73	ctu	charter	teachers	strike	obama	party	unions	sharkey	pritzker	schools
74	annie	attendance	frey	her	missouri	revolt	rauner	republican	madison	mike
75	pritzker	junk	afscme	rauner	math	'	curse	rated	rainy	fiscal

76 rows × 10 columns

Sentiment

In [42]:

```
import sklearn
from sklearn.model_selection import train_test_split
from sklearn.feature_extraction.text import CountVectorizer, HashingVectorizer, TfidfTransformer, TfidfVectorizer
from sklearn.naive_bayes import MultinomialNB
from sklearn.metrics import classification_report, confusion_matrix, accuracy_score
from sklearn.linear_model import LogisticRegression, SGDClassifier
from sklearn import metrics
```

Make every sentence from every article into its own record/observation.

In [350]:

```
class_sentences=[]
for n in range(len(articles)):
    sentences=nltk.sent_tokenize(articles.iloc[n].text_clean)
    for m in sentences:
        class_sentences.append(m)
```

Some of your "sentences" are not actually sentences. They are just numbers with a decimal at the end. Take them out.

In [351]:

```
class_sentences=[o for o in class_sentences if len(o) > 10]
len(class_sentences)
```

Out[351]:

```
3913
```

Then create a new column where you will apply a classification and do a split with 100 observations going into a dataset that you will hand label.

In [352]:

```
sentences=pd.DataFrame(columns=['Sentence'], data=class_sentences)
```

In [353]:

```
sentences['Class']=""
```

In [354]:

```
sentences.head()
```

Out[354]:

	Sentence	Class
0	Proviso Township, Illinois Located just nine miles west of the city of Chicago, Proviso Township is one of 29 townships in Cook County, Illinois.	
1	An urban community, the township spans just 30 square miles and has a population of roughly 150,000.	
2	The Township is diverse, with roughly one third Hispanic, one third black, and one third white.	
3	THE CHALLENGE Proviso Township experienced adult and childhood obesity rates much higher than national rates.	
4	There was no grocery store, and few goods and services were owned by black and Latino residents.	

Split into a training and test set. Unlike how we usually do it, the **train set will be much smaller than the test set** because we have to hand label the training set.

In [355]:

```
sentences_test, sentences_train = train_test_split(sentences, test_size=100, random_state=1)
print(sentences_train.shape)
print(sentences_test.shape)
```

```
(100, 2)
(3813, 2)
```

Then put it to a csv file that you can label by hand.

In [356]:

```
train_labels=sentences_train.to_csv('train_labels.csv')
```

Once labelled, reload the file.

In [357]:

In [357]:

```
#this was the encoding you needed to make it work
sentences_train=pd.read_csv('train_labels.csv', encoding = 'ISO-8859-1')
sentences_train=sentences_train.drop(['Unnamed: 0'], axis=1)
sentences_train.head()
```

Out[357]:

	Sentence	Class
0	Pritzker also said he would increase MAP dollars by 50 percent, providing grants for more than 70,000 additional students.	2
1	Of being homeless.	0
2	Avoidable transfers In September 2013, the Centers for Medicare & Medicaid Services said it was working to reduce avoidable transfers from nursing homes to hospitals.	0
3	Albany, GA : -10,964 population decrease due to migration 2010-2017; -3.9% net population change.	0
4	None of that mattered to the bureaucrats who saw only test scores and underutilized classrooms: numbers silent on the things that matter in educating our children.	0

So I have labelled Negative=0, Neutral=1, Positive=2.

Split the data as necessary to run the models.

In [359]:

```
X_train=sentences_train['Sentence']
X_test=sentences_test['Sentence']
y_train=sentences_train['Class']
y_test=sentences_test['Class']
```

Vectorize and transform the dataset:

In [360]:

```
vect = CountVectorizer()
X_train_dtm = vect.fit_transform(X_train)
X_test_dtm = vect.transform(X_test)
```

Training Models

Naive Bayes

In [361]:

```
# instantiate a Multinomial Naive Bayes model
nb = MultinomialNB()
nb.fit(X_train_dtm, y_train)
nb_pred_class = nb.predict(X_test_dtm)
```

Logistic Regression Model

In [362]:

```
logreg = LogisticRegression()
logreg.fit(X_train_dtm, y_train)
lr_pred_class = logreg.predict(X_test_dtm)
```

SVM Model

In [363]:

```
svm = SGDClassifier(max_iter=100, tol=None)
svm.fit(X_train_dtm, y_train)
svm_pred_class = svm.predict(X_test_dtm)
```

Then add these predictions to the original dataframe.

In [364]:

```
sentences_test['NB']=nb_pred_class.tolist()
sentences_test['LR']=lr_pred_class.tolist()
sentences_test['SVM']=svm_pred_class.tolist()
```

```
C:\Users\mjdun\Anaconda\lib\site-packages\ipykernel_launcher.py:1: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/pandas-docs/stable/indexing.html#indexing-view-versus-copy
  """Entry point for launching an IPython kernel.
C:\Users\mjdun\Anaconda\lib\site-packages\ipykernel_launcher.py:2: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/pandas-docs/stable/indexing.html#indexing-view-versus-copy

C:\Users\mjdun\Anaconda\lib\site-packages\ipykernel_launcher.py:3: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/pandas-docs/stable/indexing.html#indexing-view-versus-copy
  This is separate from the ipykernel package so we can avoid doing imports until
```

Sample from the predictions to seem which model seems best.

In [373]:

```
pd.set_option('display.max_colwidth', 200)
sentences_test.sample(25)
```

Out[373]:

	Sentence	Class	NB
1418	More people on the real estate website Redfin are looking to move out of Houston than to move in.		0
1322	As one study noted, 'older suburbs, particularly those built in the 1950s and 1960s, no longer attract new development or new residents' and 'exhibit the very symptoms of decline that US cities ex...		0
160	They are now being hit with even higher bills in an impoverished, long-struggling, largely African-American region where an outsized property tax burden already made it difficult to attract the re...		0
1501	Graduate enrollment fell 14.2 percent, contributing to an overall drop of 6.3 percent to 4,857.		0
2044	Yet, I moved into a different city than the one inhabited by the Death Row Ten, or even than Mario Flores, who was sentenced to death for a 1984 gang killing less than a mile from my first apartme...		0
3839	(Madison County Judge David Dugan, currently running for retention, jokes that there is exactly one more Republican judge in Madison County than there is in Cook County: One.)		0
1208	We have a statin Social Security.		1
598	Shana Dorsey said the staff never told her about seriousness of her father's pressure sore, which led to sepsis, a severe infection that can quickly turn deadly.		0

	Sentence	Class	NB
2959	Texas has five of the top 10 inbound cities.		0
3482	An inalterable loss.		0
3368	As of April, Miami-Dade's pre-K-12 enrollment measured 354,172.		0
2697	It's the state's extraordinarily high overall tax burden — anchored by its devastating property taxes — for far too little in return.		0
2579	Middle year programs don't have that restriction, but IB-only curriculum — which Chicago schools officials refer to as "wall-to-wall IB"— is the strong preference of the international organization...		0
2174	WorldStarHipHop.com has become a clearinghouse for amateur fight videos, with guys often shouting "Worldstar!" as they record themselves administering beatings or film someone else being pummeled;...		0
696	Molander said serious bedsores indicate "someone is being ignored for an extended time period." "When we see patients like that we file (patient neglect) complaints with adult protective services,...		0
1905	Illinois' median loan amounted to 78 percent of median home value before the housing bubble, and 79 percent after, while the median loan in the rest of the U.S. was 78 percent of median home value...		0
3145	Virginia is also seen as a more business-friendly state than Maryland.		2
1601	An Illinois Policy Institute study found the 2011 income tax hike severely hampered job creation and wage growth in Illinois, ultimately costing the state economy $55.8 billion in real GDP.		0
593	The case, pending in Cook County Circuit Court, is one of thousands across the country that allege enfeebled nursing home patients endured stressful, sometimes painful, hospital treatments for sep...		0
3654	"The fact is we have to win to get change," he told USA TODAY.		0
2796	By the time the Caterpillar name was a registered trademark in 1910, the Holt Caterpillar Co. had established a plant in East Peoria in a building that previously housed the Colean Manufacturing C...		2
2643	The lenders had one condition, however: To cut costs, the state had to abandon the deep cut for a cheaper, shallower channel.		0
1022	Having worked myself as a former sheriff with the superintendent Garry McCarthy former police superintendent Garry McCarthy.		0
3330	Everything except the politicians."		0
2093	AP Economics Writer Josh Boak contributed to this report.		0

Using our judgment, it seems that the Naive Bayes model performs the best, or at least the least poorly. Furthermore, using an ensemble does not seem to be helpful because the logistic regression does not perform well (and would only cause distortions) and the SVM does not classify thing in a way that would cancel out the tendencies of the Naive Bayes.

So **we will use the Naive Bayes model to classify for sentiment.**

Positive Sentiment (General)

In [426]:

```
pos_sentences=sentences_test[sentences_test['NB']==2]
pos_text=pos_sentences["Sentence"].tolist()
pos_text[0:5]
```

Out[426]:

```
['"When you hear a neighborhood is on fire, everyone wants to be part of that excitement," said the East Lincoln Park-based agent.',
 'Undergraduate enrollment also surpassed 20,000 for the first time, a nearly 7 percent increase over last year.',
 'Laur said urban institutions are flourishing because they offer more of what students want from a college experience.',
 'Virginia is also seen as a more business-friendly state than Maryland.',
 'Amazon has pledged $5 billion in investment and 50,000 jobs.']
```

Negative Sentiment (General)

In [431]:

```
neg_sentences=sentences_test[sentences_test['NB']==0]
neg_text=neg_sentences["Sentence"].tolist()
neg_text[0:5]
```

Out[431]:

```
['His career also included stops at the Illinois State Board of Education, where he worked as state school psychologist; the Knox-Warren Special Education District
he served as director Special Education; the LaGrange Area Department of Special Education, where he was executive director; and as a principal of a day treatment
in Galesburg for students with several emotional and problems.',
 'Even NPR initially headlined Burge's obituary describing him as "Accused of Torture." While it might make for a more cumbersome headline, a later correction on T
ot it a little closer to the truth: Today, CPD officer Jason Van Dyke stands trial for murder in the shooting of black teenager Lauquan McDonald, after the departm
ly succeeded in suppressing police camera footage of the shooting.',
 'It determined that would grow the economy by more than $5 billion a year, generate $30 million in state income and sales taxes and take 35,000 people off the pov
ls.',
 'Was it an all time.',
 'Bakersfield, CA : -7,314 population decrease due to migration 2010-2017; +6.4% net population change.']
```

PICKLE SO YOU CAN USE LATER

In [437]:

```
#the dataframes
pos_sentences.to_pickle('pos_sentences.pickle')
neg_sentences.to_pickle('neg_sentences.pickle')
```

In [438]:

```
#the lists
positive = open('pos.pkl', 'wb')
negative = open('neg.pkl', 'wb')
pickle.dump(pos_text, positive, -1)
pickle.dump(neg_text, negative, -1)
positive.close()
negative.close()
```

In [439]:

```
del pos_text
del neg_text
```

In [6]:

```
#then re-import the lists
pkl_pos = open('pos.pkl', 'rb')
pkl_neg = open('neg.pkl', 'rb')
pos_text = pickle.load(pkl_pos)
neg_text = pickle.load(pkl_neg)
pkl_pos.close()
pkl_neg.close()
#and the dataframes
pos_sentences = pd.read_pickle("pos_sentences.pickle")
neg_sentences = pd.read_pickle("neg_sentences.pickle")
```

For the positive and negative statements we will:

- clean up the statement by taking out stopwords, putting to lower, removing punctuation, and lemmatizing
- read each statement into a long string, one for positive and one for negative
- throw those long strings into a word cloud application

In [7]:

```
stop = set(stopwords.words('english'))
extra_stops=['chicago', 'illinois', 'city', 'said', 'say', 'year']
```

```
stop.update(extra_stops)
exclude = set(string.punctuation)
lemma = WordNetLemmatizer()
```

In [8]:

```
def clean(doc):
    stop_free = " ".join([i for i in doc.lower().split() if i not in stop])
    punc_free = ''.join(ch for ch in stop_free if ch not in exclude)
    normalized = " ".join(lemma.lemmatize(word) for word in punc_free.split())
    return normalized

pos_clean = [clean(doc).split() for doc in pos_text]
```

In [9]:

```
neg_clean = [clean(doc).split() for doc in neg_text]
```

First create one long string for the positive statements

In [10]:

```
pos_words=[]
for u in range(len(pos_clean)):
    words=pos_clean[u]
    for v in words:
        pos_words.append(v)
pos_string=" ".join(pos_words)
```

Then do the same for the negative statements

In [11]:

```
neg_words=[]
for w in range(len(neg_clean)):
    words=neg_clean[w]
    for x in words:
        neg_words.append(x)
neg_string=" ".join(neg_words)
```

Then save both to a .txt file that you can upload to a word cloud application.

In [12]:

```
pos_text_file = open("positive.txt", "w")
pos_text_file.write(pos_string)
pos_text_file.close()
#you have some characters in the text that prevents it from writing so you have to do it this way
with open('negative.txt', 'w', encoding='utf-8') as neg_text_file:
    neg_text_file.write(neg_string)
```

Creating a word cloud

Then go to: https://www.wordclouds.com/

Upload both the positive and negative text files. Some words common but not particularly helpful. Take them out.

For the positive: also, percent, one. And then set the word size to -10.

For the negative: illinois, county, people, population, state, year, one, new, said, it's, percent, like, also. And then set the word size to -75.

Targeted Sentiment (Stay in or Move to Illinois)

Businesses

We want to target our sentiment to articles involving only companies. In practice, the easiest way to do this to subset on articles that mention "Amazon" or "Caterpillar". Working with the data makes it apparent that these are the commonly discussed companies, and a Named Entity Recognition analysis would not add much insight.

In [3]:

```
#re-read those articles from earlier.
articles=pd.read_pickle('on_point.pickle')
```

In [7]:

```
in_terms=['amazon', 'caterpillar', 'employer']
in_stems=[porter.stem(c) for c in in_terms]
in_index=[]
out_index=[]
for a in range(0, len(articles)):
    words=articles.iloc[a].text
    stems=[porter.stem(t) for t in words.split()]
    #if any of the keywords are in the title (stems), put those stems in the list
    if any(s in stems for s in in_stems):
        in_index.append(a)
    #if none of the keywords are present insert a Null value
    else:
        out_index.append(a)
```

A sample of the "company" articles seems to be fairly on point.

In [8]:

```
companies=articles.iloc[in_index]
companies.title.head(10)
```

Out[8]:

```
1     For Shrinking Illinois Towns, A 'Smarter' Appr...
3     Dome Energy gives production update for July 2...
6     Gov. Rauner launches the Driving a Cleaner Ill...
10         Amazon's decision: Illinois or not Illinois?
14    Amazon&apos;s decision: Illinois or not Illinois?
20    Illinois employers more open to hiring people ...
30    House hunters: How high taxes hurt home invest...
38    Illinois Bicentennial: Despite HQ move, Peoria...
41    90 days until no paycheck: Time running out fo...
45    Illinois bicentennial: Despite headquarters mo...
Name: title, dtype: object
```

Then we repeat what we did on the full body of articles on this subset.

First, tokenize this new subset of articles by sentence (remember to take out the fake sentences).

In [10]:

```
company_sentences=[]
```

```
for nn in range(len(companies)):
    c_sentences=nltk.sent_tokenize(companies.iloc[nn].text_clean)
    for mm in c_sentences:
        company_sentences.append(mm)
company_sentences=[oo for oo in company_sentences if len(oo) > 10]
```

In [11]:

```
company_sentences=pd.DataFrame(columns=['Sentence'], data=company_sentences)
company_sentences['Class']=""
company_sentences.head()
```

Out[11]:

	Sentence	Class
0	By Mary Hansen • 37 minutes ago Photo by Tom Z...	
1	By 2025, rural counties with populations of le...	
2	That's according to projections from the Illin...	
3	Christopher Merrett, director of the Illinois ...	
4	"There's pull factors, sort of bright lights, ...	

Then we split into train and test, label the training data, train a Naive Bayes model, and then use that model to predict the sentiment of individual sentences.

We are retraining the model in the hope that we might be more accurate in this targeted sentiment analysis.

In [37]:

```
c_sentences_test, c_sentences_train = train_test_split(company_sentences, test_size=100, random_state=12)
print(c_sentences_train.shape)
print(c_sentences_test.shape)
company_train_labels=c_sentences_train.to_csv('company_train_labels.csv', encoding = 'utf-8')
```

```
(100, 2)
(725, 2)
```

So I have labelled Negative=0, Neutral=1, Positive=2.

Then after labelling, reload.

In [38]:

```
company_train=pd.read_csv('company_train_labels.csv', encoding = 'utf-8')
company_train=company_train.drop(['Unnamed: 0'], axis=1)
company_train.head()
```

Out[38]:

	Sentence	Class
0	In Illinois, the winner's curse is real.	2
1	California benefited from high housing price a...	2
2	He earned associate, bachelor's and master's d...	2
3	While Amazon appears to be narrowing its list ...	0
4	The potential prize for whatever city emerges ...	2

Prep the data to put into your model.

In [39]:

```
company_X_train=company_train['Sentence']
company_X_test=c_sentences_test['Sentence']
company_y_train=company_train['Class']
company_y_test=c_sentences_test['Class']
print(company_X_train.shape, company_y_train.shape)
print(company_X_test.shape, company_y_test.shape)
```

```
(100,) (100,)
(725,) (725,)
```

Train your Naive Bayes model.

In [44]:

```
company_vect = CountVectorizer()
company_X_train_dtm = company_vect.fit_transform(company_X_train)
company_X_test_dtm = company_vect.transform(company_X_test)
```

In [46]:

```
company = MultinomialNB()
company.fit(company_X_train_dtm, company_y_train)
company_pred_class = company.predict(company_X_test_dtm)
```

And then add those predictions back to your test dataframe.

In [49]:

```
c_sentences_test['NB']=company_pred_class.tolist()
c_sentences_test.head()
```

```
C:\Users\mjdun\Anaconda\lib\site-packages\ipykernel_launcher.py:1: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/pandas-docs/stable/indexing.html#indexing-view-versus-copy
  """Entry point for launching an IPython kernel.
```

Out[49]:

	Sentence	Class	NB
732	The flurry of recent discussions are part of t...		0
275	Why homeownership matters While not all econom...		0
172	"Not only have we kicked down those doors, but...		0
706	It could also help with recruiting employees, ...		2
22	Back in Illinois, Merrett, who works with town...		0

And then collect the positive and negative sentences into two separate lists.

In [50]:

```
pos_company_sentences=c_sentences_test[c_sentences_test['NB']==2]
pos_company_text=pos_company_sentences["Sentence"].tolist()
neg_company_sentences=c_sentences_test[c_sentences_test['NB']==0]
neg_company_text=neg_company_sentences["Sentence"].tolist()
```

In [51]:

```
#pickle them just in case
positive_company = open('pos_company.pkl', 'wb')
negative_company = open('neg_company.pkl', 'wb')
pickle.dump(pos_company_text, positive_company, -1)
pickle.dump(neg_company_text, negative_company, -1)
positive_company.close()
negative_company.close()
```

For the positive and negative statements we will:

- clean up the statement by taking out stopwords, putting to lower, removing punctuation, and lemmatizing
- read each statement into a long string, one for positive and one for negative
- throw those long strings into a word cloud application

In [55]:

```
stop = set(stopwords.words('english'))
extra_stops=['chicago', 'illinois', 'city', 'said', 'say', 'year']
stop.update(extra_stops)
exclude = set(string.punctuation)
lemma = WordNetLemmatizer()
```

In [56]:

```
def clean(doc):
    stop_free = " ".join([i for i in doc.lower().split() if i not in stop])
    punc_free = ''.join(ch for ch in stop_free if ch not in exclude)
    normalized = " ".join(lemma.lemmatize(word) for word in punc_free.split())
    return normalized

pos_company_clean = [clean(doc).split() for doc in pos_company_text]
neg_company_clean = [clean(doc).split() for doc in neg_company_text]
```

Create one long string for both positive and negative

In [57]:

```
pos_company_words=[]
for uu in range(len(pos_company_clean)):
    pos_words=pos_company_clean[uu]
    for vv in pos_words:
        pos_company_words.append(vv)
pos_company_string=" ".join(pos_company_words)

neg_company_words=[]
for ww in range(len(neg_company_clean)):
    neg_words=neg_company_clean[ww]
    for xx in neg_words:
        neg_company_words.append(xx)
neg_company_string=" ".join(neg_company_words)
```

Then save both to a text file

In [58]:

```
pos_company_text_file = open("positive_company.txt", "w")
pos_company_text_file.write(pos_company_string)
pos_company_text_file.close()
#you have some characters in the text that prevents it from writing so you have to do it this way
with open('negative_company.txt', 'w', encoding='utf-8') as neg_company_text_file:
    neg_company_text_file.write(neg_company_string)
```

Creating a word cloud

Then go to: https://www.wordclouds.com/

Upload both the positive and negative text files.

For the negative: Set the word size to -25.

Residents

A sample of the "resident" articles seems to be fairly on point.

In [9]:

```
residents=articles.iloc[out_index]
residents.title.head(10)
```

Out[9]:

```
0         Spotlight Stories: Proviso Township, Illinois
2       Official compares Wisconsin economy to Illinois
4     “Illinois Commitment” a Critical Step Towards ...
5     Taxpayer-subsidized manufacturer shutters Illi...
7     &apos;A perfect tax storm in the south suburbs...
8     &apos;A perfect tax storm in the south suburbs...
9     'What makes Chicago such an excellent region f...
11    Rahm Emanuel Surprises Chicago By Dropping Out...
12    Alarming Number of Chicago Minors Shot or Kill...
13                                Chicago, After Rahm
Name: title, dtype: object
```

As before, make a dataframe of all the sentences in these articles.

In [14]:

```
resident_sentences=[]
for pp in range(len(residents)):
    r_sentences=nltk.sent_tokenize(residents.iloc[pp].text_clean)
    for qq in r_sentences:
        resident_sentences.append(qq)
resident_sentences=[rr for rr in resident_sentences if len(rr) > 10]
```

In [15]:

```
resident_sentences=pd.DataFrame(columns=['Sentence'], data=resident_sentences)
resident_sentences['Class']=""
resident_sentences.head()
```

Out[15]:

	Sentence	Class
0	Proviso Township, Illinois Located just nine...	
1	An urban community, the township spans just 30...	
2	The Township is diverse, with roughly one thir...	
3	THE CHALLENGE Proviso Township experienced adu...	
4	There was no grocery store, and few goods and ...	

Then we split into train and test, label the training data, train a Naive Bayes model, and then use that model to predict the sentiment of individual sentences.

We are retraining the model in the hope that we might be more accurate in this targeted sentiment analysis.

In [33]:

```
r_sentences_test, r_sentences_train = train_test_split(resident_sentences, test_size=100, random_state=12)
print(r_sentences_train.shape)
print(r_sentences_test.shape)
resident_train_labels=r_sentences_train.to_csv('resident_train_labels.csv', encoding='utf-8')
```

```
(100, 2)
(2988, 2)
```

So I have labelled Negative=0, Neutral=1, Positive=2.

Then after labelling, reload.

In [35]:

```
resident_train=pd.read_csv('resident_train_labels.csv', encoding = 'utf-8')
resident_train=resident_train.drop(['Unnamed: 0'], axis=1)
resident_train.head()
```

Out[35]:

	Sentence	Class
0	Well Manuel has always been.	1
1	And while he never heeded calls to resign, his...	0
2	Cause and effect The homeowner tax breaks took...	0
3	They may be from preschool programs through hi...	2
4	Court records show that Willie Jackson's hospi...	0

Prep the data to put into your model. Train the model. And then add those predictions back to your test dataframe.

In [60]:

```
resident_X_train=resident_train['Sentence']
resident_X_test=r_sentences_test['Sentence']
resident_y_train=resident_train['Class']
resident_y_test=r_sentences_test['Class']

resident_vect = CountVectorizer()
resident_X_train_dtm = resident_vect.fit_transform(resident_X_train)
resident_X_test_dtm = resident_vect.transform(resident_X_test)

resident = MultinomialNB()
resident.fit(resident_X_train_dtm, resident_y_train)
resident_pred_class = resident.predict(resident_X_test_dtm)
```

And then add those predictions back to your test dataframe.

In [61]:

```
r_sentences_test['NB']=resident_pred_class.tolist()
r_sentences_test.head()
```

```
C:\Users\mjdun\Anaconda\lib\site-packages\ipykernel_launcher.py:1: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/pandas-docs/stable/indexing.html#indexing-view-versus-copy
  """Entry point for launching an IPython kernel.
```

Out[61]:

	Sentence	Class	NB
2532	Both candidates said they support the program,...		1
2830	In another ad, Pritzker blames Rauner for the ...		0
812	And at the end of the day Rahm Emanuel is a co...		0
457	This candidate will need to have the infrastru...		0
2627	We are so glum about our fiscal, tax, politica...		0

And then collect the positive and negative sentences into two separate lists.

In [62]:

```
pos_resident_sentences=r_sentences_test[r_sentences_test['NB']==2]
pos_resident_text=pos_resident_sentences["Sentence"].tolist()
neg_resident_sentences=r_sentences_test[r_sentences_test['NB']==0]
neg_resident_text=neg_resident_sentences["Sentence"].tolist()
```

In [63]:

```
#pickle them just in case
positive_resident = open('pos_resident.pkl', 'wb')
negative_resident = open('neg_resident.pkl', 'wb')
pickle.dump(pos_resident_text, positive_resident, -1)
pickle.dump(neg_resident_text, negative_resident, -1)
positive_resident.close()
negative_resident.close()
```

Then clean

In [64]:

```
pos_resident_clean = [clean(doc).split() for doc in pos_resident_text]
```

```
pos_resident_clean = [clean(doc).split() for doc in pos_resident_text]
neg_resident_clean = [clean(doc).split() for doc in neg_resident_text]
```

Create one long string for both positive and negative

In [65]:

```
pos_resident_words=[]
for uuu in range(len(pos_resident_clean)):
    pos_words=pos_resident_clean[uuu]
    for vvv in pos_words:
        pos_resident_words.append(vvv)
pos_resident_string=" ".join(pos_resident_words)

neg_resident_words=[]
for www in range(len(neg_resident_clean)):
    neg_words=neg_resident_clean[www]
    for xxx in neg_words:
        neg_resident_words.append(xxx)
neg_resident_string=" ".join(neg_resident_words)
```

Then save both to a text file.

In [66]:

```
pos_resident_text_file = open("positive_resident.txt", "w")
pos_resident_text_file.write(pos_resident_string)
pos_resident_text_file.close()
#you have some characters in the text that prevents it from writing so you have to do it this way
with open('negative_resident.txt', 'w', encoding='utf-8') as neg_resident_text_file:
    neg_resident_text_file.write(neg_resident_string)
```

Creating a word cloud

Then go to: https://www.wordclouds.com/

Upload both the positive and negative text files. Some words common but not particularly helpful. Take them out.

For the positive: Set the word size to +75.

For the negative: state, population, one, year, also, said, time, many, illinois, people, it's, say, even. And then set the word size to -70.