A Halloween treat: what is the best Dracula movie for you to watch?

case study of a novel and its numerous movie adaptations, using automatic extraction of sentiments and named entities with spaCy

Choosing a good movie to watch isn't as simple as finding the movie with the best score or our favorite genre

Another problem is spoilers

iconolocode / Dracula: best movie competition Published at Mar 12, 2022

depends highly on the watcher, their tastes, their mood and with who they watch it.

- Movie ratings and tags are great tools to make choices, but cinema is a form of art, and art resonates differently with each individual.
- This isn't about the *the* best movie, but the one that will interest *you* the most.

movie's mysteries. Isn't it more entertaining for the viewer to be the one to discover each plot twist?

- These reviewer's point of view often affects our perception of the movie. You know that feeling when someone points out a minor detail and now you can't unsee it. You don't want this.

Our approach

→ using tech to extract two key features: tone and content.

First we will look at the tone of movie. Do you want to see something fun? Serious? Tragic? A rollercoaster of emotions? Automatic sentiment analysis will help us to characterise this.

For the second point, content, you may wonder how we can handle this without spoilers. Well, we will just look at the content, only the *content* and not the story. To do this we will used named entity extraction.

Some context on this notebook

- This is an entry for the competition *Deepnote + spaCy: NLP in Notebooks*, under the theme *Challenge #2: Analyse sentiment from GoodReads vs IMDB reviews*
- Our focus thus is sentiments in books and movies.
- I wanted to focus on one of the cores of spaCy's uses: natural language processing. To highlight this, I didn't want to use a big corpora, but instead use a small set of texts that we will analyse meticulously.
- I study Art History and French Literature, so that explains why I choose for the latter approach. There is no good or wrong, I just took the path that compels me the most

and where my background can add an interesting layer to it.

- For this contest I wanted to show and explain some of my favorite spaCy features. So I'll walk with the reader through the NLP process and will be using an informal writing style to make it more catching.
- Actually, even if this is a notebook / article, I published it with the Deepnote dashboard format, because it offers more options to play with to make a nice layout.
- The techniques showcased here can be used on a larger dataset or to build a Deepnote app. These paths would lead to more findings or better usablity. But at this stage I want to focus on the library's features and educational aspects.
- I really want to explore these other options another time and am looking forward to see how the other participants implemented them :)

Preparation

This is a more boring part, but it's essential, so I'll write about it and you can chose to read or skip it.

Requirements

First you need to make sure that you have everything to make the program work. You may need to install some stuff. This can be done in the terminal. Here in Deepnote, there is a terminal tab, but we also have a really handy option: a `!` before a line allows it be be executed directly in the notebook. There is also a `requirements.txt` file, here it makes so that spacytextblob is available. If not, you should do `pip install spacytextblob`.

```
!python -m spacy download en_core_web_sm
```

```
Requirement already satisfied: pydantic!=1.8
Collecting typing-extensions<4.0.0.0,>=3.7.4
  Using cached typing_extensions-3.10.0.2-py
```

```
!python -m textblob.download_corpora
```

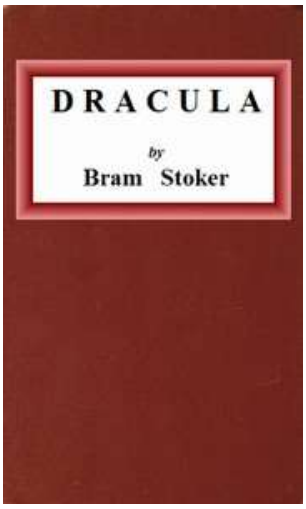
```
[nltk_data] Downloading package brown to /roc
```

```
import requests
import spacy
from spacytextblob.spacytextblob import SpacyTextBlob
from spacy import displacy
import csv
from matplotlib import pyplot as plt
import pandas as pd
from collections import Counter
```

First we need to get the novel, we can download the file manually or like this:

```
response = requests.get("https://gutenberg.org/files/345

if response.status_code == 200:
    response.encoding = 'utf-8'
    contents = response.text
    print(contents[:500])
```

```
The Project Gutenberg eBook of Dracula, by Bram Stoker

This eBook is for the use of anyone anywhere in the United S
most other parts of the world at no cost and with almost no
```



The novel's cover on Project Gutenberg

As you can see we can the novel's text without copyright problems. Lot of thanks to Project Gutenberg!

But these bits of text are not want we want to study. It's called the paratext and here, similarly to metadata, it gives information on the content

similarly to metadata, it gives information on the content.

We want only to have the novel's text, the story of Count Dracula. So let's find the start and the end.

```
text = contents[contents.find('\r\n\r\n\r\n\r\n\r\nDRACULA\r\n\r\n\r\n\r\n\r\n')
                                : contents.find('THE END')]
```

Preprocessing

We want to make sure the text is usable before analysing it.

Let's split it in chapters

```
chapters = text.split('CHAPTER')
```

```
print(chapters[0])
```

```


DRACULA
```

The first item in the chapters list isn't a chapter, but the title, let's remove it.

```
chapters.pop(0)
```

Let's look at the start and end to see how it is there

```
print(chapters[0][:50])
print(chapters[1][:50])
print(chapters[-1][:50])
```

```
 I

JONATHAN HARKER'S JOURNAL
```

We don't really want to have the chapter numbers. Assuming they are always formatted in the same way, we will remove the first line of each chapter.

```
chapters = [chap[chap.find('\r\n'):]
            for chap in chapters]
```

```
print(chapters[0][:50]
print(chapters[1][:50]
print(chapters[-1][:50
```

```


JONATHAN HARKER'S JOU
```

Now let's assemble back all the chapters together.

I also decided to remove the underscores that indicate text formatting and the old end of line symbol. These are replaced with a space to prevent words to be unwantedly glued together.

```
text = ''.join([chap.replace('_', ' ')
                .replace('\r\n', ' ')
                for chap in chapters])
```

In the preprocessing stage I often ask myself which parts of a text could be affect the analyses I want to do. Capitalisation and punctuation can affect our results. Do we want to treat 'Hello', 'hello', 'hello...' and 'hello!' as meaning the same or as being different.

Functions such as `.lower()` or `.isalnum()` in Python are handy for this.

We won't use them this time because spaCy handle these cases in an intelligent manner. Capitalisation and punctuation are useful to determinate where the sentences are and to communicate sentiments. So we don't want to remove them.

spacytextblob allows us to study sentiments, so we have to make sure we have it

```
nlp = spacy.load('en_core_web_sm')
nlp.add_pipe('spacytextblob')

print(nlp.pipe_names)
```

```
['tok2vec', 'tagger', 'parser', 'attribute_ruler', 'lemmatizer'
```

Now, let's start the main dish.

DraCuly

We feed the novel's text to the natural language processing pipeline:

```
doc_book = nlp(text)
```

Now we can look at the sentiments!

Sentiments

```
sentence_sentiments = [sentence._.polarity
                       for sentence
                       in doc_book.sents]
```

To have more insight on what these numbers reflect, let's print the corresponding sentences

```
sentence_sentiments[:10]
```

```
[0.0,
 0.0,
 -0.049999999999999996,
 0.40625,
 -0.017499999999999995,
 0.4833333333333334,
 0.31666666666666665,
 0.0,
 0.10333333333333332,
 0.0]
```

```
sent_dic = list(zip(sentence_sentiments, list(doc_book.sents)))
```

```
[(0.0,   JONATHAN HARKER'S JOURNAL  ( Kept in shorthand. )),
 (0.0,    3 May. Bistritz.),
 (-0.049999999999999996,
  --Left Munich at 8:35 P. M., on 1st May, arriving at Vienna early next morr
 (0.40625,
  Buda-Pesth seems a wonderful place, from the glimpse which I got of it from
 (-0.017499999999999995,
  I feared to go very far from the station, as we had arrived late and would
 (0.4833333333333334,
```

What do you think?

This is what it means:

- Sentences bad sentiments are negative
- and more happy sentences get a positive score

```
[(-1.0, What sort of grim adventur
 (-1.0,
  How was it that all the people a
 (-1.0,
  ( Mem. , this diary seems horrib
 (-1.0,
  God knows that there is ground t
 (-1.0,
```

```
[(1.0,
  I was told that this road is in
 (1.0,  "Indeed," I said, "you spe
 (1.0,
  The view was magnificent, and fr
 (1.0, Great God!),
 (1.0, I can fancy what a wonderfu
 (1.0,
```

-1 and 1 are the most extreme sentiments, but there is everything in between too.

0.0 scores however often are sentences that are neutral or where the sentiment is unclear. These are less interesting for our analysis, so we will keep them out.

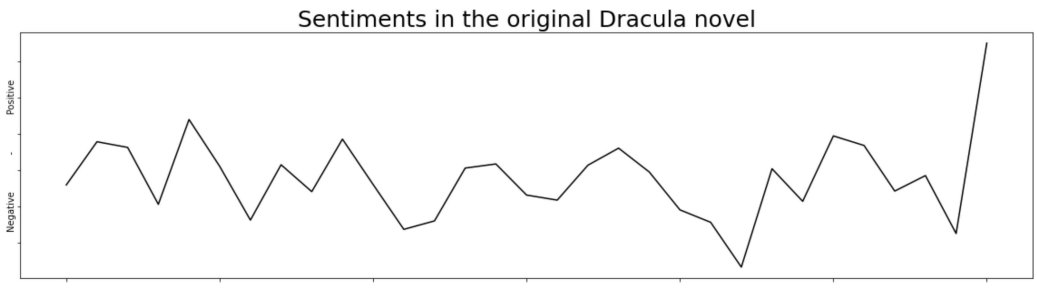
```
sentence_sentiments = [sentiment
                       for sentiment
                       in sentence_sentiments
                       if sentiment != 0]
```

```
sentence_sentiments[:10]
```

```
[-0.049999999999999996,
 0.40625,
 -0.017499999999999995,
 0.4833333333333334,
 0.31666666666666665,
```

Let's see what it looks like:

```
plot_sentiments_average(sentence_sentiments, 30, 'the original Dracula novel')
```



There are a lot of ups and downs in the story.

Around one third of the story things get really bad.

This is typical for the final act, where the protagonists must overcome a difficult challenge.

After that things get better.

What we saw just now is the text in the original Dracula novel.

```
book_plot_wiki = """Jonathan Harker, a new
Lucy Westenra's letter to her best friend,
Everyone stays at Dr. Seward's asylum as t
In Galatz, Romania, the hunters split up,
```

Let's apply the same technique on a description from Wikipedia of the novel:

```
in Galatz, Romania, the hunters split up.
```

```
doc_book_plot = nlp(book_plot_wiki)
```

We will use this again later on, so I made a function for it:

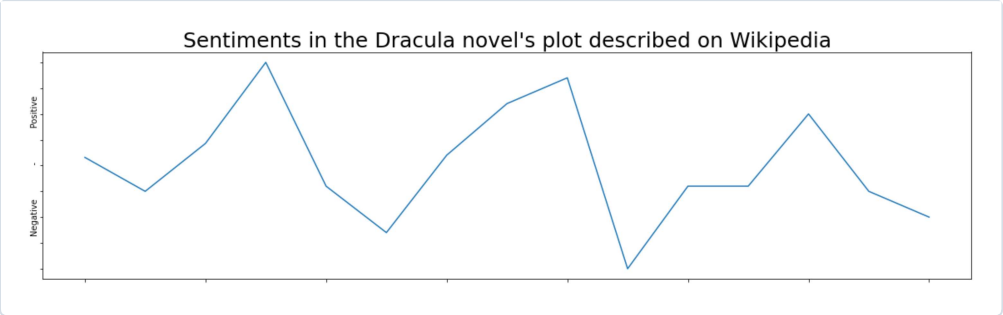
```
def calc_sentence_sentiments(doc):
    #retrieve sentiment scores
    temp = [sentence._.polarity for senten

    # remove null values
    temp = [sentiment for sentiment in tem

    return temp
```

```
[0.07878787878787878,
 -0.25,
 0.21428571428571427,
 1.0,
 -0.2,
```

```
wiki_sentiments = calc_sentence_sentiments
```

The summary is really short compared to the novel, which is actually the point for a summary.

Here are all the sentiments extracted from it:



We can see a big dip around on third of the story again.

Because the style of writing in a summary differs a lot from the novel, we can't really use these sentiments to make precise conclusions, but they still give an overall idea.

The end described on the Wikipedia page of the novel looks to have a more negative tone.

We can compute the average mood of the texts. Here are the sentiment scores of the origial Dracula novel and the Wikipedia plot summary. But due to the texts being different formats we can't really interpret these.

```
doc_book._.polarity
```

```
0.07495122086295743
```

```
doc_book_plot._.polari
```

```
-0.00766733266733266
```

Named entites recognition

spaCy can detect extract information from texts. This pipeline can be trained for specific goals, but the default one is sufficient for today.

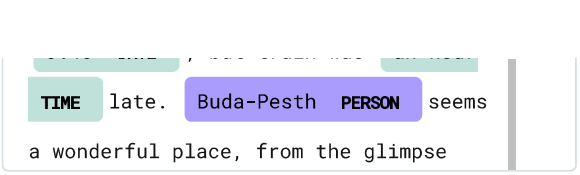
Here is what it looks like:

```
displacy.render(doc_book[:200], style='ent
```

JONATHAN HARKER'S PERSON JOURNAL ORG (Kept in shorthand.) 3 May. Bistritz. --Left Munich PERSON at 8:35 P. M. PERSON , on 1st May DATE , arriving at Vienna GPE early next morning TIME ; should have arrived at 6:46 DATE , but train was an hour

These entities are classified in different types of information.

We can filter for a specific type, such as characters, with the entity label `PERSON`:




```
characters_wiki = [ent.text for ent in doc_book_plot.ents if ent.label_ == 'PERSON']
characters_book = [ent.text for ent in doc_book.ents if ent.label_ == 'PERSON']
```

To compare the two texts, let's look at the most frequent person entities:

```
#novel's text
Counter(characte
```

```
[('Lucy', 27
 ('Van Helsi
 ('Jonathan'
 ('Harker',
 ('Arthur',
 ('Seward',
 ('Mina', 67
 ('Quincey',
 ('Morris',
```

```
#plot on Wiki
Counter(characte
```

```
[('Dracula',
 ('Lucy', 10
 ('Harker',
 ('Mina', 6)
 ('Jonathan
 ('Quincey',
 ('John Sewa
 ('Quincey M
 ('Arthur Ho
```

It is interesting to see that Wikipedia gives more attention to Dracula than in the book. This could also be because in the original text, Darcula is referenced in an indirect manner. This is something we can't know right now, or only if we start reading.

A cool thing we can do also is set theory!

We can look if in the summary there are characters that aren't in the book

```
print(set(characters_wiki) - set(characters_book))
```

```
set()
```

The resulting set is empty, it is quite obvious because else the summary is incorrect.

This would happen if it wasn't the case:

```
print(set(characters_wiki + ['another character']) - set(characters_book))
```

```
{'another character'}
```

My favorite entity type is `WORK_OF_ART` which returns titles of books, artworks and objects.

(I'm really biased because I study art and literature haha)

This entity isn't used showed really often in spaCy tutorials, probably because it is less useful than the others, but I wanted to shine some light on it.

It doesn't help the results aren't really accurate, it has some confusion with characters and speech expressions.

Mina seems to have artsy vibes in spaCy's eyes.

But these misclassifications are really interesting and fun.

```
set([ent.text for ent
in doc_book.ents
if ent.label_ == 'WORK_OF_ART'])
```

```
{'"Count Dracula',
 '"Dear Madam,--  ',
 '"Lord Godalming',
 '"The English Herr',
 '"The Host',
 '"The Pall Mall Gazette',
 '"The Szgany',
 '"The Westminster Gazette',
 '"We Szekelys',
 '"Westminster Gazette',
 '4 May. --I',
 'ARTHUR',
 'Amen'
```

They could be works of art, and some of them could be metal album names or horror movie titles!

note: I made a set to remove duplicates and order the strings

```
'At Purfleet',
'Bersicker',
'Blue',
'Charcot'
```

Times are also interesting to extract, looking quickly at them we can see that the story happens mostly at night!

```
print([ent.text for ent
in doc_book.ents
if (ent.label_ == 'TIME')])
```

```
['early next morning', 'an hour', 'the night', 'all night', 'forcemeat', 'a little t
```

We can look at the other entities in the summary to have some insights on the content of the story.

(Mina is quite onmipresent, she's a location here too)

```
for ent in doc_book_plot.ents
    if ent.label_ != 'PERSON'
        print(ent.text, ent.l
```

```
English NORP
the Carpathian Mountains EVEN
Count ORG
London GPE
Count ORG
Budapest GPE
England GPE
Whitby ORG
Budapest GPE
Van Helsing ORG
```

Okay, this is interesting, but we are not here to have a literature class on Bram Stoker's oeuvre. We want to find a good Halloween movie.

Movie adaptations

Screenplays are most of the time under copyright, so we will forget this, I am not a Law student.

Movie descriptions are made to sell and don't tell everything, so this isn't the best.

Luckily we have another option to get a glimpse in the content of a movie: Wikipedia~~!

Dataset

JustinR, *Wikipedia Movie Plots (Plot descriptions for ~35,000 movies)*, 2018: www.kaggle.com/jrobischon/wikipedia-movie-plots

```
SELECT * FROM plots
WHERE Title LIKE '%Dracula%'
```

The dataset has a lot of movies. Using SQLite I filtered out movies with the word Dracula in the titles, and saved them in a CSV file which I uploaded here.

There are a lot of movies that take inspiration from the Dracula legend and vampire myths. This is only a selection of them and I won't dig too deep today. But I still had to do something: add the *Nosferatu* (1922) movie, the first 'adaptation' of the novel, where the names were changed due to copyrigth. We can't miss on this classic.

I also did minor some changes, such as replacing the plot of *The Return of Dracula* with the correct one or removing the Bollywood version by Bhooshan Lal as this one sadly doesn't have a plot description on wiki (the plot of the book has used instead, even if

there's overlap it isn't the description of the movie). It's always important to check your data for errors!

SQL | Saved to variable `df_1`

```
SELECT *, LEN(Plot)
FROM '/work/dracula_nosferatu_plots.csv'
```

	ReleaseYear int… 1922 - 2014	Title object Dracula ..... 11.1% Dracula's … 3.7% 23 others 85.2%	Origin/Ethnic… American ..... 55.6% British ..... 33.3% 3 others 11.1%	Dir Ter Al 20
0	1931	Dracula	American	Tod
1	1936	Dracula's Daughter	American	Lam
2	1943	Son of Dracula	American	Rob
3	1945	House of Dracula	American	Erl
4	1958	The Return of Dracula	American	Pau
5	1966	Billy the Kid vs. Dracula	American	Wil Bea
6	1969	Blood of Dracula's…	American	Al
7	1971	Dracula vs. Frankenstein	American	Al
8	1979	Dracula	American	Joh

Deepnote allows us to explore the data in an interactive way. If you want you want, there are features to visualize it too.

We can see that Dracula movies aren't always Horror movie: the genre can be Fantasy, Western and Dance too.

Terence Fisher directed the most of Dracula movies.

It is also important to note that the length of the plot description varies. The shortest being around 800 characters and the longuest almost 7000.

For my own convenience I choose to use a nested lists format for the CSV file, but you can ude Pandas dataframes or SQL integrations too. These allow for extra features on Deepnote.

```
with open('/work/dracula_nosferatu_p
    reader = csv.reader(f)
    movie_plots = list(reader)
```

As there are a lot of movies, I made a class for them

I want each to have a distinct variable name, so I will use a combination of

```
class movie:
    instances = []

    def __init__(self, item, varname):
        self.name = varname
        self.release_year = item[0]
        self.title = item[1]
        self.plot = item[-1]
        self.__class__.instances.append(self)

    def spacy_magic(self):
        self.nlp = nlp(self.plot)
```

will use a combination of the release year and three first letters of the title

```
for film in movie_plots[1:]:
    #make name
    varname = film[0] + film[1][:3]

    #make movie object
    globals()[varname] = movie(film, varname)
```

Let's put their plot through the NLP pipeline

```
for film in movie.instances:
    film.spacy_magic()
```

We can now look at characters that are in the movies but not in the original book:

```
 Dracula: Prince of Darkness 1966 new characters:
{'Diana', 'Helen', 'Sandor', 'Klove', 'Alan', 'Charl

 Dracula Has Risen from the Grave 1968 new character
{'Barry Andrews', 'Zena', 'Barbara Ewing', 'Veronica

 Taste the Blood of Dracula 1969 new characters:
{'Jeremy', 'Courtley', 'Alice', 'Hargood', 'William

 Scars of Dracula 1970 new characters:
{'Sarah Framsen', 'Simon Carlson', 'Klove', 'Tania',

 Countess Dracula 1971 new characters:
{'Ilona', 'Fabio', 'Toth', 'Dobi', 'Julie', 'Elisabe

 Dracula AD 1972 1972 new characters:
{'Marsha Hunt', 'Laura Bellows', 'Stephanie Beacham'

 The Satanic Rites of Dracula 1974 new characters:
{'Valerie Van Ost', 'Torrence', 'Jessica', "Maurice

 Dracula: Pages from a Virgin's Diary 2002 new chara
```

Some have a totally different cast of characters!

Dracula: Pages from a Virgin's Diary seem to be the least different in terms of characters. The new names that spaCy found are due to different spellings (Jonathon or Jonathan, a 'Soon' or "s' that shouldn't be there).

We can also look where the stories take place:

```
for film in movie.instances:
    print('\n', film.title, film.release_year, 'focus:')
    focus = [ent.text for ent in film.nlp.ents if (ent.label_ in ('GPE', 'FA
    print(set(focus))
```

```
{'Diana', 'Count', 'Kents', 'Van Helsing', 'Helen', 'Ludwig', 'Alan', 'Karlsbad'}

 Dracula Has Risen from the Grave 1968 focus:
```

Or the overall tone of the plots

```
for film in movie.instances:
    print('\n', film.title, film.release_year, 'average plot mood: ', round
```

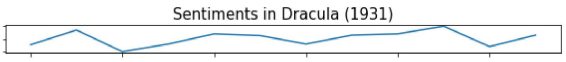
```
 Dracula Reborn 2012 average plot mood:  0.04
```

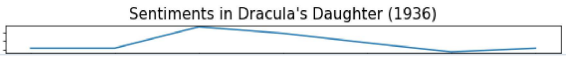
	id object	title object	mood float64
	1945Hou ..... 3.7% 1972Dra ..... 3.7% 25 others 92.6%	Dracula ..... 11.1% House of D… 3.7% 23 others 85.2%	-0.098897058823… 
3	1945Hou	House of Dracula	-0.098897058823 52944
22	1972Dra	Dracula AD 1972	-0.098351158645 27628
4	1958The	The Return of Dracula	-0.095652958152 95814
9	1992Bra	Bram Stoker's Dracula	-0.094640852974 1863
19	1969Tas	Taste the Blood of Dracula	-0.077209235209 23523

Even if it is a horror movie, *Billy the Kid vs. Dracula* seems to be the most positive one

We can also look at how the mood changes through the movies:

```
/shared-libs/python3.7/py-core/lib/python3.7/site-pa
  This is separate from the ipykernel package so we
```





With some widgets we can browse the content of the movies in a more fluid way, and without revealing the plot!

```
def display_only_entities(doc, types):
    previous = [] # to look for duplica

    for ent in doc.nlp.ents:
        if (ent.text in previous) == Fal
```

focus

Universe

movienum

15 / 28

ReleaseYear	2014
Title	Dracula Untold
Origin/Ethnicity	American
Director	Gary Shore
Cast	Luke Evans\nSarah Gadon\nDominic Cooper\nZach ...
Genre	fantasy
WikiPage	https://en.wikipedia.org/wiki/Dracula_Untold
Plot	Before the Renaissance, Vlad Ţepeş is the Prin...
len(plot)	4597

Name: 14, dtype: object

Renaissance ORG

the Prince of Wallachia ORG

Transylvania GPE

Sultan LOC

```
the Ottoman Empire  GPE
```

- If you get an error message, got to `view source` and run the code there

Conclusion

Now you have some insights in each movie. Your turn to make a choice.

What does pick your interest? The cast of new characters? The similarity to the original novel? A hectic or fun ride in sentiments? The universe where the story takes place? A victorian or modern setting?

To explore further

- How different are the movies from another?
- Can we identify specific topics?
- What do reviewers think of the movies?

More details on the NLP techniques:

SpacyTextBlob, https://spacytextblob.netlify.app/

Named Entities, https://spacy.io/usage/linguistic-features#named-entities

On plots and sentiments: https://melaniewalsh.github.io/Intro-Cultural-Analytics/05-Text-Analysis/04-Sentiment-Analysis.html#calculate-sentiment-scores-for-little-red-riding-hood

https://www.theparisreview.org/blog/2015/02/04/man-in-hole/

https://observablehq.com/@bmschmidt/book-visualizations-sandbox

On reviews:

https://post45.org/2021/04/the-goodreads-classics-a-computational-study-of-readers-amazon-and-crowdsourced-amateur-criticism/

Small extra's for you

The Helsing family is present in a lot of Dracula movies!

```
family = []
for film in movie.instances:
    family = family + [ent.te
    if ent.label_ == 'PERSON'
    and len(ent.text.split('

print(*set(family), sep='\n')
```

```
Lorrimer Van Helsing
Jessica Van Helsing
Abraham Van Helsing
Matthew Van Helsing
Leonardo Van Helsing
```

I uploaded in the files of this notebook a CSV with Dracula movie reviews that you can analyse with the NLP techniques we saw today.

(source dataset: Stefano Leone, www.kaggle.com/stefanoleone992/rotten-tomatoes-movies-and-critic-reviews-dataset, 2020)

```
SELECT rotten_tomatoes_link, review_content
FROM rotten_tomatoes_critic_reviews
WHERE rotten_tomatoes_link IN (
    SELECT rotten_tomatoes_link
        FROM rotten_tomatoes_movies
    WHERE movie_title LIKE '%Dracula%'
    OR movie_title LIKE '%Nosferatu%')
```