Table of Contents:

1. Overview

1.1 Background Information

Natural language processing (NLP) is a discipline of artificial intelligence that deals with the interaction between computers and humans through the natural language. The main objective of NLP is to analyze, decipher and to aid in better understanding the human language and forms of communication; verbal (voice commands to a machine), non-verbal (processing text data) or both (audio to text conversion). One of the many applications of NLP is to conduct sentiment analysis on text. Simply stated, sentiment analysis is the process of determining whether textual data is either positive, negative or neutral through the combined use of NLP and machine learning techniques. Sentiment analysis can be a valuable tool for large enterprises and content creators to gauge public opinion on their products and services, perform market research and monitor their reputation. A popular application of sentiment analysis involves it's use to analyze movie reviews. Movie rating websites such as IMDb and rotten tomatoes are used by both critics and viewers as a vehicle to express their love, appreciation or their distain for the latest movies and tv shows. Websites such the these can drastically affect the perfomance (box office), ratings and the general public's view of a movie or tv show, as many potential viewers often use the reviews and metrics provided by these websites to decide if something is worth watching. Sentiment analysis of a movie review can be used to determine whether it was positive or negative, which in turn, can be used to determine an overall rating.

1.2 Problem Statement

The objective of this project is to develop a model using the machine learning life cycle principles studied in this course to perform sentiment analysis on movie reviews. The model will take written reviews and classify them as expressing either a positive or negative sentiment. This document presents the deliverables required as part of milestone 1 which is the data description, preparation and exploration.

2. Data Description

The data used in the project is the Large Movie Review Dataset, obtained from the ai.standford.edu website. The dataset contains 25k reviews, which were subdivided into two separate folders, positive and negative. Each folder contains 12.5k reviews stored in a numbered text file. The reviews were obtained from IMDb, which is a website that has movie reviews, ratings and overall film and television content. Users are able to rate content on a scale of 1 to 10, for this dataset, the original authors decided to considers reviews with a rating greater or equal to 7 as being positive and reviews with a rating less than or equal to 4 as being negative. Reviews which had ratings equal to 5 or 6 were not incorporated into the dataset. Lastly, it is important to note that the data that was provided is balanced, as there is an equal amount of both positive and negative reviews.

For the purpose of this project, a subset of the original dataset was used for the data preparation, cleaning and exploration. We selected 7k reviews, 3.5k positive and 3.5k negative. Prior to the start of our analysis, the individual text files which contained the reviews were combined into a single tsv file titled Movie_Reviews.tsv.

3. Data Preparation and Cleaning

In [1]:
```
import pandas as pd
import numpy as np
import text_normalizer as tn
import model_evaluation_utils as meu

np.set_printoptions(precision=2, linewidth=80)
```

3.1 Load the dataset

In [4]:
```
dataset = pd.read_csv(r'Movie_Reviews.tsv',sep = '\t')
dataset = dataset.drop(columns=['Unnamed: 0'])
```

In [8]:
```
# take a peek at the data
print(dataset.head())
reviews = np.array(dataset['Review'])
sentiments = np.array(dataset['Sentiment'])
```

```
Review  \
0  Bromwell High is a cartoon comedy. It ran at the same time as some other pro
grams about school life, such as "Teachers". My 35 years in the teaching profes
sion lead me to believe that Bromwell High's satire is much closer to reality t
han is "Teachers". The scramble to survive financially, the insightful students
who can see right through their pathetic teachers' pomp, the pettiness of the w
hole situation, all remind me of the schools I knew and their students. When I
saw the episode in which a student repeatedly tried to burn down the school, I
immediately recalled ......... at .......... High. A classic line: INSPECTOR:
I'm here to sack one of your teachers. STUDENT: Welcome to Bromwell High. I exp
ect that many adults of my age think that Bromwell High is far fetched. What a
pity that it isn't!
1  Homelessness (or Houselessness as George Carlin stated) has been an issue fo
r years but never a plan to help those on the street that were once considered
human who did everything from going to school, work, or vote for the matter. Mo
st people think of the homeless as just a lost cause while worrying about thing
s such as racism, the war on Iraq, pressuring kids to succeed, technology, the
elections, inflation, or worrying if they'll be next to end up on the streets.<
br /><br />But what if you were given a bet to live on the streets for a month
without the luxuries you once had from a home, the entertainment sets, a bathro
om, pictures on the wall, a computer, and everything you once treasure to see w
hat it's like to be homeless? That is Goddard Bolt's lesson.<br /><br />Mel Bro
oks (who directs) who stars as Bolt plays a rich man who has everything in the
world until deciding to make a bet with a sissy rival (Jeffery Tambor) to see i
f he can live in the streets for thirty days without the luxuries; if Bolt succ
eeds, he can do what he wants with a future project of making more buildings. T
he bet's on where Bolt is thrown on the street with a bracelet on his leg to mo
nitor his every move where he can't step off the sidewalk. He's given the nickn
ame Pepto by a vagrant after it's written on his forehead where Bolt meets othe
r characters including a woman by the name of Molly (Lesley Ann Warren) an ex-d
ancer who got divorce before losing her home, and her pals Sailor (Howard Morri
s) and Fumes (Teddy Wilson) who are already used to the streets. They're surviv
ors. Bolt isn't. He's not used to reaching mutual agreements like he once did w
hen being rich where it's fight or flight, kill or be killed.<br /><br />While
the love connection between Molly and Bolt wasn't necessary to plot, I found "L
ife Stinks" to be one of Mel Brooks' observant films where prior to being a com
edy, it shows a tender side compared to his slapstick work such as Blazing Sadd
les, Young Frankenstein, or Spaceballs for the matter, to show what it's like h
aving something valuable before losing it the next day or on the other hand mak
ing a stupid bet like all rich people do when they don't know what to do with t
heir money. Maybe they should give it to the homeless instead of using it like
Monopoly money.<br /><br />Or maybe this film will inspire you to help others.
2  Brilliant over-acting by Lesley Ann Warren. Best dramatic hobo lady I have e
ver seen, and love scenes in clothes warehouse are second to none. The corn on
face is a classic, as good as anything in Blazing Saddles. The take on lawyers
is also superb. After being accused of being a turncoat, selling out his boss,
and being dishonest the lawyer of Pepto Bolt shrugs indifferently "I'm a lawye
r" he says. Three funny words. Jeffrey Tambor, a favorite from the later Larry
Sanders show, is fantastic here too as a mad millionaire who wants to crush the
ghetto. His character is more malevolent than usual. The hospital scene, and th
```

```
e scene where the homeless invade a demolition site, are all-time classics. Loo
k for the legs scene and the two big diggers fighting (one bleeds). This movie
gets better each time I see it (which is quite often).
3  This is easily the most underrated film inn the Brooks cannon. Sure, its fla
wed. It does not give a realistic view of homelessness (unlike, say, how Citize
n Kane gave a realistic view of lounge singers, or Titanic gave a realistic vie
w of Italians YOU IDIOTS). Many of the jokes fall flat. But still, this film is
very lovable in a way many comedies are not, and to pull that off in a story ab
out some of the most traditionally reviled members of society is truly impressi
ve. Its not The Fisher King, but its not crap, either. My only complaint is tha
t Brooks should have cast someone else in the lead (I love Mel as a Director an
d Writer, not so much as a lead).
4  This is not the typical Mel Brooks film. It was much less slapstick than mos
t of his movies and actually had a plot that was followable. Leslie Ann Warren
made the movie, she is such a fantastic, under-rated actress. There were some m
oments that could have been fleshed out a bit more, and some scenes that could
probably have been cut to make the room to do so, but all in all, this is worth
the price to rent and see it. The acting was good overall, Brooks himself did a
good job without his characteristic speaking to directly to the audience. Agai
n, Warren was the best actor in the movie, but "Fume" and "Sailor" both played
their parts well.

  Sentiment
0  positive
1  positive
2  positive
3  positive
4  positive
```

3.2 Normalize the dataset

In [1]:
```
# Normalize the data
norm_reviews = tn.normalize_corpus(reviews)
```

3.3 Save cleaned data to csv

In [2]:
```
# Save the normalized dataframe for later use
df = pd.DataFrame({'Reviews': norm_reviews, 'Sentiments': sentiments}, columns=['
df.to_csv('Movie_Reviews_Clean.csv')
```

4. Data Exploration

Load cleaned data

In [2]:
```
dataset = pd.read_csv('Movie_Reviews_Clean.csv')
#remove extra column
dataset = dataset.drop(columns=['Unnamed: 0'])
```

4.1 Exploring Text Categories

In [3]:
```
# list column names and datatypes
dataset.dtypes
```

Out[3]:
```
Reviews       object
Sentiments    object
dtype: object
```

In [4]:
```
#data frame length
len(dataset)
```

Out[4]:
```
7000
```

In [5]:
```
#number of values per column
dataset.count()
```

Out[5]:
```
Reviews       7000
Sentiments    7000
dtype: int64
```

In [6]:
```
# select a sample of some data frame columns
pd.set_option('display.max_colwidth',False)
dataset[['Reviews','Sentiments']] \
  .sample(2, random_state=42)
```

Out[6]:

	Reviews	Sentiments
6500	anybody really want understand hitler read wwi history not wwii history find happen war soldier live around dead corpse time many soldier go insane see wwi time call shellshocke call post traumatic stress disorder learn true horror wwi begin understand hitler understand human become desensitize death not evil simply way cope horror around movie unfortunately miss many read book subject watch movie path glory good wwi movie ever make see frustration soldier movie sense helplessness utter devaluation human life nothing bullet catcher movie miss really key point understand germany lose war million million germans lose life no real reason come utter economic collapse follow war factor create extremism loss family member massive poverty create always lead extremism unfortunately movie ignore factor become another throw away piece crap throw pile really no real value fictional movie base upon fictional character could give well idea hitler throw hitlers name would sell	negative
2944	production quality cast premise authentic new england waterbury ct locale lush john williams score result 3 4 star collector item unfortunately get passable 2 star decent flick mostly memorable try bring art house style film mainstream small town locale story ordinary people genre well satisfy grownup jane fonda unable hide braininess enough make character believable wonder not post doctorate yale instead work dead end factory job waterbury robert diniros character bit contrived illiterate nice guy loser turn actually little help janes character 1990 version henry ford thomas edison genre successfully handle nobodys fool mid 90 year 2003 schmidt wish main stream studio would try stuff post adolescent reserve couple screen multi cinema complex effort give effort	positive

In [7]:
```
# describe categorical columns of type np.object
dataset[['Reviews','Sentiments']] \
  .describe(include=np.object) \
  .transpose()
```

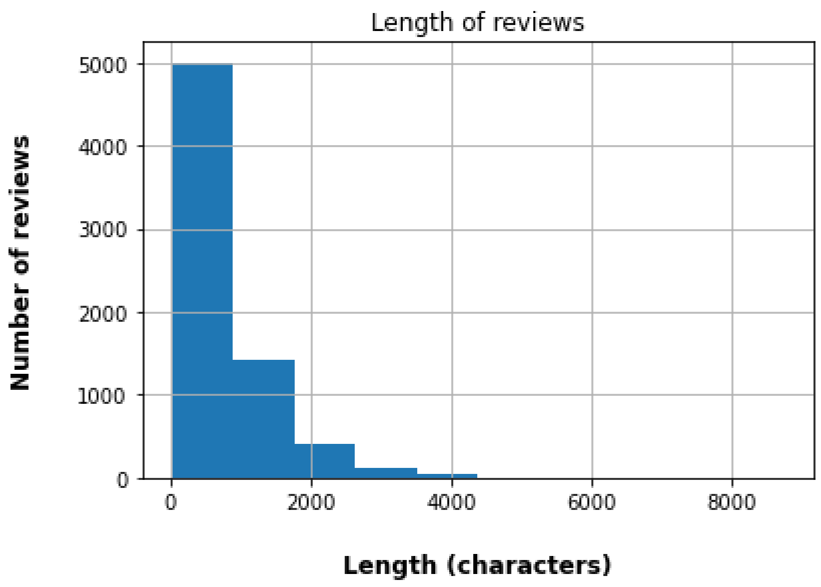
Out[7]:

	count	unique	top	freq
Reviews	7000	6988	get movie free job along three similar movie watch low expectation movie not bad per se get pay tale love betrayal lie sex scandal everything want movie definitely not hollywood blockbuster cheap thrill not bad would probably never watch movie nutshell kind movie would see either late night local television station want take time would see sunday afternoon local television station try take time despite bad acting cliche line sub par camera work not desire turn movie pretend like never pop dvd player story many time many movie one no different no well no bad average movie	3
Sentiments	7000	2	positive	3501

4.1.1 Creating Plots to count the length and words

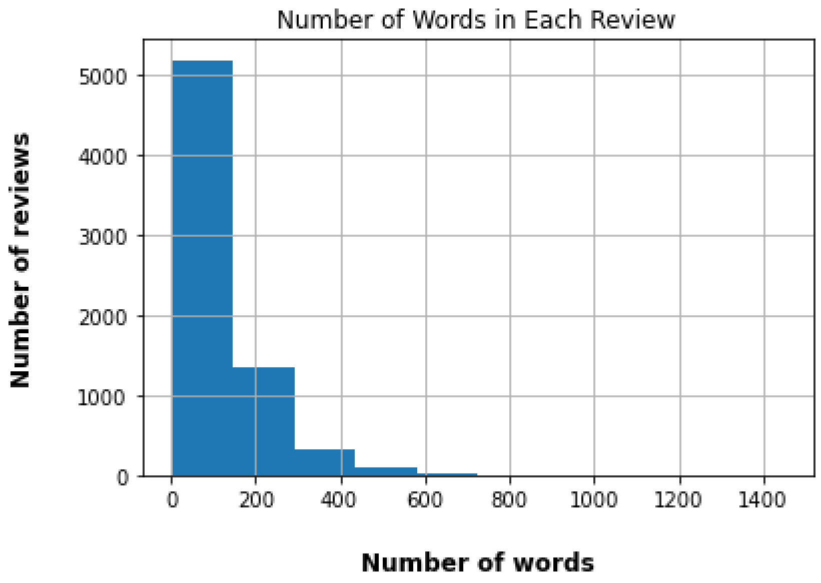
In [8]:
```
#check length of reviews
review_length_hist = dataset['Reviews'].str.len().hist()
review_length_hist.set_title("Length of reviews")
review_length_hist.set_xlabel("Length (characters)", labelpad=20, weight='bold',
review_length_hist.set_ylabel("Number of reviews", labelpad=20, weight='bold', si
```

Out[8]:
```
Text(0, 0.5, 'Number of reviews')
```



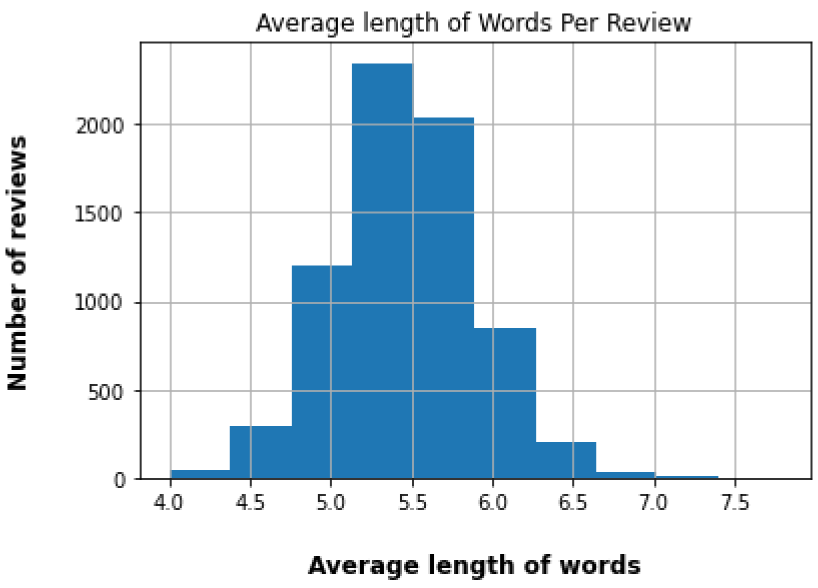
In [9]:
```
#number of words in each review
num_of_words_hist = dataset["Reviews"].str.split().\
    map(lambda x: len(x)).\
    hist()
num_of_words_hist.set_title("Number of Words in Each Review")
num_of_words_hist.set_xlabel("Number of words", labelpad=20, weight='bold', size=
num_of_words_hist.set_ylabel("Number of reviews", labelpad=20, weight='bold', siz
```

Out[9]:
```
Text(0, 0.5, 'Number of reviews')
```



In [10]:
```
#check average length of words in each sentence
import numpy as np
length_of_words_hist = dataset['Reviews'].str.split().\
   apply(lambda x : [len(i) for i in x]). \
   map(lambda x: np.mean(x)).hist()
length_of_words_hist.set_title("Average length of Words Per Review")
length_of_words_hist.set_xlabel("Average length of words", labelpad=20, weight='b
length_of_words_hist.set_ylabel("Number of reviews", labelpad=20, weight='bold',
```

Out[10]:
```
Text(0, 0.5, 'Number of reviews')
```



4.1.2 Checking for unique values

In [11]:
```
dataset['Reviews'].value_counts()[:10]
```

Out[11]:
```
get movie free job along three similar movie watch low expectation movie not ba
d per se get pay tale love betrayal lie sex scandal everything want movie defin
itely not hollywood blockbuster cheap thrill not bad would probably never watch
movie nutshell kind movie would see either late night local television station
want take time would see sunday afternoon local television station try take tim
e despite bad acting cliche line sub par camera work not desire turn movie pret
end like never pop dvd player story many time many movie one no different no we
ll no bad average movie
3
movie disgrace major league franchise live minnesota even not believe dump clev
eland yes realize time real indians pretty good twin take spot bottom american
league still consistent anyway love first major league like second always look
forward third indians would finally go way series not tell not plan second film
complete happen anyways true fan original major league favor not watch junk
2
smallville episode justice good episode smallville favorite episode smallville
2
enjoy watch well acted movie much well acted particularly actress helen hunt ac
tor steven weber jeff fahey interesting movie fill drama suspense beginning end
reccomend everyone take time watch make television movie excellent great acting
2
tasty little frenchman tell story alternately sad scary life affirming end brut
al finale know happen even though hope maybe even beleive would not utlimately
film great strength expertly play emotion expectation drop bomb see film theory
class usc back mid90 not easy find definitely worth hunt
2
sondra locke stinks film awful actress anyway unfortunately drag everyone else
include real life boyfriend clint eastwood drain clint eastwood think agree sta
r one one read script tell one go real snorer exceptionally weak story basicall
y no story plot add bored poor acting even normally good eastwood absolutely no
action except couple argument far concern film rank top heap natural sleep enha
ncer wow could film boring think watch paint dry grass grow may fun real stinke
r not bother one
2
german filmmaker ulli lommel manage task many horror fan think impossible unsea
ted fellow teuton uwe boll crown director bad horror film ever make lommel trul
y ed wood new millennium film shoddy laughable good bad ew proud embarrassed sa
y watch toto morbidly fascinated see low bar could set answer subterranean lomm
el dig pit bury fun begin cast international nobody someone live los angeles ev
ery auto mechanic doctor mailman actor screenwriter wait discover could easily
understand lommel manage find many wannabe actor willing spew ridiculous dialog
straight face main character villainous beat cop play german actor thick german
accent aside serial killer also old beat cop la despite fact stop innocent woma
n driver take custody drag home inexplicably top floor furniture warehouse plai
n sight rookie partner lapd refuse investigate go far physically attack one acc
user ninja style raid apartment set excruciatingly bad production designer budg
et apparently include enough money paint enough paint precinct 707 cardboard wa
ll since actor obviously unpaid non professional sad assortment european emigre
possibly deportee act native land bimbos mimbos desperate middle aged woman sin
ce little money spend set special efx location production value fair mention sp
ring genuine look police uniform sadly could not afford police car uniformed co
p cruise street shiny new mercury rental half story focus dirty deed deranged g
erman lapd officer futile effort two young rookie stop one young actor especial
ly pitiable actor whole mess even vague shot real career movie fit right rockab
```

```
illy hairdo tortured brando pose need see appreciate latter part film title get
zombie victim killer resurrect murder girl visit voodoo priestess protective sp
ell put not ask girl romania would resort voodooism anticipation murder accept 
lommel logic enjoy absurd ride much prolonged hand clawing straw cover roadside
grave zombie girl manage make appearance look exactly death maybe even pretty b
lack glamor make generously airbrush around eye look nothing like zombie look l
ike high fashion model ready runway point movie lommel borrow creative note lau
d countryman boll inject large dose cheesy euro trash techno soundtrack talk pr
ehistoric electronic bumblebee noise stuff may play ibiza disco lommel still yo
ung enough shake booty unlike zombie lommel girl speak function normal er mean 
become zombifie give auteur ample opportunity shower us golden dialog yes golde
n shower not spoil anything reveal shock end say perfectly tune rest masterpiec
e spirit ed wood live say geist    2
one disney good film enjoy watch often may easily guess outcome care plain fun 
escape 1 hour forty two minute not movie mean get away reality short time anywa
y cast sparkle delight magictrain
2
though structure totally different book tim krabbe write original vanishing spo
orloos overall feel except koolhovens style less business like lyric beginning 
great middle fine sting end surprise emotional ending could read several magazi
ne sex film beautifully never explicit lot warmth sometimes even humour shame a
merican film not open honoust one dutch film tend go edge come subject de grot 
stay always within boundary good taste de grot tell amazing story stretch 30 ye
ar leave cinema move ask film anyway film even give
2
everyone know zero day event think movie elephant not make us see guy show life
year throughout movie get like laugh even though totally know go give chill cau
se feel guilty cheer comment think cal sweet guy even though know go happen kno
w even end movie commit suicide decide count 3 4 think funny still horrified se
e head blow course get like wicked maybe feel like really normal guy not really
realize know imo main force movie make us realize friend relative anyone plan s
omething crazy not even notice movie good make feel bad not go sleep right stil
l little feeling stomach butterfly
2
Name: Reviews, dtype: int64
```

In [12]:
```python
# number of unique values = count distinct
dataset['Reviews'].nunique()
```

Out[12]:
```
6988
```

4.2 Exploring the Dataset as a Whole

4.2.1 Creating a list of tokens

In [13]:
```python
# Define tokenizer function
def my_tokenizer(text):
    return text.split() if text != None else []
```

In [14]:
```
# transform list of documents into a single list of tokens
tokens = dataset.Reviews.map(my_tokenizer).sum()
```

In [15]:
```
print(tokens[:200])
```

```
['bromwell', 'high', 'cartoon', 'comedy', 'run', 'time', 'program', 'school',
'life', 'teacher', '35', 'year', 'teaching', 'profession', 'lead', 'believe',
'bromwell', 'highs', 'satire', 'much', 'close', 'reality', 'teacher', 'scrambl
e', 'survive', 'financially', 'insightful', 'student', 'see', 'right', 'patheti
c', 'teacher', 'pomp', 'pettiness', 'whole', 'situation', 'remind', 'school',
'know', 'student', 'see', 'episode', 'student', 'repeatedly', 'try', 'burn', 's
chool', 'immediately', 'recall', 'high', 'classic', 'line', 'inspector', 'sac
k', 'one', 'teacher', 'student', 'welcome', 'bromwell', 'high', 'expect', 'man
y', 'adult', 'age', 'think', 'bromwell', 'high', 'far', 'fetch', 'pity', 'not',
'homelessness', 'houselessness', 'george', 'carlin', 'state', 'issue', 'year',
'never', 'plan', 'help', 'street', 'consider', 'human', 'everything', 'go', 'sc
hool', 'work', 'vote', 'matter', 'people', 'think', 'homeless', 'lost', 'caus
e', 'worry', 'thing', 'racism', 'war', 'iraq', 'pressure', 'kid', 'succeed', 't
echnology', 'election', 'inflation', 'worry', 'next', 'end', 'street', 'give',
'bet', 'live', 'street', 'month', 'without', 'luxury', 'home', 'entertainment',
'set', 'bathroom', 'picture', 'wall', 'computer', 'everything', 'treasure', 'se
e', 'like', 'homeless', 'goddard', 'bolts', 'lesson', 'mel', 'brooks', 'direc
t', 'star', 'bolt', 'play', 'rich', 'man', 'everything', 'world', 'decide', 'ma
ke', 'bet', 'sissy', 'rival', 'jeffery', 'tambor', 'see', 'live', 'street', 'th
irty', 'day', 'without', 'luxury', 'bolt', 'succeed', 'want', 'future', 'projec
t', 'make', 'building', 'bet', 'bolt', 'throw', 'street', 'bracelet', 'leg', 'm
onitor', 'every', 'move', 'not', 'step', 'sidewalk', 'give', 'nickname', 'pept
o', 'vagrant', 'write', 'forehead', 'bolt', 'meet', 'character', 'include', 'wo
man', 'name', 'molly', 'lesley', 'ann', 'warren', 'ex', 'dancer', 'get', 'divor
ce', 'lose', 'home', 'pal', 'sailor', 'howard']
```

4.2.2 Counting frequencies with a counter

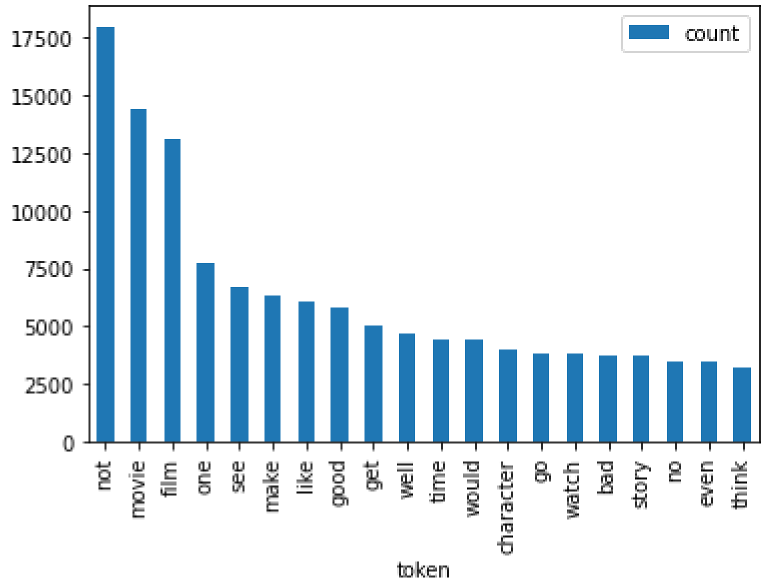
In [16]:
```
#count 20 most common tokens
from collections import Counter

counter = Counter(tokens)
counter.most_common(20)
```

Out[16]:
```
[('not', 17988),
 ('movie', 14396),
 ('film', 13101),
 ('one', 7767),
 ('see', 6709),
 ('make', 6338),
 ('like', 6110),
 ('good', 5852),
 ('get', 5031),
 ('well', 4676),
 ('time', 4463),
 ('would', 4406),
 ('character', 3982),
 ('go', 3824),
 ('watch', 3779),
 ('bad', 3763),
 ('story', 3759),
 ('no', 3460),
 ('even', 3426),
 ('think', 3215)]
```

In [17]:
```
print([t[0] for t in counter.most_common(200)])
```

```
['not', 'movie', 'film', 'one', 'see', 'make', 'like', 'good', 'get', 'well',
'time', 'would', 'character', 'go', 'watch', 'bad', 'story', 'no', 'even', 'thi
nk', 'really', 'show', 'scene', 'great', 'look', 'much', 'know', 'could', 'sa
y', 'first', 'people', 'give', 'also', 'take', 'way', 'come', 'love', 'find',
'thing', 'play', 'end', 'man', 'life', 'work', 'seem', 'year', 'two', 'plot',
'never', 'actor', 'many', 'want', 'little', 'may', 'try', 'ever', 'still', 'fee
l', 'old', 'director', 'woman', 'part', 'back', 'acting', 'use', 'lot', 'someth
ing', 'real', 'performance', 'funny', 'nothing', 'star', 'new', 'though', 'gu
y', 'tell', 'role', 'big', 'cast', 'us', 'become', 'horror', 'point', 'start',
'another', 'long', 'actually', 'day', 'turn', 'leave', 'young', 'fact', 'ever
y', 'comedy', 'world', 'quite', 'action', 'girl', 'act', 'happen', 'series', 'p
retty', 'minute', 'however', 'set', 'line', 'around', 'right', 'script', 'alway
s', 'family', 'enough', 'need', 'bit', 'fan', 'last', 'kill', 'enjoy', 'must',
'write', 'live', 'without', 'almost', 'friend', 'interesting', 'place', 'far',
'mean', 'keep', 'book', 'kind', 'effect', 'laugh', 'put', 'music', 'kid', 'sinc
e', 'believe', 'whole', 'least', 'anything', 'tv', 'lead', 'reason', 'away', 'o
riginal', 'hard', 'house', 'probably', 'call', 'help', 'let', 'episode', 'nam
e', 'audience', 'anyone', 'read', 'screen', 'sure', 'rather', 'course', 'idea',
'yet', 'fun', 'although', 'run', 'moment', 'child', 'true', '2', 'money', 'expe
ct', 'especially', 'job', 'second', 'everything', 'fall', 'night', 'problem',
'someone', 'bring', 'together', 'american', 'three', 'different', 'black', 'eve
ryone', 'sound', '10', 'sense', 'wife', 'excellent', 'version', 'main', 'move',
'maybe', 'recommend', 'high', 'beautiful', 'eye']
```

In [18]:
```
# convert list of tuples into data frame
frequency_dataset = pd.DataFrame.from_records(counter.most_common(20),
                                              columns=['token', 'count'])

# create bar plot
frequency_dataset.plot(kind='bar', x='token');
```

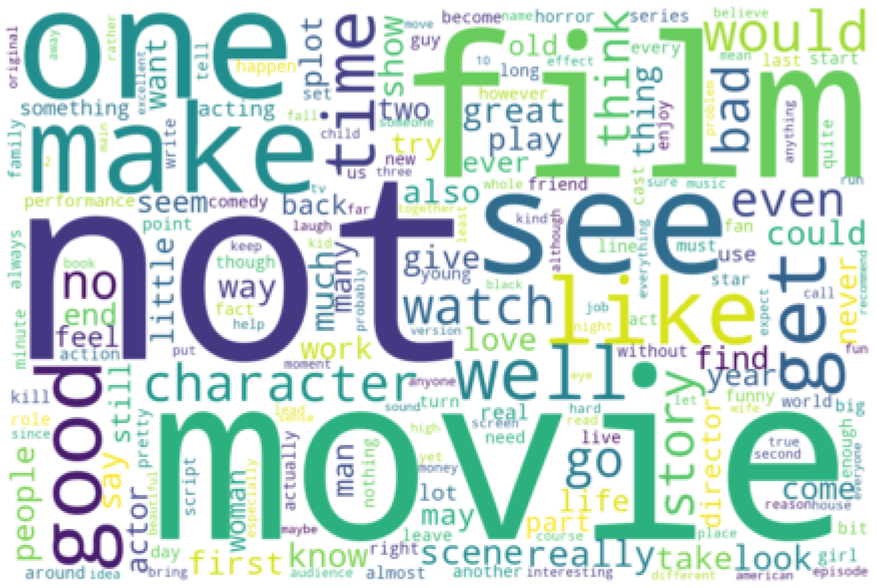


4.2.3 Creating a word cloud

In [19]:
```
%matplotlib inline
import matplotlib.pyplot as plt
```

In [20]:
```
from wordcloud import WordCloud

def wordcloud(counter):
    """A small wordloud wrapper"""
    wc = WordCloud(width=1200, height=800,
                   background_color="white",
                   max_words=200)
    wc.generate_from_frequencies(counter)

    # Plot
    fig=plt.figure(figsize=(6, 4))
    plt.imshow(wc, interpolation='bilinear')
    plt.axis("off")
    plt.tight_layout(pad=0)
    plt.show()
```

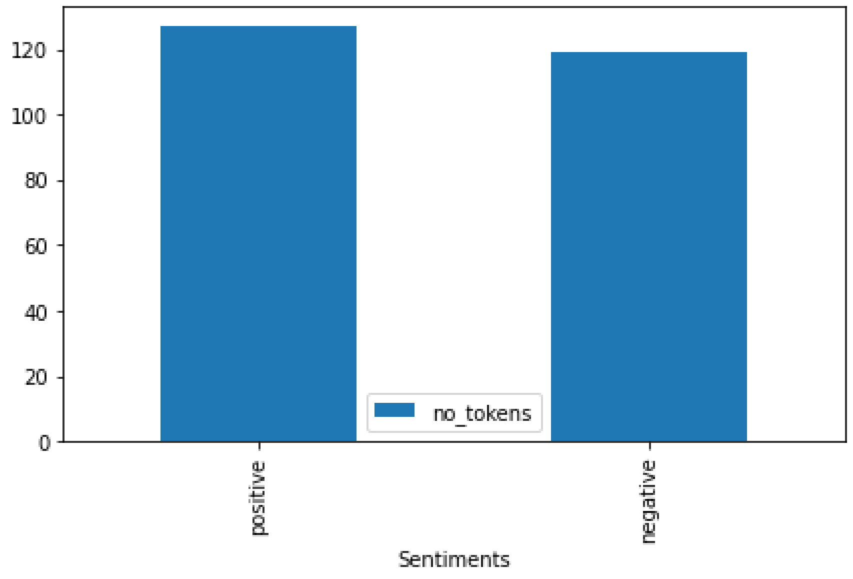
```
In [21]: # create wordcloud
         wordcloud(counter)
```



4.3 Exploring the Data Using Sentiments

```
In [22]: dataset['no_tokens'] = dataset.Reviews\
           .map(lambda l: 0 if l==None else len(l.split()))
```

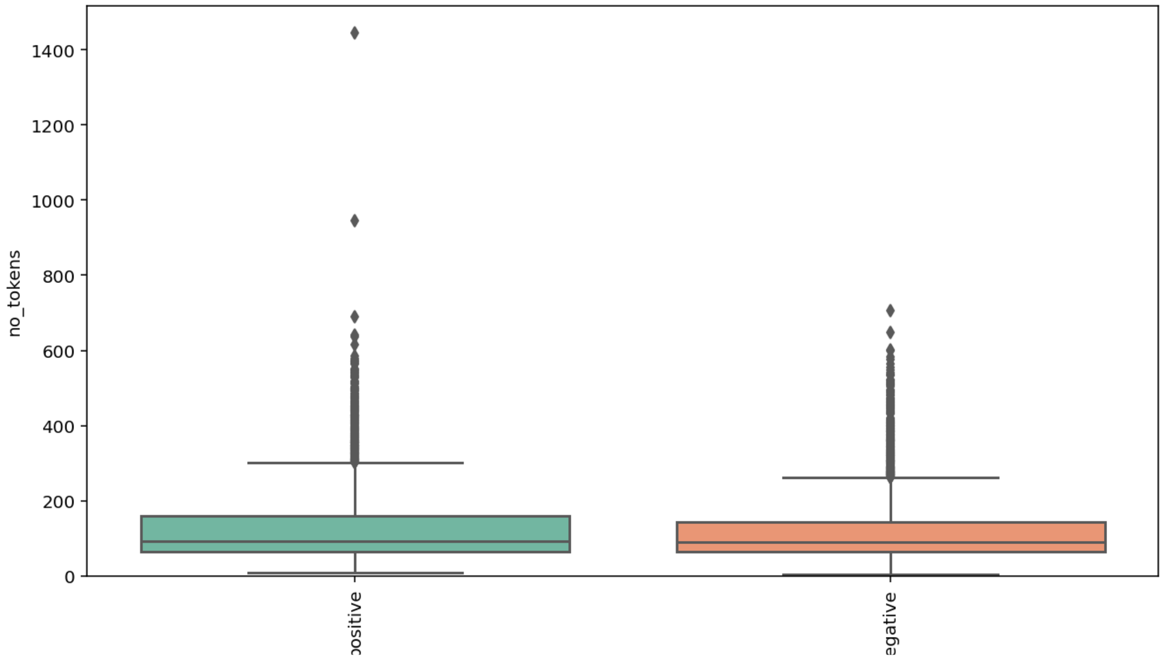
In [23]:
```
# mean number of tokens by category
dataset.groupby(['Sentiments']) \
  .agg({'no_tokens':'mean'}) \
  .sort_values(by='no_tokens', ascending=False) \
  .plot(kind='bar', figsize=(7,4));
```



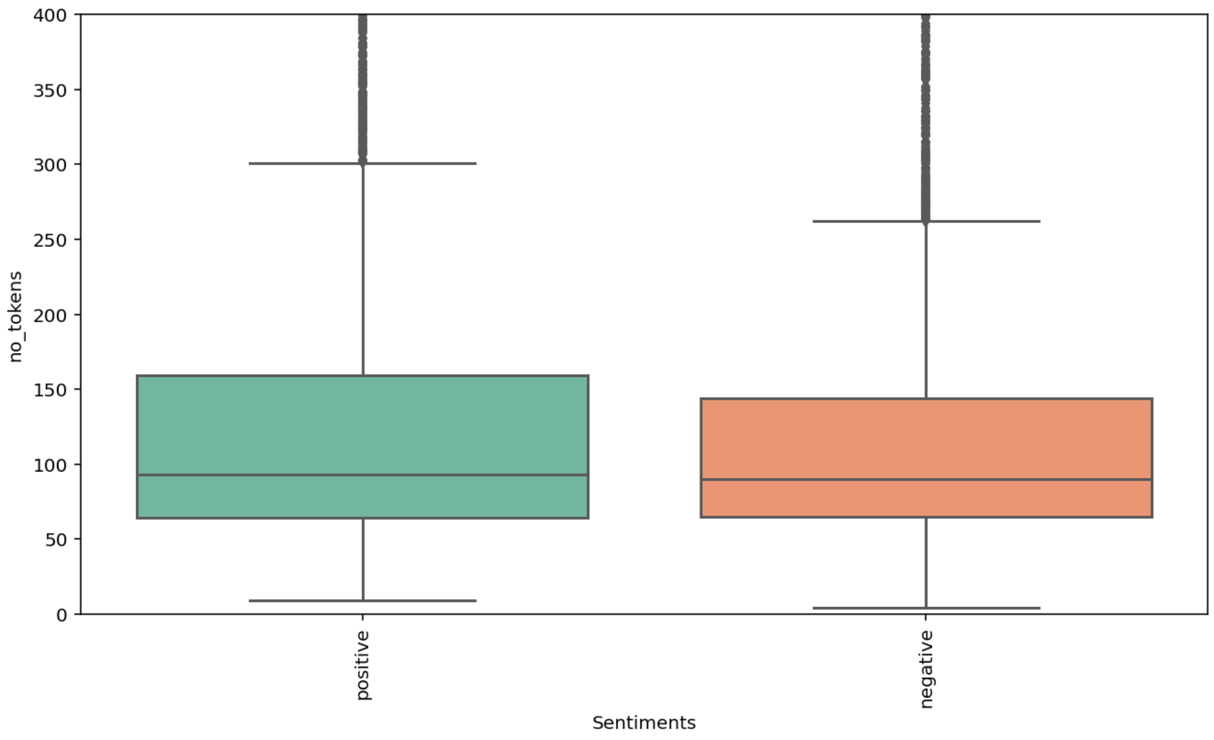
4.3.1 Outlier exploration

In [24]:
```
%config InlineBackend.figure_format = 'retina'

import seaborn as sns

def multi_boxplot(data, x, y, ylim = None):
    '''Wrapper for sns boxplot with cut-off functionality'''
    # plt.figure(figsize=(30, 5))
    fig, ax = plt.subplots()
    plt.xticks(rotation=90)

    # order boxplots by median
    ordered_values = data.groupby(x)[[y]] \
                         .median() \
                         .sort_values(y, ascending=False) \
                         .index

    sns.boxplot(x=x, y=y, data=data, palette='Set2',
                order=ordered_values)

    fig.set_size_inches(11, 6)

    # cut-off y-axis at value ylim
    ax.set_ylim(0, ylim)
```

```
In [25]: multi_boxplot(dataset, 'Sentiments', 'no_tokens');
```



```
In [26]: # cut-off diagram at y=400
         multi_boxplot(dataset, 'Sentiments', 'no_tokens', ylim=400)
```



4.3.3 Exploring the dataset by observing each sentiment separately

In [27]:
```
# create a data frame slice (positive and negative)
sub_data_positive = dataset[dataset['Sentiments']=='positive']
sub_data_negative = dataset[dataset['Sentiments']== 'negative']
```

In [28]:
```
# transform list of documents into a single list of tokens for a subset of the re
tokens_positive = sub_data_positive.Reviews[:].map(my_tokenizer).sum()
tokens_negative = sub_data_negative.Reviews[:].map(my_tokenizer).sum()

# Write positive tokens to a text file
with open('positive_tokens.txt', 'w') as f:
    for item in tokens_positive:
        f.write("%s\n" % item)

# Write positive tokens to a text file
with open('negative_tokens.txt', 'w') as f:
    for item in tokens_negative:
        f.write("%s\n" % item)
```

In [53]:
```
# Determining the number of tokens which are common between positive and negative
common_tokens = set(tokens_positive).intersection(set(tokens_negative))
# Determine the total number of tokens
total_tokens = len(tokens_positive) + len(tokens_negative)
total_tokens
```

Out[53]:
```
863405
```

In [54]:
```
# Calculated the percentage of tokens which are common
p_common = round((len(common_tokens)/total_tokens)*100,2)
print(str(p_common)+'%')
```

```
1.73%
```

In [65]:
```
# Print common tokens
count = 0
for elem in iter(common_tokens):
    count = count + 1
    if count == 20:
        break
    print(elem)
```

```
console
inference
kimberly
comparison
naivety
ungrateful
crotch
marvel
corrupt
damn
gruff
capital
likewise
immoral
heartbreake
last
center
elephant
fateful
```

4.3.3.a Counting frequencies

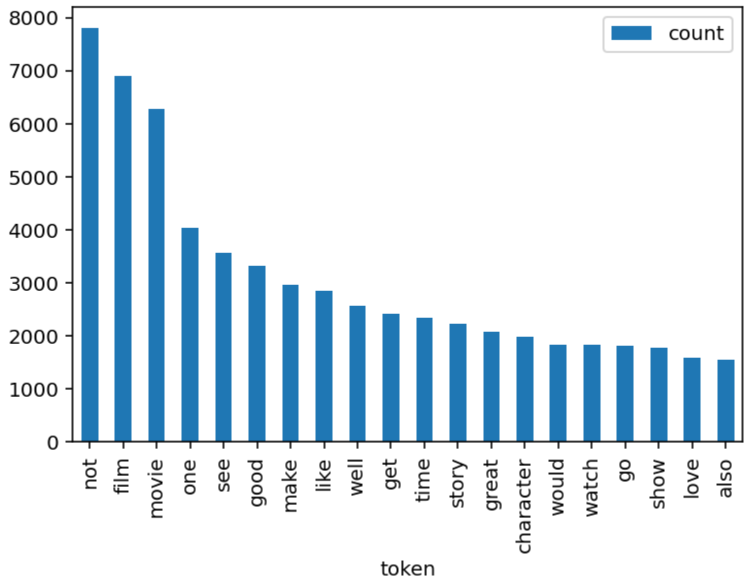
In [29]:
```
from collections import Counter

# Display the 20 most common tokens for positive reviews
counter_positive = Counter(tokens_positive)
counter_positive.most_common(30)
```

Out[29]:
```
[('not', 7808),
 ('film', 6899),
 ('movie', 6272),
 ('one', 4029),
 ('see', 3558),
 ('good', 3316),
 ('make', 2969),
 ('like', 2841),
 ('well', 2560),
 ('get', 2407),
 ('time', 2346),
 ('story', 2227),
 ('great', 2084),
 ('character', 1974),
 ('would', 1828),
 ('watch', 1825),
 ('go', 1804),
 ('show', 1766),
 ('love', 1592),
 ('also', 1550),
 ('think', 1522),
 ('really', 1464),
 ('scene', 1380),
 ('first', 1366),
 ('even', 1363),
 ('know', 1318),
 ('much', 1292),
 ('take', 1281),
 ('give', 1275),
 ('life', 1274)]
```

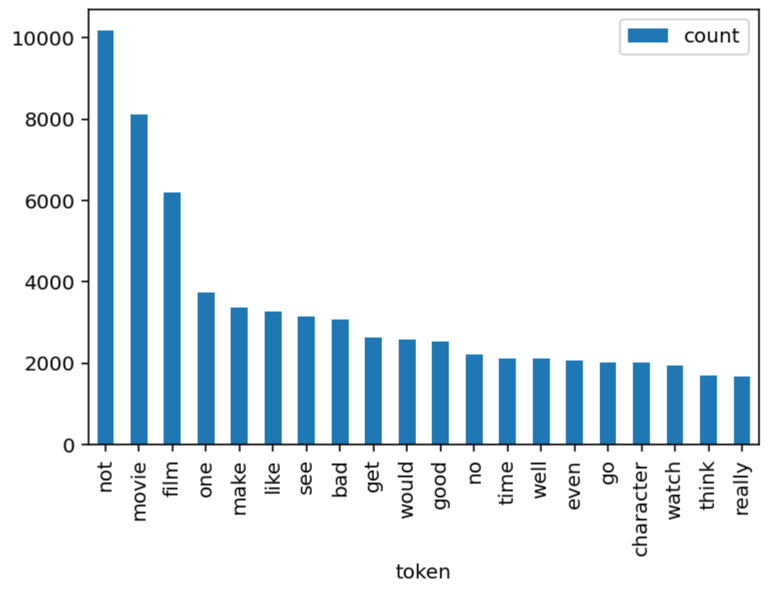
In [30]:
```
# Display the 20 most common tokens for negative reviews
counter_negative = Counter(tokens_negative)
counter_negative.most_common(30)
```

Out[30]:
```
[('not', 10180),
 ('movie', 8124),
 ('film', 6202),
 ('one', 3738),
 ('make', 3369),
 ('like', 3269),
 ('see', 3151),
 ('bad', 3083),
 ('get', 2624),
 ('would', 2578),
 ('good', 2536),
 ('no', 2208),
 ('time', 2117),
 ('well', 2116),
 ('even', 2063),
 ('go', 2020),
 ('character', 2008),
 ('watch', 1954),
 ('think', 1693),
 ('really', 1686),
 ('could', 1608),
 ('look', 1607),
 ('story', 1532),
 ('scene', 1516),
 ('say', 1424),
 ('much', 1417),
 ('know', 1305),
 ('people', 1299),
 ('thing', 1245),
 ('give', 1229)]
```

In [31]:
```
import matplotlib.pyplot as plt

# Plot frequency of 20 most common tokens for positive reviews
# convert list of tuples into data frame
freq_df_pos = pd.DataFrame.from_records(counter_positive.most_common(20),
                                        columns=['token', 'count'])

# create bar plot
freq_df_pos.plot(kind='bar', x='token');
```



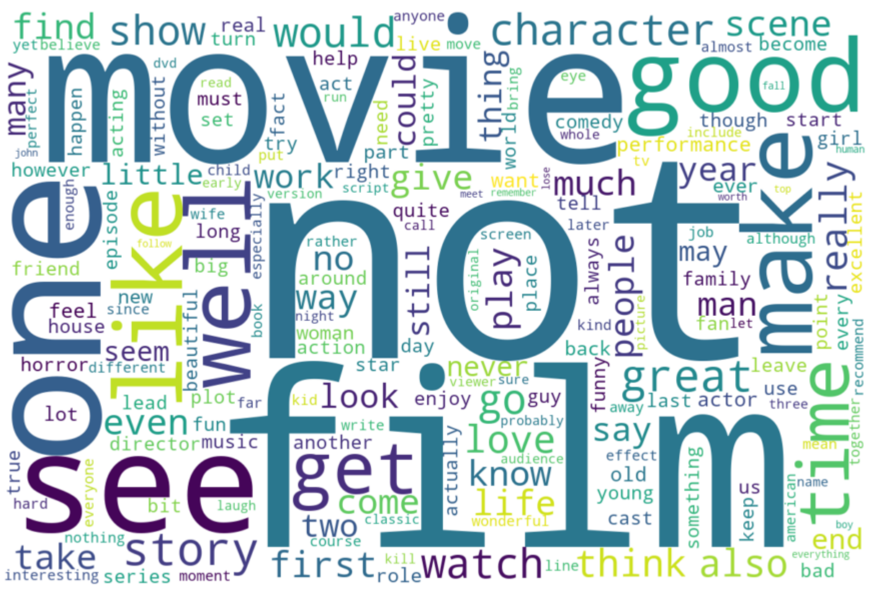
In [32]:
```
# Plot frequency of 20 most common tokens for negative reviews
# convert list of tuples into data frame
freq_df_neg = pd.DataFrame.from_records(counter_negative.most_common(20),
                                        columns=['token', 'count'])

# create bar plot
freq_df_neg.plot(kind='bar', x='token');
```



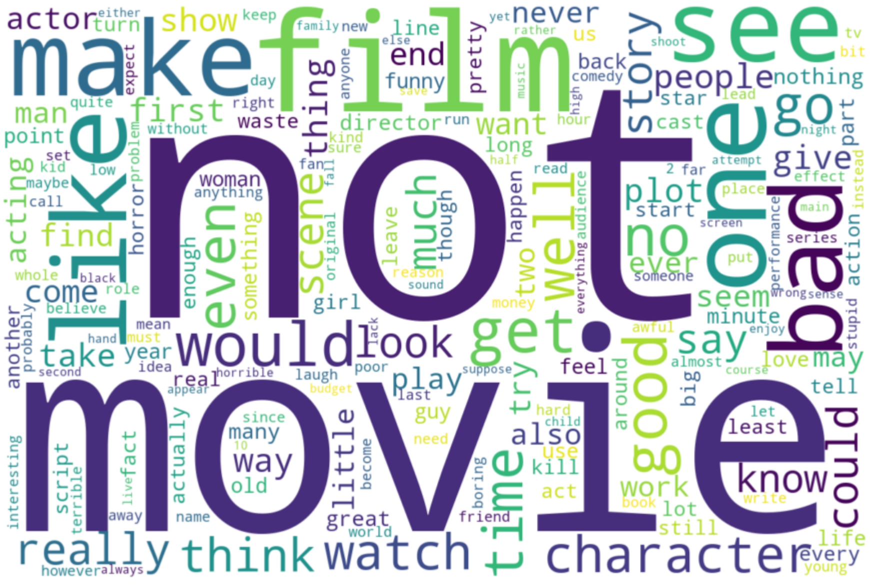
4.3.3.b Creating WorldClouds

In [33]:
```
from wordcloud import WordCloud
def wordcloud(counter):
    """A small wordloud wrapper"""
    wc = WordCloud(width=1200, height=800,
                   background_color="white",
                   max_words=200)
    wc.generate_from_frequencies(counter)

    # Plot
    fig=plt.figure(figsize=(6, 4))
    plt.imshow(wc, interpolation='bilinear')
    plt.axis("off")
    plt.tight_layout(pad=0)
    plt.show()
```

In [34]:
```
# Display wordcloud associated to positive reviews
wordcloud(counter_positive)
```



In [35]:
```
# Display wordcloud associated to negative reviews
wordcloud(counter_negative)
```



5. Summary of Key Findings

In this section we summarize key observations which were derived from the data exploration:

1. Looking at the wordclouds, we could observe that some of the most frequeny words used in both the positive and negative reviews were similar or common. We examined the percentage of tokens which were common and determined that approximately 1.73% of the tokens were common.
2. Based on the observations in the first point, in the next phase of the project, it might be beneficial to consider using bag of n-grams to better capture the sentiment of the review.
3. The vast majority of the reviews contain between 0 and 2000 characters, which corresponds to approximately 0 to 300 words. Additionally, the average word length present in our dataset was 5 characters.