Sentiment Analysis in Python

AUTHOR
Nicola Rennie

Sentiment analysis is a natural language processing technique used to analyse whether textual data is positive, negative, or neutral. A common application is the analysis of customer feedback. This tutorial serves as a brief introduction to performing sentiment analysis in R.

Data

The data used here is from Week 19 of #TidyTuesday. The data can be downloaded from here. You can either read it into Python directly from the URL, or read in a locally saved copy. Here, I've saved a copy called `nyt_titles.tsv` in a directory one level up from my Python script.

The data is saved as a tab-separated values (.tsv) file, and we can read it into Python using the `read_csv()` function from `pandas`. We also need to specify that it is a *tab* separated file, rather than a *comma* separated file.

```
import pandas as pd
nyt_titles = pd.read_csv("../nyt_titles.tsv", sep="\t")
nyt_titles.head()
```

	id	title	author	year	total_weeks	first_week	debut_rank	best_rank
0	0	"H" IS FOR HOMICIDE	Sue Grafton	1991	15	1991-05-05	1	2
1	1	"I" IS FOR INNOCENT	Sue Grafton	1992	11	1992-04-26	14	2
2	10	''G'' IS FOR GUMSHOE	Sue Grafton	1990	6	1990-05-06	4	8
3	100	A DOG'S JOURNEY	W. Bruce Cameron	2012	1	2012-05-27	3	14

	id	title	author	year	total_weeks	first_week	debut_rank	best_rank
4	1000	CHANGING FACES	Kimberla Lawson Roby	2006	1	2006-02-19	11	14

This data set contains data on books that have been on the New York Times Bestsellers List. Information is included on the title, author, the year of release, the number of weeks spent on the list, the first week on the list, the book's debut rank, and it's best rank.

For this analysis we limit ourselves to the 1,000 most popular titles, in terms of the total number of weeks spent on the list. To extract these 1,000 titles, we use `nlargest()`. For the sentiment analysis here, we will analyse the sentiment of the titles. Therefore we extract only the `"title"` column.

```
text_data = nyt_titles.nlargest(n=1000, columns=['total_weeks'])
text_data = text_data[["title"]]
```

	title
2606	OH, THE PLACES YOU'LL GO!
4219	THE CELESTINE PROPHECY
4355	THE DA VINCI CODE
4139	THE BRIDGES OF MADISON COUNTY
3124	ALL THE LIGHT WE CANNOT SEE

Sentiment analysis

In python, there are two main libraries that we'll be using for sentiment analysis and generating word clouds. The `afinn` library gives a list of known words, and their sentiment score. For the afinn data, the sentiment is given as a value between -5 and 5, where -5 is very negative, 0 is neutral, and 5 is very positive. Secondly, the `wordcloud` package provides functionality for generating word cloud images, as well

as a list of stopwords. This stopwords data set contains a list of stop words - commonly used words such as “the”, “is” or “and”. We will later use the `stop_words` data to eliminate these unimportant words.

Before the sentiment analysis is carried out, our text data requires some pre-processing. Firstly, we wish to split up the titles into individual words (using `str.split()`), and make each row in our data frame relate to each word (using `explode()`).

```
text_data["title"] = text_data["title"].str.split()
text_data = text_data.explode("title")
```

In order to match our text data to the list of stopwords and words with sentiment scores, we also need to convert them all to lowercase using `str.lower()`, and remove any punctuation. Punctuation (and numbers) can be removed using `str.extractall()` and a regular expression. Here, we also re-format the data frame to make it a little more readable.

```
text_data["title"] = text_data["title"].str.lower()
text_data = text_data["title"].str.extractall(r"\s*([a-z@\d]+)[:\s]*")
text_data = text_data.droplevel(-1)
text_data = text_data.rename(columns={0: "word"})
text_data = text_data.reset_index()
```

	index	word
0	2606	oh
1	2606	the
2	2606	places
3	2606	you
4	2606	ll

To get the data frame of known words from the `afinn` library, we use the `Afinn()` function, and specify the language as English.

```
from afinn import Afinn
afn = Afinn(language = "en")
```

Before matching up our text data with the sentiments in the afinn data, we remove the stopwords. Here, we also import the libraries we'll need for the word clouds later, as they also come from the `wordcloud` package.

```
from wordcloud import WordCloud, STOPWORDS, ImageColorGenerator
stopwords = set(STOPWORDS)
text_data = text_data[~text_data.word.isin(stopwords)]
```

Now, we run through the list of words from the book titles, and match them up with their sentiment score. Words which have no match in the afinn database are given a neutral score of zero. We store the results in a data frame called `sentiments`.

```
words = tuple(text_data.word.tolist())
scores = [afn.score(word) for word in words]
sentiments = pd.DataFrame()
sentiments["word"] =  words
sentiments["scores"] = scores
```

	word	scores
0	oh	0.0
1	places	0.0
2	ll	0.0
3	go	0.0
4	celestine	0.0

To calculate the average sentiment across all titles, we extract the import the `statistics` library and calculate the mean of the `"scores"` column. The average sentiment is 0.04 - overall only slightly more positive than neutral!

```
import statistics as st
st.mean(sentiments["scores"])
```

```
0.037575150300601205
```

Word clouds

Word clouds are commonly used to visualise the sentiments, and number of instances of words in textual data. First, we can calculate the number of times each word is used in the titles. We use the `value_counts()` function from the `numpy` library to count the number of occurrences of each word.

```
import numpy as np
counts = pd.value_counts(np.array(words))
plot_data = pd.DataFrame()
plot_data["word"] =  counts.index
plot_data["n"] =  counts.values
```

To make our word cloud a bit nicer looking, we filter out any words that only occur once or twice.

```
plot_data = plot_data.query('n >= 3')
```

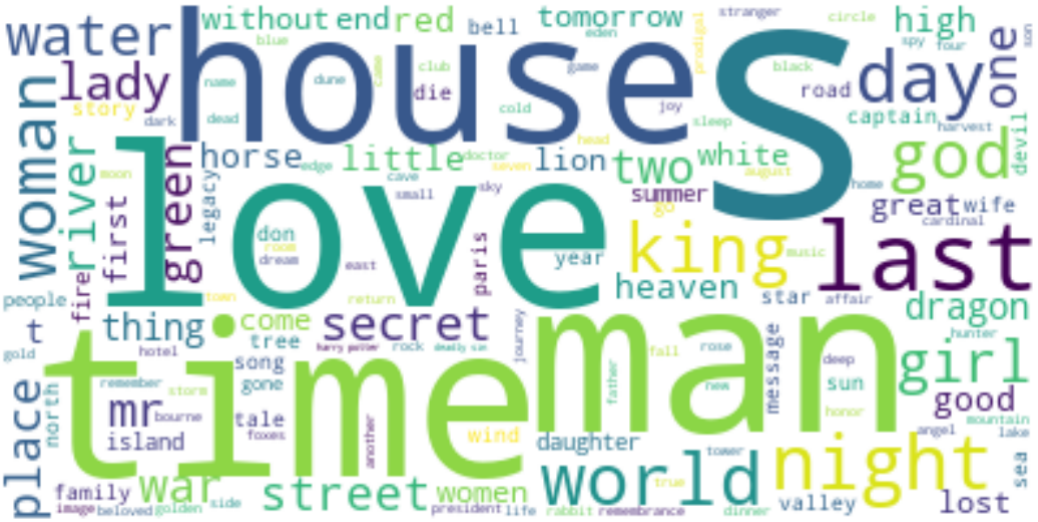
We can then join the sentiments data set to the counts data set using `merge()`, and we remove any duplicate rows where words are represented multiple times using `drop_duplicates()`.

```
plot_data = plot_data.merge(sentiments, on="word", how="left")
plot_data = plot_data.drop_duplicates()
```

	word	n	scores
0	s	48	0.0
48	man	13	0.0
61	love	12	3.0
73	time	12	0.0
85	night	10	0.0

Finally, we can create a word cloud showing the number of occurrences of each word. We've already loaded the `wordcloud` library, but we also need to use `matplotlib` for plotting.

```
import matplotlib.pyplot as plt
text = " ".join(i for i in text_data.word)
wordcloud = WordCloud(stopwords = stopwords,
                      background_color = "white").generate(text)
plt.figure(figsize=(7,5))
plt.imshow(wordcloud, interpolation='bilinear')
plt.axis("off")
plt.show()
```



And that's it! This was a simple introduction to sentiment analysis and word clouds in Python. There are many more extensions to what's covered here. For this data set, I'd be really interested in looking at the changing sentiments of titles over time, and colouring the word cloud by sentiment rather than randomly.

You can download the data used in this analysis from here, and the Python script is available on GitHub.