

Praveen Nair

LIGN 6

20 March 2019

Does Winning Correlate with More Positive Sentiment in Sportswriting?

It's an open secret that sports journalism often has little to do with the play on the field. Narratives, biases, and selective focus make sportswriting just as unpredictable and chaotic as the sports themselves. But, as the saying goes, winning cures all. But does winning really lead to a more positive spin in media portrayals, or do the higher expectations of elite teams lead to more negative articles? For this project, I decided to use sentiment analysis to study the association between a sports team's winning percentage and the positivity of articles about that team in sportswriting.

In order to accomplish this, I used a few Python packages as well as some Unix text manipulation. I was using Python 3.7.2 run on a Jupyter Notebook (which I personally find a bit easier for immediately viewing text and table output than running a script in an IDE). Chiefly, I used the built-in Element Tree module to read in and parse the provided corpus into an iterable object that I could more easily understand, NLTK (Natural Language Toolkit) to run the actual sentiment analysis, and Pandas to store and manipulate the many tables I used in the project. Within NLTK, I used the VADER sentiment analyzer to derive compound scores for the sentiment of the sports articles. In a more peripheral role, I used the Python packages numpy (for math tasks), os (to iterate through the corpus files), matplotlib and seaborn (for basic visualization), and json (to write to and read from a JSON file). All packages were installed using Python's pip package manager, and the Jupyter Notebook software came packaged with an installation of Anaconda.

The corpus I used was a corpus from the *New York Times* spanning the dates July 1, 1994 and June 30, 2002. It has an approximate size of 5.6 GB, so it contains approximately 5,600,000,000 characters. The data is in an XML format, with each article containing its headline, dateline, and text. This corpus includes *everything* from the *Times*, so I cut it down to a smaller database of 190 MB of NBA and NFL articles. One benefit of using the *New York Times* is that it has relatively few articles that are about routine games and scores, which cuts down on the number of articles that are about two teams at once. In addition to this data, I used data on winning percentage from Sports Reference (specifically basketball-reference.com and pro-football-reference.com), which tracks a huge variety of sports data from a multitude of sports. I extracted a table for each football and basketball, and then joined them (using Pandas) with my dataset of sentiment scores. Finally, I created my own lists of teams and possible names using my own research online (most complications arose from relocations and renaming.) As I mentioned before, I used NLTK's included language model, VADER (Valence Aware Dictionary and sEntiment Reasoner). VADER is trained on social media data, which is a *very* different context than the more formal context of the *New York Times*. However, although a disparity is sure to exist, I trust that VADER is more accurate than if the roles of social media and newspaper were flipped, and it was easily available as part of NLTK.

What follows is the Python I used for this analysis. Prior to running the Python, however, I did some work in Unix to clean the data up a bit. I used sed (stream editor) to replace certain tokens that the Element Tree parser couldn't understand, I added an html tag at the top and bottom of each file, and I also converted the files to .txt and formatted them in UTF-8. Note: the Jupyter Notebook also serves as my presentation, so you can ignore some of those elements.

How does the positivity of sportswriting about a sports team correlate with their success?

A slightly disappointing look at the turn-of-the-century New York Times sports section.

By Praveen Nair

In [1]:

```
import nltk
from nltk.sentiment.vader import SentimentIntensityAnalyzer
import xml.etree.ElementTree as ET
import os
import numpy as np
import matplotlib
import json
import pandas as pd
import seaborn as sns
from IPython.display import Image
```

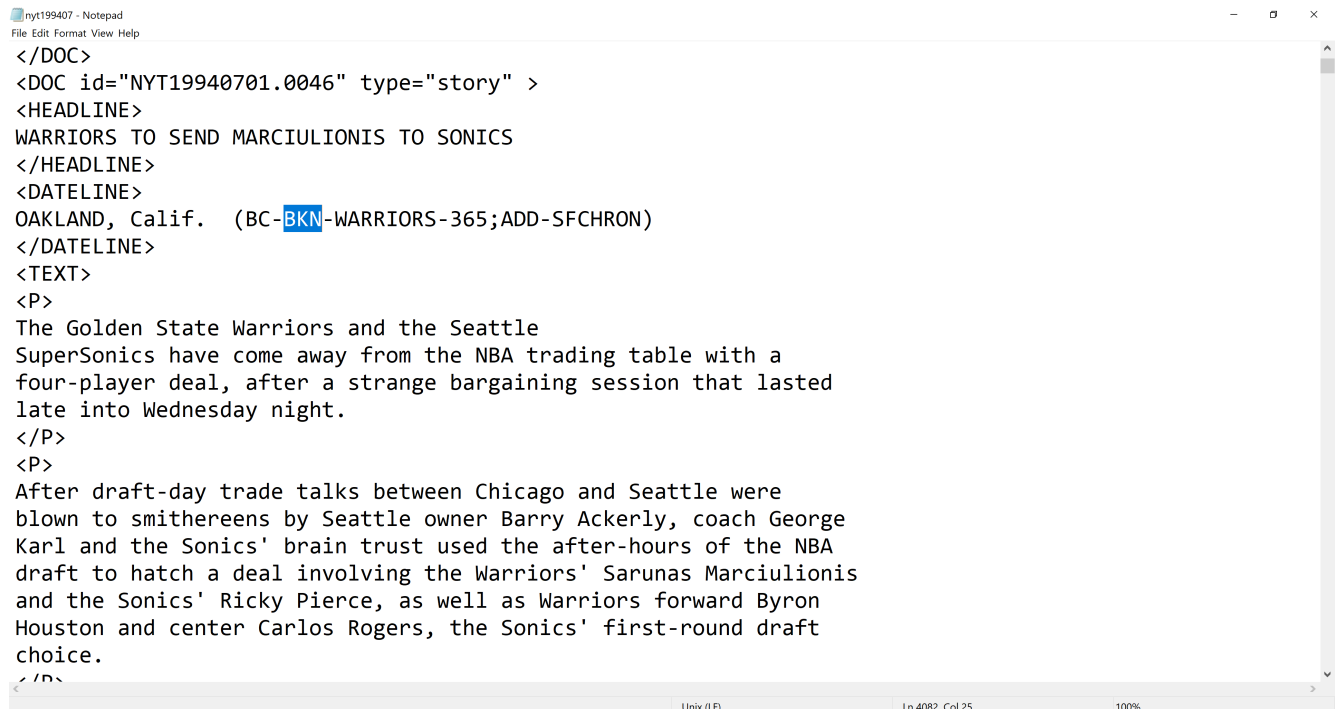
The Corpus

The entire New York Times from July 1, 1994 to June 30, 2006. It's around 6 GB, so around a billion words.

In [2]:

```
Image(filename = 'corpus_sample.png')
```

Out[2]:



```
ny1199407 - Notepad
File Edit Format View Help
</DOC>
<DOC id="NYT19940701.0046" type="story" >
<HEADLINE>
WARRIORS TO SEND MARCIULIONIS TO SONICS
</HEADLINE>
<DATELINE>
OAKLAND, Calif. (BC-BKN-WARRIORS-365;ADD-SFCHRON)
</DATELINE>
<TEXT>
<P>
The Golden State Warriors and the Seattle
SuperSonics have come away from the NBA trading table with a
four-player deal, after a strange bargaining session that lasted
late into Wednesday night.
</P>
<P>
After draft-day trade talks between Chicago and Seattle were
blown to smithereens by Seattle owner Barry Ackerly, coach George
Karl and the Sonics' brain trust used the after-hours of the NBA
draft to hatch a deal involving the Warriors' Sarunas Marciulionis
and the Sonics' Ricky Pierce, as well as Warriors forward Byron
Houston and center Carlos Rogers, the Sonics' first-round draft
choice.
</P>
```

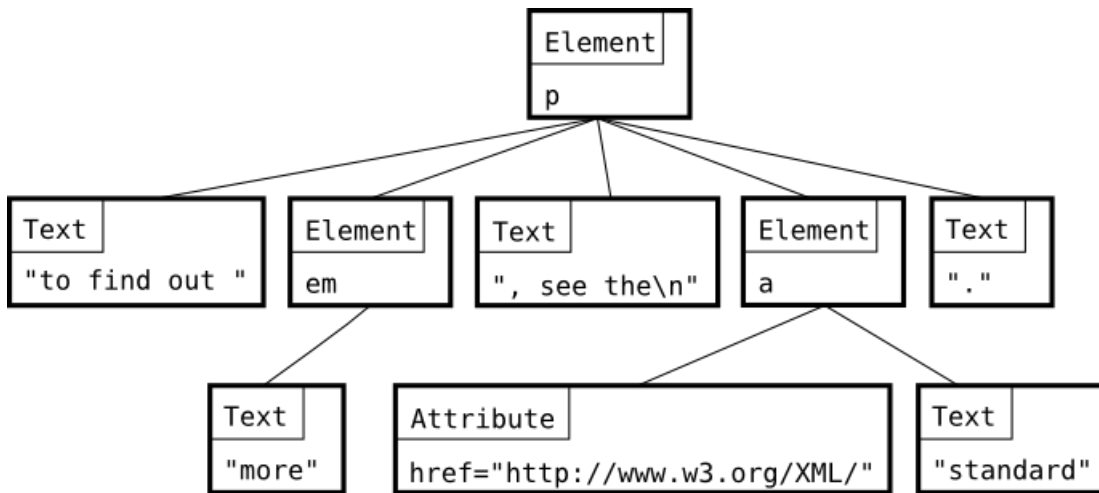
Cool. But how do I access all this information?

Python has a built-in library for interacting with xml, and modeling it very easily as a tree of elements.

In [3]:

```
Image(filename = 'element_tree.png')
```

Out [3]:



Step 1: Get only NBA and NFL articles.

We need to delete articles about such topics as:

- The OJ trial
- The impeachment of Bill Clinton
- and OJ Simpson.

But how? It turns out the dateline of each article gives us some information about what kind of news it is. 'BKN' refers to NBA basketball, and 'FBN' refers to NFL football.

In [4]:

```
Image(filename = 'corpus_sample.png')
```

Out [4]:

```
ny1199407 - Notepad
File Edit Format View Help
</DOC>
<DOC id="NYT19940701.0046" type="story" >
<HEADLINE>
WARRIORS TO SEND MARCIULIONIS TO SONICS
</HEADLINE>
<DATELINE>
OAKLAND, Calif. (BC-BKN-WARRIORS-365;ADD-SFCHRON)
</DATELINE>
<TEXT>
<P>
The Golden State Warriors and the Seattle
SuperSonics have come away from the NBA trading table with a
four-player deal, after a strange bargaining session that lasted
late into Wednesday night.
</P>
<P>
After draft-day trade talks between Chicago and Seattle were
blown to smithereens by Seattle owner Barry Ackerly, coach George
Karl and the Sonics' brain trust used the after-hours of the NBA
draft to hatch a deal involving the Warriors' Sarunas Marciulionis
and the Sonics' Ricky Pierce, as well as Warriors forward Byron
Houston and center Carlos Rogers, the Sonics' first-round draft
choice.
</P>
```

In [5]:

```
# Gets every NBA and NFL article in the tree and returns it
def get_sports_articles(element_tree):
    count = 0
    root = element_tree.getroot()
    # Iterates through every document
    for doc in root[:]:
        dateline = doc.find('DATELINE').text
        # Finds if it's a football/basketball article
        if 'BKN' in dateline or 'FBN' in dateline:
            count += 1
        # If it isn't, we remove it from the tree -- not the file!
        else:
            root.remove(doc)
    return element_tree
```

In [6]:

```
run_parser = False
if run_parser:
    # Iterate through corpus
    for file in os.listdir():
        if file[-3:] == '.txt':
            print(file)
            tree = ET.parse(file)
            # Write trimmed tree to a new file
            get_sports_articles(tree).write('sports_' + file)
```

Step 2: Get the sentiment of every NBA/NFL article.

How are we going to do that? Well, NLTK includes a sentiment analysis tool called VADER.

VADER, HUGE disclaimer, is trained on social media data.

In [7]:

```
analyzer = SentimentIntensityAnalyzer()
```

In [8]:

```
analyzer.polarity_scores('Boy, that last Transformers film stunk. They should fire the director in to the sun.')
```

Out[8]:

```
{'neg': 0.294, 'neu': 0.706, 'pos': 0.0, 'compound': -0.6124}
```

In [9]:

```
analyzer.polarity_scores('Transformers: Age of Extinction is a historic film. It is simply fantastic. Better than The Godfather.')
```

Out[9]:

```
{'neg': 0.0, 'neu': 0.667, 'pos': 0.333, 'compound': 0.7579}
```

Now we iterate through the directory and get all the headline, dateline, and sentiment of the text.

In [10]:

```
# Takes element tree of text and turns it into a list of dictionaries
# Each dictionary contains the headline, dateline, and sentiment of that article's text
```

```
# Parameter = tree
def tree_to_dicts(element_tree):
    root = element_tree.getroot()
    articles = []
    for doc in root:
        this_doc = {}
        doc_elems = [i for i in doc]
        for i in doc_elems:
            if i.tag == 'HEADLINE':
                this_doc['HEADLINE'] = i.text
            elif i.tag == 'DATELINE':
                this_doc['DATELINE'] = i.text
            elif i.tag == 'TEXT':
                text_str = ''
                for j in i:
                    text_str += j.text
                this_doc['SENTIMENT'] = analyzer.polarity_scores(text_str)['compound']
        articles.append(this_doc)
    return articles
```

In [11]:

```
if run_parser:
    sports_sentiment = []
    # Iterates through trimmed sports files,
    # and runs the above function on each
    for file in os.listdir():
        if file[:7] == 'sports_':
            tree = ET.parse(file)
            sports_sentiment += tree_to_dicts(tree)
            print(file)
```

Step 3: Turn it into a JSON so I don't have to redo steps 1 and 2 every time.

In [12]:

```
# with open('sports_sentiments', 'w') as file_out:
#     json.dump(sports_sentiment, file_out)
```

Step 4: Read back from JSON.

In [13]:

```
with open('sports_sentiments', 'r') as file_in:
    sports_sentiment = json.load(file_in)
```

Step 5: Make a list of every team in the NBA and NFL. Some teams have alternate names, so we want to include those.

In [14]:

```
nba_teams = [['76ers', 'Sixers', 'Philadelphia', 'Philly'],
             ['Heat', 'Miami'], ['Knicks', 'New York'],
             ['Magic', 'Orlando'], ['Celtics', 'Boston'],
             ['Nets', 'Jersey'], ['Wizards', 'Bullets', 'Washington', 'D.C.'],
             ['Bucks', 'Milwaukee'], ['Raptors', 'Toronto'],
             ['Hornets', 'New Orleans', 'Charlotte'],
             ['Pistons', 'Detroit'], ['Cavaliers', 'Cavs', 'Cleveland'],
             ['Hawks', 'Atlanta'], ['Bulls', 'Chicago'],
             ['Spurs', 'San Antonio'], ['Jazz', 'Utah'],
             ['Mavericks', 'Mavs', 'Dallas'],
             ['Timberwolves', 'Wolves', 'Minnesota'],
             ['Rockets', 'Houston'], ['Nuggets', 'Denver'],
             ['Grizzlies', 'Vancouver', 'Memphis'],
             ['Lakers'], ['Kings', 'Sacramento'],
             ['Suns', 'Phoenix'], ['Blazers', 'Portland'],
             ['Sonic', 'Seattle'], ['Clippers']]
```

```

    ['Warriors', 'Golden State', 'Oakland'],
    ['Pacers', 'Indiana']
]
len(nba_teams)

```

Out[14]:

29

In [15]:

```

nfl_teams = [['Seahawks', 'Seattle'],
             ['49ers', 'San Francisco', 'Niners'],
             ['Raiders', 'Oakland'], ['Rams', 'St. Louis'],
             ['Chargers', 'San Diego'], ['Cardinals', 'Arizona', 'Phoenix'],
             ['Broncos', 'Denver'], ['Chiefs', 'Kansas City', 'K.C.'],
             ['Cowboys', 'Dallas'], ['Vikings', 'Minnesota'],
             ['Saints', 'New Orleans'], ['Packers', 'Green Bay'],
             ['Bears', 'Chicago'], ['Colts', 'Indianapolis'],
             ['Titans', 'Tennessee', 'Oilers', 'Houston'],
             ['Bengals', 'Cincinnati'], ['Lions', 'Detroit'],
             ['Browns', 'Cleveland'], ['Panthers', 'Carolina'],
             ['Bills', 'Buffalo'], ['Steelers', 'Pittsburgh'],
             ['Falcons', 'Atlanta'], ['Jaguars', 'Jags', 'Jacksonville'],
             ['Buccaneers', 'Bucs', 'Tampa'], ['Dolphins', 'Miami'],
             ['Patriots', 'New England'], ['Jets'], ['Giants'],
             ['Eagles', 'Philadelphia', 'Philly'], ['Ravens', 'Baltimore'],
             ['Redskins', 'Washington']
]
len(nfl_teams)

```

Out[15]:

31

Step 6: Now we're going to iterate through those team names, find articles matching them, and find the average sentiment of those articles.

In [16]:

```

nba_team_sentiment = []
for team in nba_teams:
    team_articles = []
    # For each article, we find if the team is mentioned
    # in the headline or dateline
    for article in sports_sentiment:
        team_in_article = False
        has_headline = len(article) == 3
        if has_headline:
            lines = article['HEADLINE'] + article['DATELINE']
        else:
            lines = article['DATELINE']
        for name in team:
            if name.upper() in lines:
                team_in_article = True
        if team_in_article:
            team_articles.append(article['SENTIMENT'])
    nba_team_sentiment.append((team[0], np.mean(team_articles), len(team_articles)))

```

In [17]:

```

nfl_team_sentiment = []
for team in nfl_teams:
    team_articles = []
    # For each article, we find if the team is mentioned
    # in the headline or dateline
    for article in sports_sentiment:
        team_in_article = False
        has_headline = len(article) == 3
        if has_headline:
            lines = article['HEADLINE'] + article['DATELINE']

```

```

else:
    lines = article['DATELINE']
for name in team:
    if name.upper() in lines:
        team_in_article = True
if team_in_article:
    team_articles.append(article['SENTIMENT'])
nfl_team_sentiment.append((team[0], np.mean(team_articles), len(team_articles)))

```

In [18]:

```

# Make the outputs of the above functions into DataFrames
nba = pd.DataFrame(nba_team_sentiment, columns = ['Team', 'Sentiment Score', 'n'])
nfl = pd.DataFrame(nfl_team_sentiment, columns = ['Team', 'Sentiment Score', 'n'])

```

Step 7: Now let's introduce winning percentage. This data was extracted from Basketball and Football Reference.

In [19]:

```

# Read in data as DataFrame
nba_win_pct = pd.read_csv('nba_win_pct.csv')
nfl_win_pct = pd.read_csv('nfl_win_pct.csv')

nba = nba.set_index('Team')
nba_win_pct = nba_win_pct.set_index('Team')
nfl = nfl.set_index('Team')
nfl_win_pct = nfl_win_pct.set_index('Team')

# Join the win percentage tables with our existing ones
nba = nba.join(nba_win_pct)

nfl = nfl.join(nfl_win_pct)

```

In [20]:

nba

Out[20]:

Team	Sentiment Score	n	win_pct
76ers	0.499861	571	0.434
Heat	0.543800	1181	0.579
Knicks	0.532696	2859	0.572
Magic	0.649549	651	0.583
Celtics	0.697678	1811	0.413
Nets	0.509169	1395	0.407
Wizards	0.595312	338	0.399
Bucks	0.411393	166	0.466
Raptors	0.533801	226	0.413
Hornets	0.656013	686	0.579
Pistons	0.557420	171	0.510
Cavaliers	0.510956	290	0.468
Hawks	0.632464	1949	0.498
Bulls	0.629662	971	0.506
Spurs	0.646225	696	0.646
Jazz	0.633806	387	0.689
Mavericks	0.549045	1567	0.441
Timberwolves	0.504612	182	0.487

Team	Sentiment Score	n	win_pct
Rockets	0.644631	1160	0.530
Nuggets	0.644631	425	0.359
Grizzlies	0.590410	171	0.229
Lakers	0.681265	2637	0.689
Kings	0.543329	774	0.522
Suns	0.645970	1049	0.582
Blazers	0.498918	401	0.603
Sonics	0.625120	1549	0.638
Clippers	0.537423	929	0.309
Warriors	0.516587	595	0.303
Pacers	0.633136	667	0.598

In [21]:

```
nfl
```

Out [21]:

Team	Sentiment Score	n	win_pct
Seahawks	0.646313	1481	0.477
49ers	0.633198	1287	0.648
Raiders	0.570546	1001	0.516
Rams	0.584291	398	0.492
Chargers	0.764495	447	0.398
Cardinals	0.601700	831	0.375
Broncos	0.709480	599	0.617
Chiefs	0.595625	418	0.570
Cowboys	0.575306	3863	0.531
Vikings	0.568066	290	0.602
Saints	0.725607	492	0.383
Packers	0.757065	533	0.672
Bears	0.564256	630	0.445
Colts	0.633979	534	0.477
Titans	0.622307	856	0.516
Bengals	0.445499	157	0.328
Lions	0.539910	170	0.445
Browns	0.616966	318	0.350
Panthers	0.688494	127	0.411
Bills	0.624590	265	0.508
Steelers	0.729824	438	0.617
Falcons	0.646237	1536	0.438
Jaguars	0.587309	228	0.554
Buccaneers	0.688262	413	0.523
Dolphins	0.628907	1181	0.602
Patriots	0.739133	2281	0.547
Jets	0.555291	1796	0.453
Giants	0.583697	1580	0.504
Eagles	0.571196	616	0.496
Ravens	0.644011	232	0.484
Redskins	0.594705	400	0.457

In [22]:

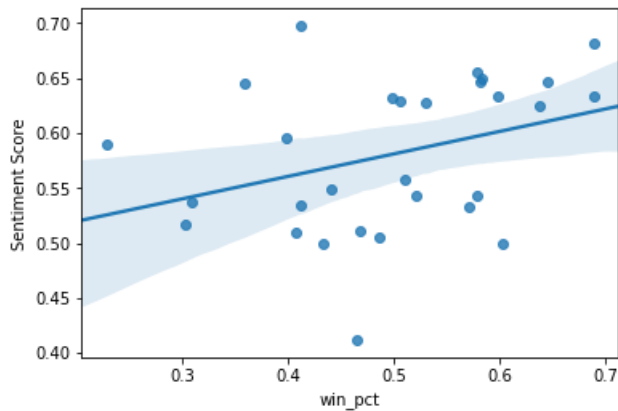
```
sns.regplot(nba['win_pct'], nba['Sentiment Score'])
```

C:\Users\prave\Anaconda3\lib\site-packages\scipy\stats\stats.py:1713: FutureWarning: Using a non-tuple sequence for multidimensional indexing is deprecated; use `arr[tuple(seq)]` instead of `arr[seq]`. In the future this will be interpreted as an array index, `arr[np.array(seq)]`, which will result either in an error or a different result.

```
return np.add.reduce(sorted[indexer] * weights, axis=axis) / sumval
```

Out[22]:

<matplotlib.axes._subplots.AxesSubplot at 0x18a7549a080>

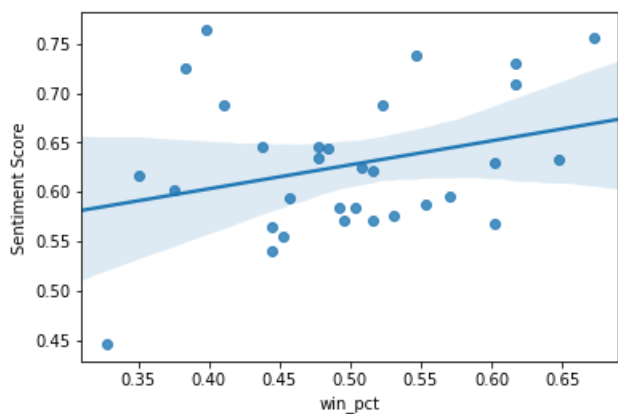


In [23]:

```
sns.regplot(nfl['win_pct'], nfl['Sentiment Score'])
```

Out[23]:

<matplotlib.axes._subplots.AxesSubplot at 0x18a757fac88>



In [24]:

```
r_nba = np.corrcoef(nba['win_pct'], nba['Sentiment Score'])[0][1]  
r_nba
```

Out[24]:

0.34246493082060614

In [25]:

```
r_nfl = np.corrcoef(nfl['win_pct'], nfl['Sentiment Score'])[0][1]  
r_nfl
```

Out[25]:

0.294423915329532

In [26]:

```
Image(filename = 'alf.png')
```

Out[26]:

Me: i'm going to analyze this really complicated relationship despite there being a million confounding factors i can't account for

r: 0.34

Me:



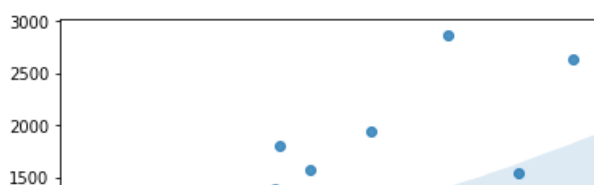
Well, while we're here, we might as well see another relationship: How does win percentage correlate with mentions in the corpus?

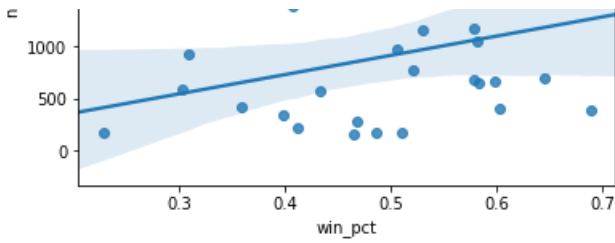
The teams at the top right are the Knicks and Lakers, predictably. At the bottom left are the Grizzlies.

In [27]:

```
sns.regplot(nba['win_pct'], nba['n'])  
print('r:', np.corrcoef(nba['win_pct'], nba['n'])[0][1])
```

r: 0.2979703551139872





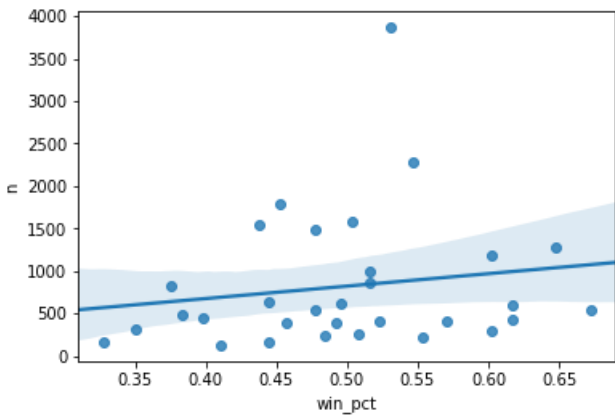
This graph is called "Why Nobody Likes the Dallas Cowboys."

Bottom left teams, by the way? Cleveland Browns and Cincinnati Bengals. Sorry, Ohio.

In [28]:

```
sns.regplot(nfl['win_pct'], nfl['n'])
print('r:', np.corrcoef(nfl['win_pct'], nfl['n'])[0][1])
```

r: 0.16237368454380122

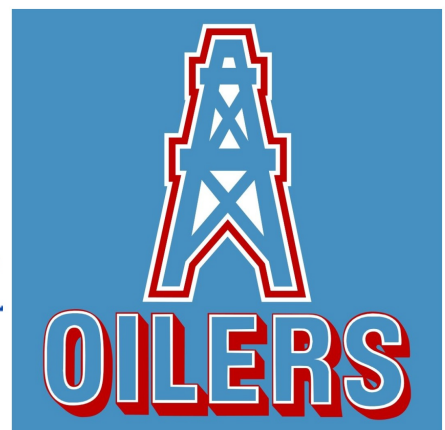


This presentation is dedicated to the loving memory of the three teams that changed their names/locations during the period of this corpus.

In [29]:

```
Image(filename = 'deadteams.png')
```

Out [29]:



As far as I can tell, there isn't much I can do in the way of error analysis, since the whole design of my project was to derive some values that, to the best of my knowledge, I obtained correctly. This project could be improved in a lot of ways. I ended up throwing away a lot of data about the date of each article, settling instead for average win percentage over the period of the corpus. I had originally planned to join the sentiment data with data about the team's Elo, as is publicly available in datasets from FiveThirtyEight. I did this for two reasons: to cut down on processing time, and because individual article sentiments could often have very little to do with the preceding week's action on the field. I felt that, especially since the *Times* doesn't do a lot of game-by-game recaps, normalizing the sentiment over time would lead to a more representative conclusion, albeit a less substantive one. This project could also do better in how it determined the sentiment when referring to a team. After all, I had an extremely high granularity of an article each to check the overall sentiment. Especially in a zero-sum game like basketball and football, this can change results. For example, saying "The Pistons absolutely destroyed the Wizards, who put up a pathetic 78 points" might return a very negative sentiment score, but it is a positive thing to write about the Pistons. Once again, I threw away this data about the reference of words to cut down on processing time (and I'm not sure I have the expertise to decode such patterns anyway.)

I believe this project is easily scalable, as long as processing power and acquiring text data are not an issue (which, to be fair, they always are). The *Times* corpus is written in NITF (News Industry Text Format), so any similarly-formatted dataset could be passed into the program (although one might have to repeat the moderate amount of cleaning I did through Unix.) The list of team names would have to be edited, since mine is specific to the time period of the given corpus, and teams often change names and locations (R.I.P, San Diego Chargers).

Acknowledgements and Citation

I'd just like to thank Will Styler for helping me formulate my ideas when I asked him about them far earlier in this process than I should've, and for sending me the magnificent *Times* corpus. I want to thank Eric Meinhardt for giving me the idea to store my dictionary of sports article sentiments in a JSON file, which has saved me a lot of processing time. I should also mention Sports Reference, which is one of my favorite websites in the world and which makes sports data collection so easy even I could do it.

Hutto, C.J. & Gilbert, E.E. (2014). VADER: A Parsimonious Rule-based Model for Sentiment Analysis of Social Media Text. Eighth International Conference on Weblogs and Social Media (ICWSM-14). Ann Arbor, MI, June 2014.

Sandhaus, Evan. The New York Times Annotated Corpus LDC2008T19. DVD. Philadelphia: Linguistic Data Consortium, 2008.

Sports Reference LLC. Basketball-Reference.com - Basketball Statistics and History.
<https://www.basketball-reference.com/>.

Sports Reference LLC. Pro-Football-Reference.com - Pro Football Statistics and History.
<https://www.pro-football-reference.com/>.