Audience Reviews

The Last Airbender Season 1 (Netflix)

- This project involves analyzing audience reviews from Rotten Tomatoes for season 1 of The Last Airbender. At the time of writing this, the audience score is 75%. That's like 3.75 stars. That could mean people generally liked it, or overall people think season 1 was pretty good. However, this metric alone is not very informative.

- The goal is to try and find out *what* people liked or disliked about the first season. The idea here is to find words of high interest from the reviews, select some of those words as topics to be explored, and analyze only the parts of the reviews containing those topics.

- A review may be something like 'The acting is great. However the CGI is terrible'. If the selected topic is 'acting', the string will be split and only contain 'The acting is great'. From there, sentiment analysis will be done to see if the feelings are more negative or positive regarding that topic. Word clouds of adjectives around the topics will also be produced to aid in understanding the reviews.

- I noticed that in these reviews, thoughts and opinions are often split up with periods, exclamation points and question marks. So that is how strings containing our topics will be split. While this method will not be perfect, and these split strings will sometimes contain multiple topics, it should work fairly well for isolating them.

- The sentiments are being determined with a Logistic Regression pipeline which was created in the IMDB_Logistic_Regression notebook.

- I scraped reviews from Rotten Tomatoes and that is the data being used.

Contents

Libraries

In [1]:
```
import pandas as pd

import joblib

import matplotlib.pyplot as plt
from wordcloud import WordCloud

import re
import string
from nltk.corpus import stopwords
from collections import Counter
import itertools
from nltk.tokenize import word_tokenize
from nltk.stem.porter import PorterStemmer
import nltk
```

Data

In [2]:
```
df = pd.read_csv('last_airbender.csv')
```

In [3]:

```
df
```

Out[3]:

	audience-reviews__review
0	It was well done as compared to the anime. Cas...
1	For the most part this hits the spot. Don’t ge...
2	Me a encantado es difícil recrear una serie de...
3	Overall, it's a pretty meh show that had a lot...
4	I think the show did a lot of things well. I l...
...	...
1355	Honestly nothing I will be rewatching, i found...
1356	Pros: Great visuals and stellar acting by Prin...
1357	I think the show missed a few story arc of the...
1358	I was so excited at first... the bending looks...
1359	Often laughably bad, why is Aang so depressed?...

1360 rows × 1 columns

In [4]:

```
reviews = df['audience-reviews__review']
```

Prepare Text

In [13]:

```
class Prep_Text:
    def __init__(self,text):
        self.text = text

    def to_lower(self):
        self.text =  [x.lower() for x in self.text]

    def remove_punc(self):
        self.text = [''.join(char for char in x if char not in string.punctuation) for x in self.text]
    def remove_stop_words(self):
        stop_words = set(stopwords.words('english'))
        self.text  =  [' '.join(x for x in word_tokenize(words) if x not in stop_words) for words in self.text]

    def get_stems(self):
        porter = PorterStemmer()
        self.text =  [' '.join(porter.stem(word) for word in word_tokenize(char)) for char in self.text]
        return self.text

    def get_words(self):
        words = ' '.join(self.text)
        return word_tokenize(words)
```

In [14]:
```
#run methods
prep = Prep_Text(reviews)
prep.to_lower()
prep.remove_punc()
prep.remove_stop_words()
prep.get_stems();
```

Select Topics of High Interest

In [15]:
```
words = prep.get_words()
Counter(words).most_common(150)
```

```
Out[15]: [('show', 1046),
          ('origin', 881),
          ('seri', 682),
          ('charact', 640),
          ('’', 568),
          ('like', 556),
          ('season', 556),
          ('anim', 469),
          ('adapt', 445),
          ('stori', 438),
          ('watch', 436),
          ('good', 412),
          ('great', 412),
          ('love', 397),
          ('realli', 363),
          ('action', 338),
          ('episod', 337),
          ('live', 324),
          ('cast', 290),
          ('see', 285),
          ('chang', 285),
          ('enjoy', 276),
          ('better', 267),
          ('much', 264),
          ('time', 263),
          ('act', 262),
          ('feel', 260),
          ('make', 255),
          ('aang', 254),
          ('get', 252),
          ('avatar', 252),
          ('fan', 236),
          ('2', 234),
          ('well', 225),
          ('look', 224),
          ('made', 222),
          ('one', 219),
          ('hope', 218),
          ('cartoon', 217),
          ('think', 214),
          ('thing', 202),
          ('actor', 201),
          ('also', 196),
          ('would', 189),
          ('de', 185),
          ('overal', 181),
          ('la', 175),
          ('way', 171),
          ('could', 168),
          ('even', 164),
          ('scene', 162),
          ('bad', 158),
          ('movi', 156),
          ('que', 156),
          ('lot', 155),
          ('amaz', 155),
          ('perfect', 150),
          ('still', 149),
          ('zuko', 144),
          ('dont', 142),
          ('new', 140),
          ('first', 140),
          ('go', 138),
          ('expect', 138),
          ('peopl', 133),
          ('bend', 132),
          ('need', 131),
          ('katara', 131),
          ('visual', 130),
          ('world', 129),
          ('want', 126),
          ('im', 123),
          ('sokka', 122),
          ('plot', 117),
          ('littl', 116),
          ('cgi', 116),
          ('job', 114),
          ('wait', 114),
          ('effect', 113),
          ('part', 112),
          ('materi', 110),
          ('version', 107),
          ('mani', 107),
          ('done', 106),
          ('bit', 106),
          ('come', 105),
          ('moment', 105),
          ('write', 104),
          ('tri', 103),
          ('know', 98),
          ('develop', 97),
          ('improv', 97),
          ('tell', 97),
          ('give', 97),
          ('fun', 96),
          ('iroh', 94),
          ('work', 93),
          ('sourc', 92),
          ('felt', 92),
          ('pretti', 91),
          ('dialogu', 91),
          ('best', 90),
          ('far', 88),
          ('everyth', 87),
          ('remak', 87),
          ('seem', 87),
          ('next', 87),
          ('cant', 86),
          ('take', 86),
          ('say', 86),
          ('didnt', 82),
          ('absolut', 82),
          ('differ', 82),
          ('es', 80),
          ('second', 80),
          ('start', 80),
          ('understand', 78),
          ('disappoint', 78),
          ('forward', 78),
          ('point', 77),
          ('definit', 77),
          ('thought', 76),
          ('person', 75),
          ('life', 74),
          ('actual', 73),
          ('line', 73),
          ('hard', 73),
          ('compar', 72),
          ('beauti', 72),
          ('complet', 72),
          ('1', 72),
          ('though', 72),
          ('keep', 72),
          ('storylin', 71),
          ('costum', 70),
          ('kid', 70),
          ('put', 70),
          ('miss', 69),
          ('someth', 68),
          ('never', 68),
          ('set', 68),
          ('rush', 68),
          ('last', 68),
          ('azula', 67),
          ('true', 66),
          ('ad', 66),
          ('howev', 66),
          ('emot', 65),
          ('everi', 65),
          ('liveact', 65)]
```

- Topics will be 'charact'(640), 'stori'(438), 'act'(262), 'cgi'(116), 'dialogu'(91)

Subset Reviews Around Topics of High Interest

```
In [19]: reviews_split = [re.split('[.!?]', x) for x in prep.text]

         story      = [part for sublist in reviews_split for part in sublist if 'stori' in part]
         characters = [part for sublist in reviews_split for part in sublist if 'charact' in part]
         acting     = [part for sublist in reviews_split for part in sublist if 'act' in part]
         cgi        = [part for sublist in reviews_split for part in sublist if 'cgi' in part]
         dialogue   = [part for sublist in reviews_split for part in sublist if 'dialogu' in part]
```

```
In [34]: story[:3]
```

```
Out[34]: ['part hit spot ’ get wrong ’ hit high sourc materi noth ever go compar genuin good adapt fun charact still found hope
get renew allow see full stori properli want see iroh ’ zuko ’ stori play',
 'think show lot thing well like special effect think visual stun like costum also realli like cast im even mad lot cha
ng made think lot made sens stori wasnt expect 11 remak think show fall flat charact develop dynam charact dont get fee
l writer realli understood charact instead easilyexcit kid natur talent even wise occasion doesnt want respons aang sup
er seriou broodi guy interact other deal problem doesnt add doesnt seem genuin especi play kid og aang love seem like h
e put show actual fun im gon na go detail everi charact convers feel flat overli heavi mean dont natur evolv stori feel
like instead recogn origin trio meet three new charact interest consid mayb felt chang charact reflect brutal come live
act show lead charact see dont think good job im quit disappoint unfortun also dont believ turn around come season',
 'wy chang someth work main point stori readi wouldnt rewrit someth realli good seri import peopl stori arc chang avata
r favorit anim stori realli tri like live action didnt']
```

Sentiment Analysis & Word Clouds

```
In [21]: class sentiment_and_clouds:

             def __init__(self, topic):
                 self.topic = topic
                 self.pipeline = joblib.load('tuned_logreg_model_imdb.pkl')

             def adjectives(self):
                 '''Function to get adjectives'''
                 all_adjectives = []
                 for sentence in self.topic:
                     tokens = nltk.word_tokenize(sentence)
                     tagged = nltk.pos_tag(tokens)
                     part_of_speech = [word for word, pos in tagged if pos in ['JJ', 'JJR', 'JJS']]
                     all_adjectives.extend(part_of_speech)
                 return all_adjectives

             def make_cloud(self, adjectives):
                 '''Function to make word clouds'''
                 text = ' '.join(adjectives)
                 wordcloud = WordCloud(width = 400, height = 400,
                                       background_color ='white',
                                       stopwords = None,
                                       min_font_size = 4).generate(text)
                 plt.figure(figsize = (8, 8), facecolor = None)
                 plt.imshow(wordcloud)
                 plt.axis("off")
                 plt.tight_layout(pad = 0)
```

```
    def predict_sentiment(self):
        '''Function to determine if sentiment is negative(0) or positive(1)'''
        predictions = self.pipeline.predict(self.topic)
        return predictions

    def plot_scores(self):
        '''Function to plot sentiment scores'''
        scores = self.predict_sentiment()
        all_count = len(scores)
        negative_percent = scores.tolist().count(0) / all_count
        positive_percent = scores.tolist().count(1) / all_count

        plt.figure(figsize=(10,6))
        plt.text(0, negative_percent + 0.05, round(negative_percent, 2), fontsize=15, color='black', ha='center')
        plt.text(1, positive_percent + 0.05, round(positive_percent, 2), fontsize=15, color='black', ha='center')

        plt.bar(['Negative', 'Positive'], [negative_percent, positive_percent], color=['tab:red', 'tab:green'])
        plt.ylim(0, 1)
        plt.ylabel('Proportion')
```

In [22]:

```
story_adjectives      = sentiment_and_clouds(story).adjectives()
characters_adjectives = sentiment_and_clouds(characters).adjectives()
acting_adjectives     = sentiment_and_clouds(acting).adjectives()
cgi_adjectives        = sentiment_and_clouds(cgi).adjectives()
dialogue_adjectives   = sentiment_and_clouds(dialogue).adjectives()
```

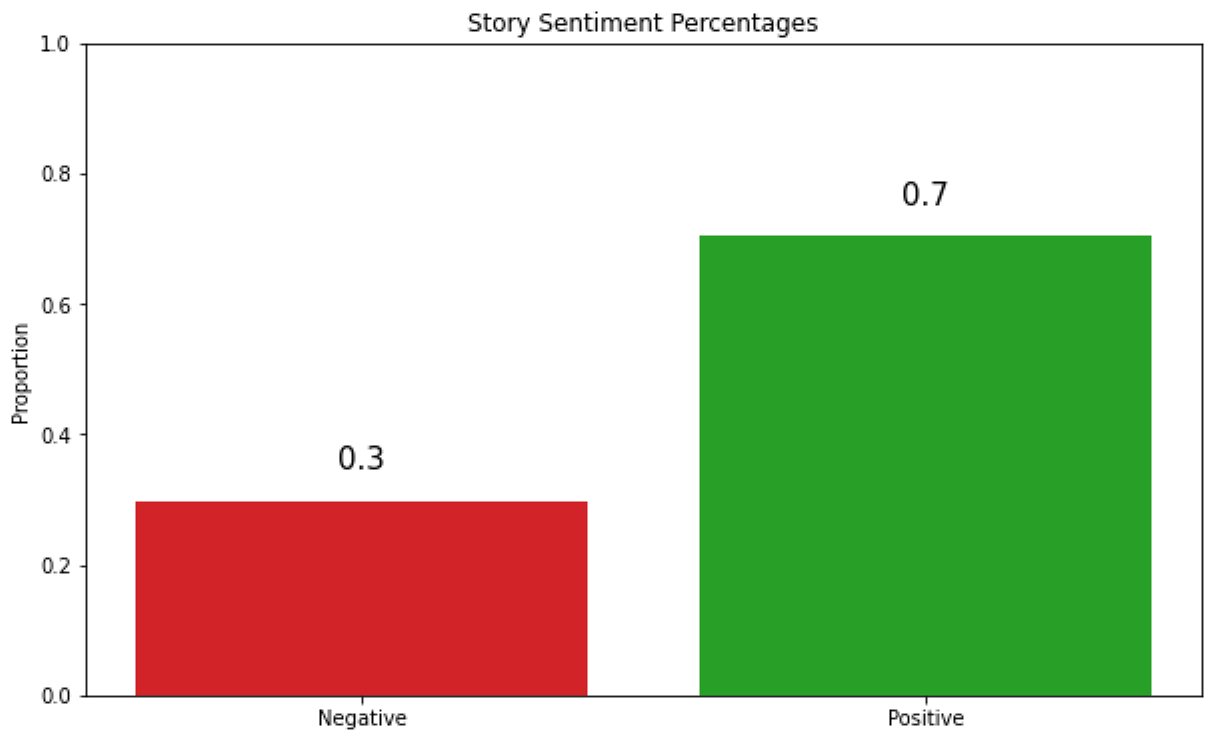
> The predict_sentiment function uses the pipeline from the IMDB_Logistic_Regression file.

Results

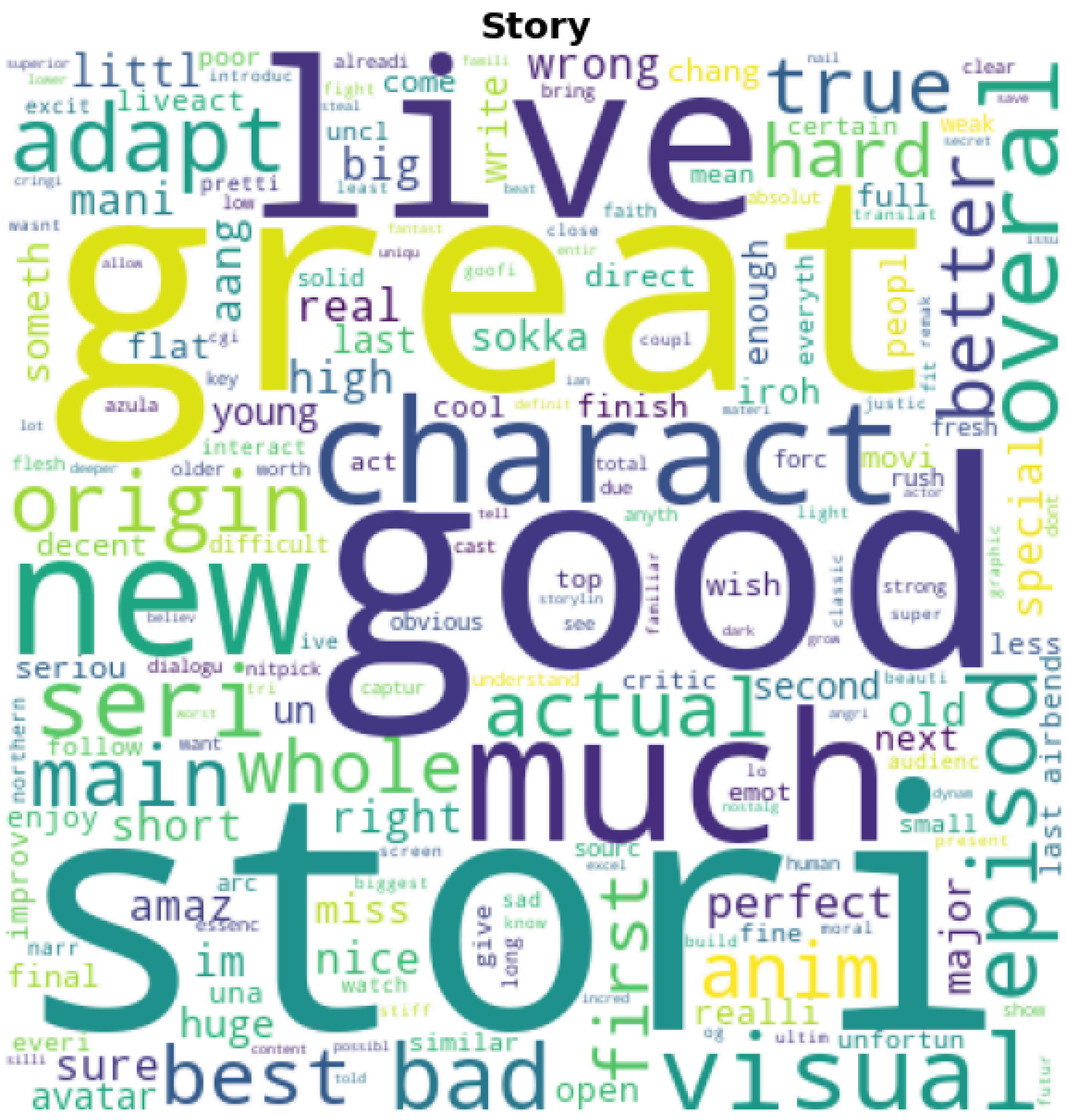
Story

In [23]:

```
sentiment_and_clouds(story).plot_scores()
plt.title('Story Sentiment Percentages')
plt.show()
```

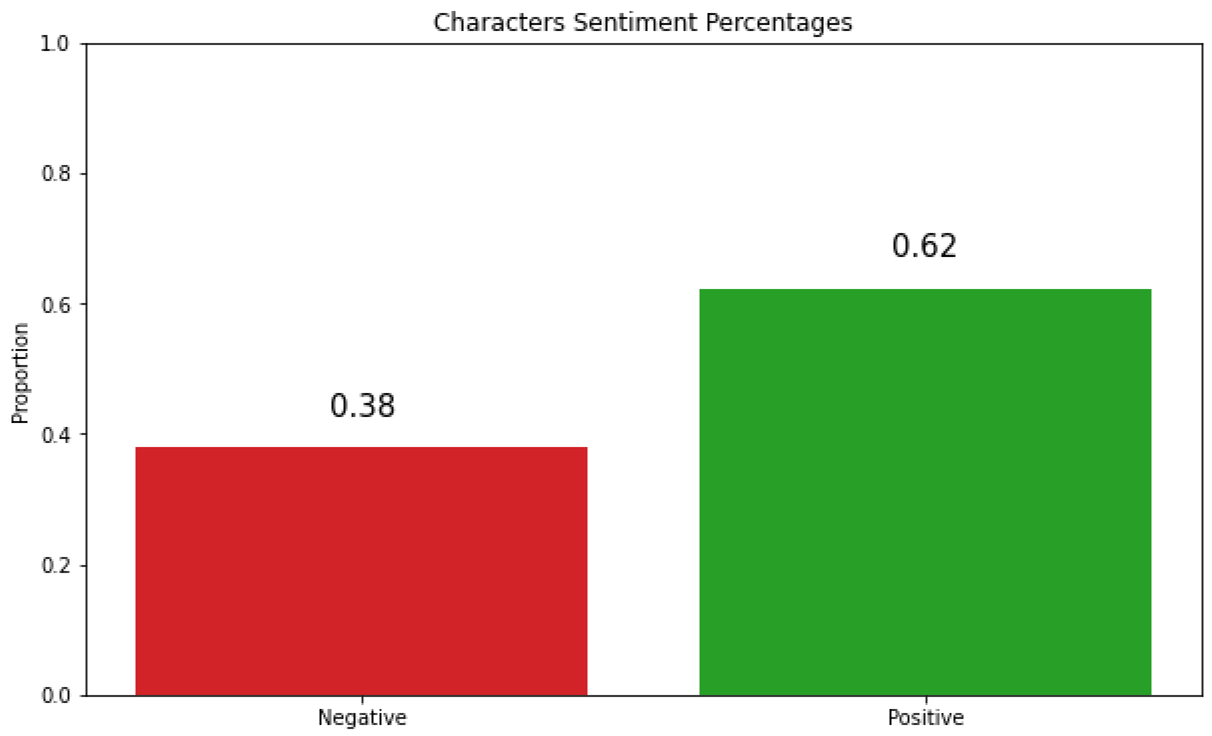


```
In [25]: sentiment_and_clouds(story).make_cloud(story_adjectives)
         plt.title('Story', fontsize=20, fontweight='bold')
         plt.show()
```

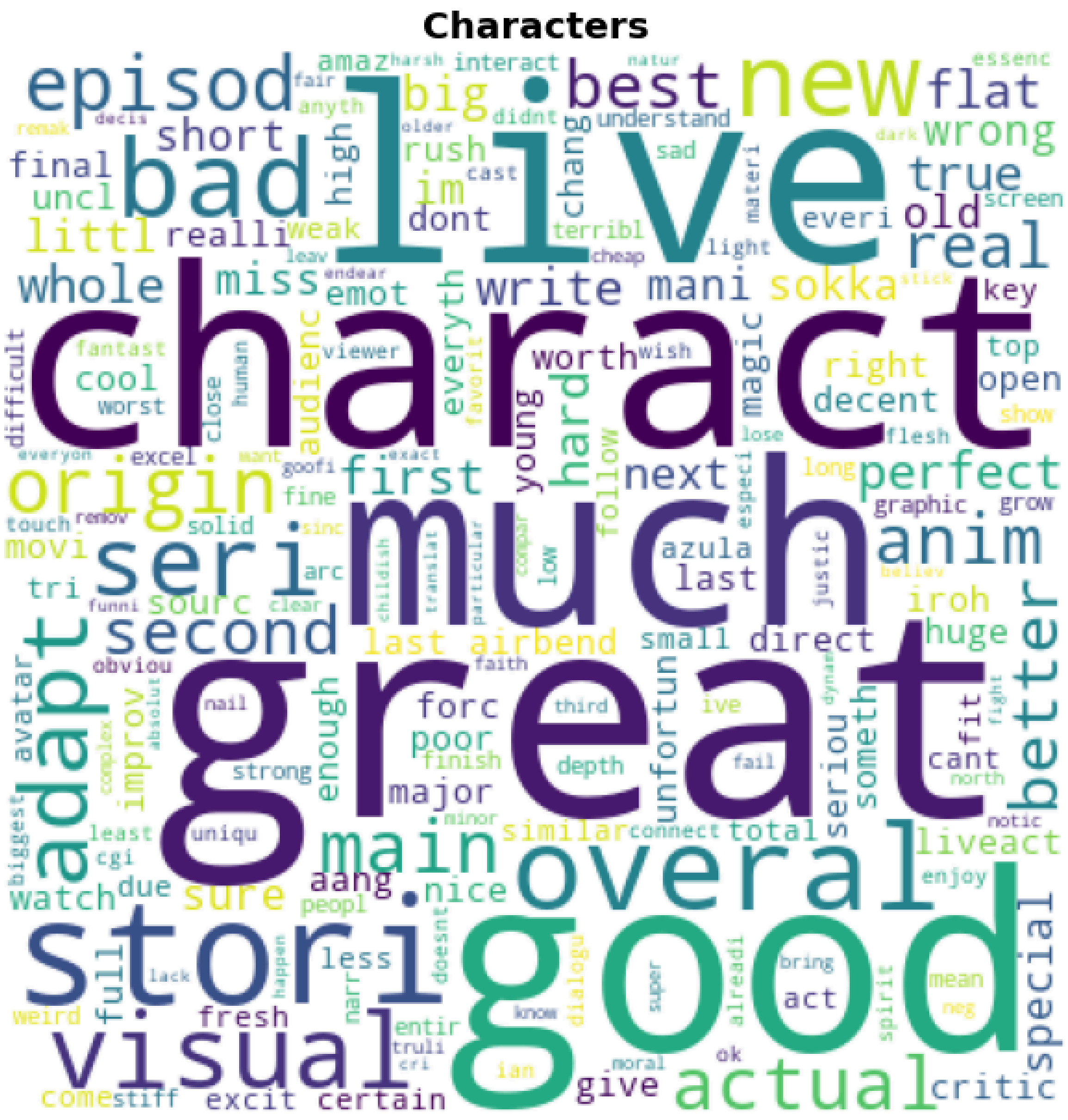


Characters

```
In [26]: sentiment_and_clouds(characters).plot_scores()
         plt.title('Characters Sentiment Percentages')
         plt.show()
```

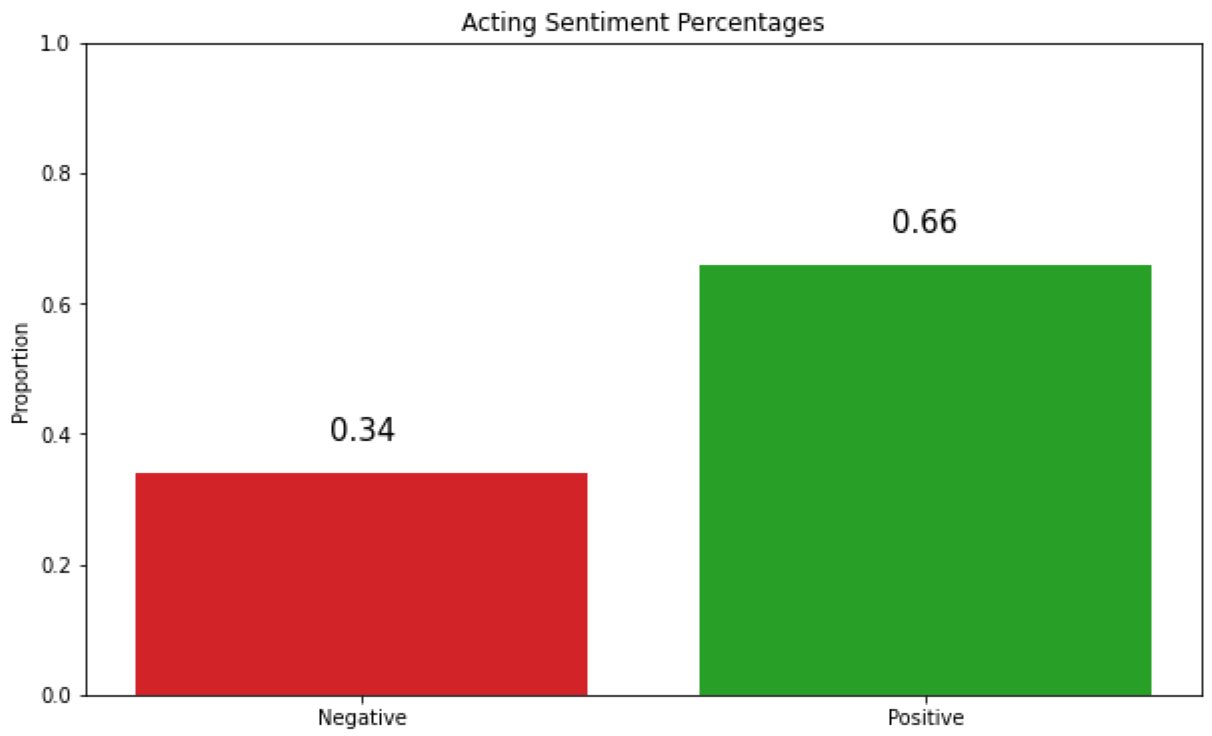


In [27]:
```
sentiment_and_clouds(characters).make_cloud(characters_adjectives)
plt.title('Characters', fontsize=20, fontweight='bold')
plt.show()
```

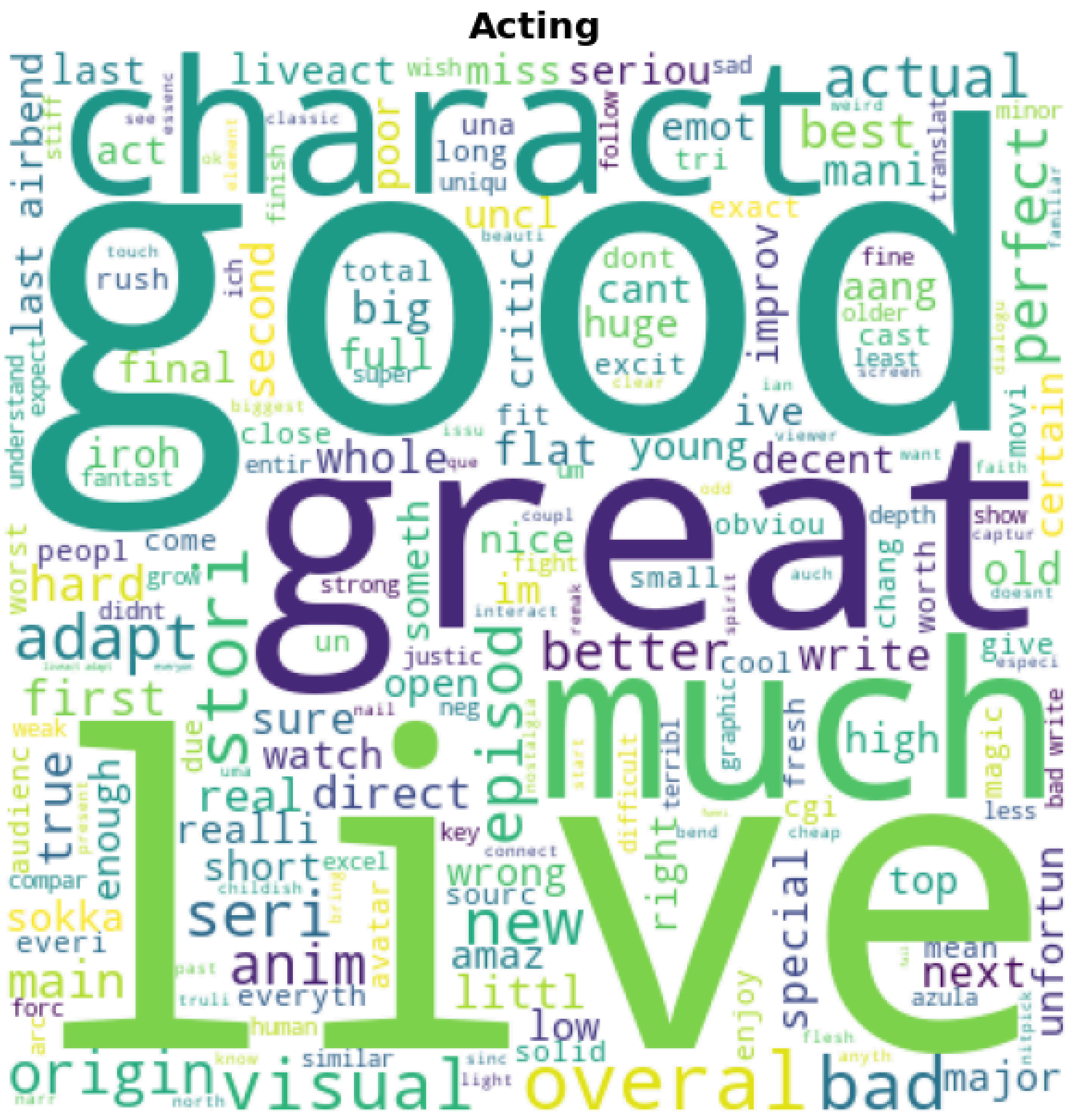


Acting

```
In [28]: sentiment_and_clouds(acting).plot_scores()
         plt.title('Acting Sentiment Percentages')
         plt.show()
```

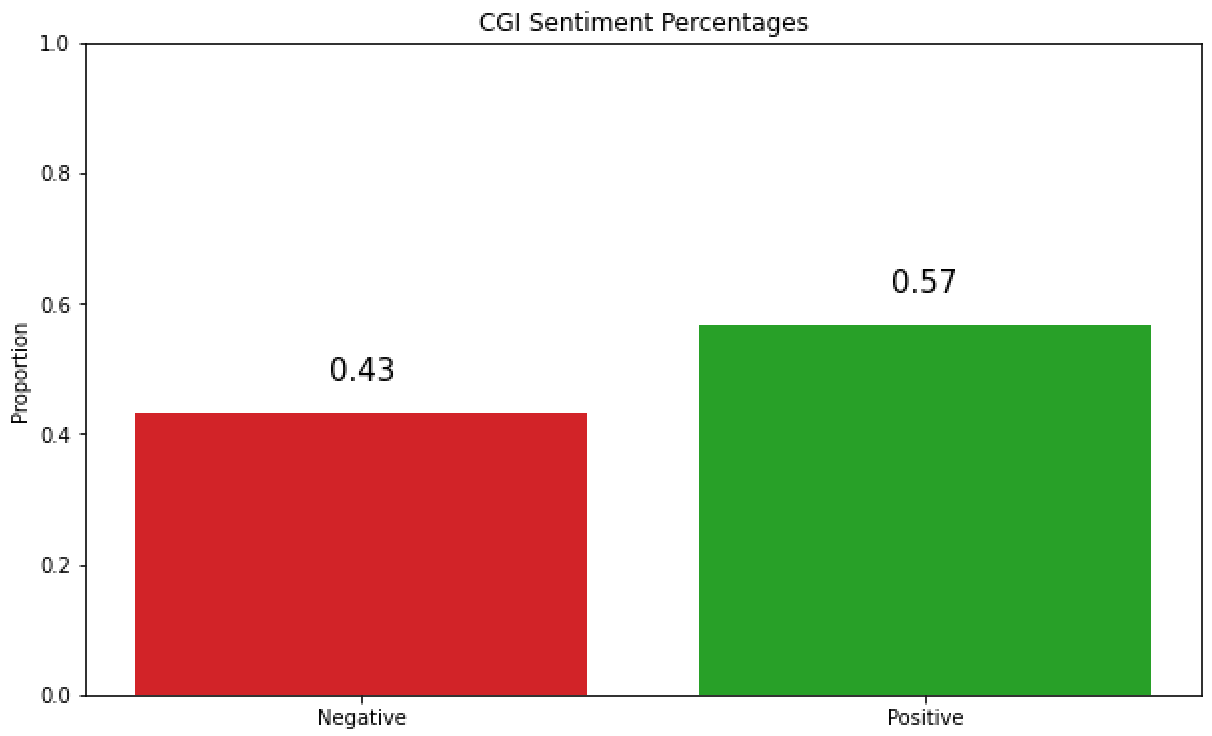


In [29]:
```
sentiment_and_clouds(acting).make_cloud(acting_adjectives)
plt.title('Acting', fontsize=20, fontweight='bold')
plt.show()
```



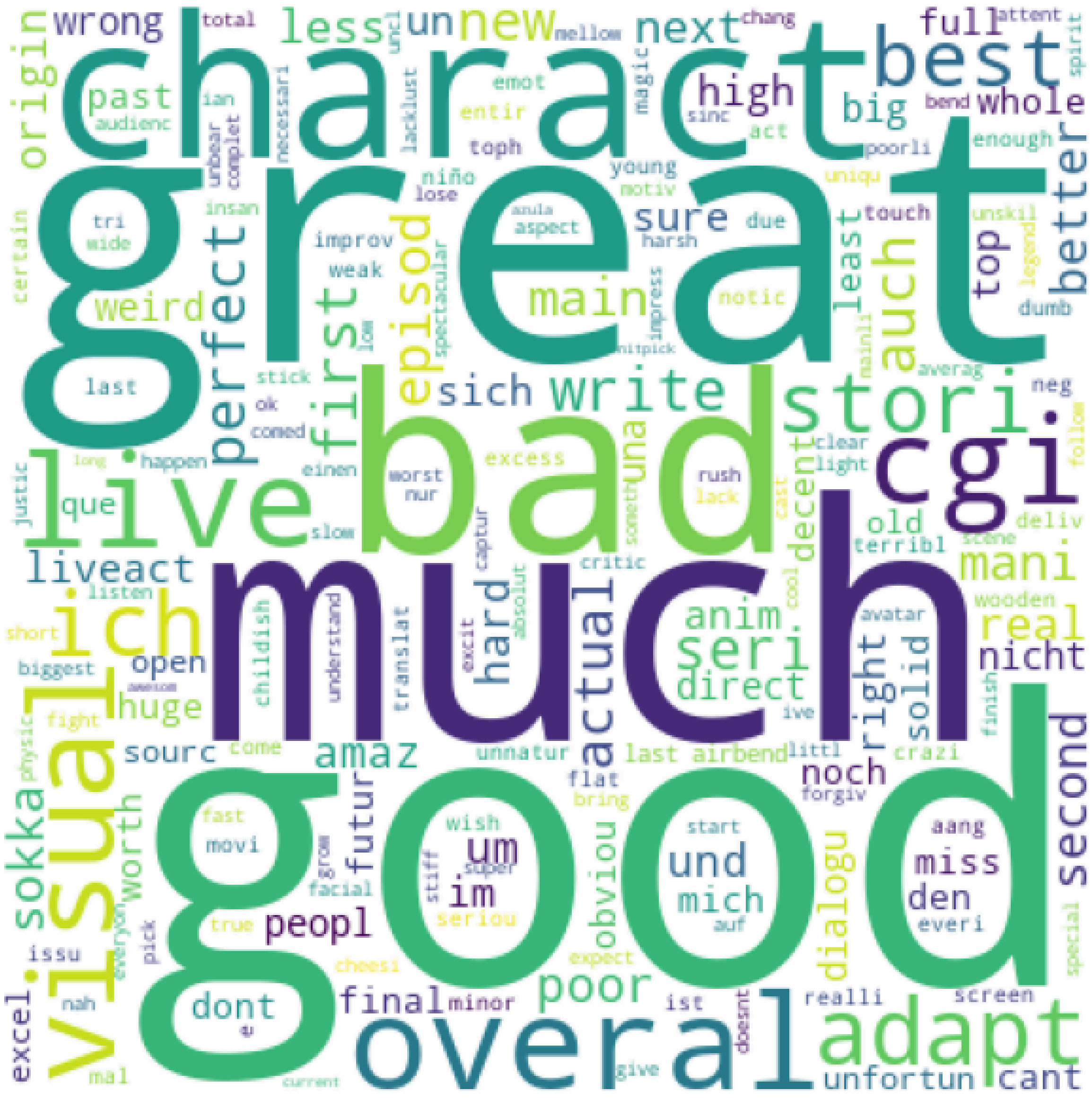
CGI

```
In [30]: sentiment_and_clouds(cgi).plot_scores()
         plt.title('CGI Sentiment Percentages')
         plt.show()
```



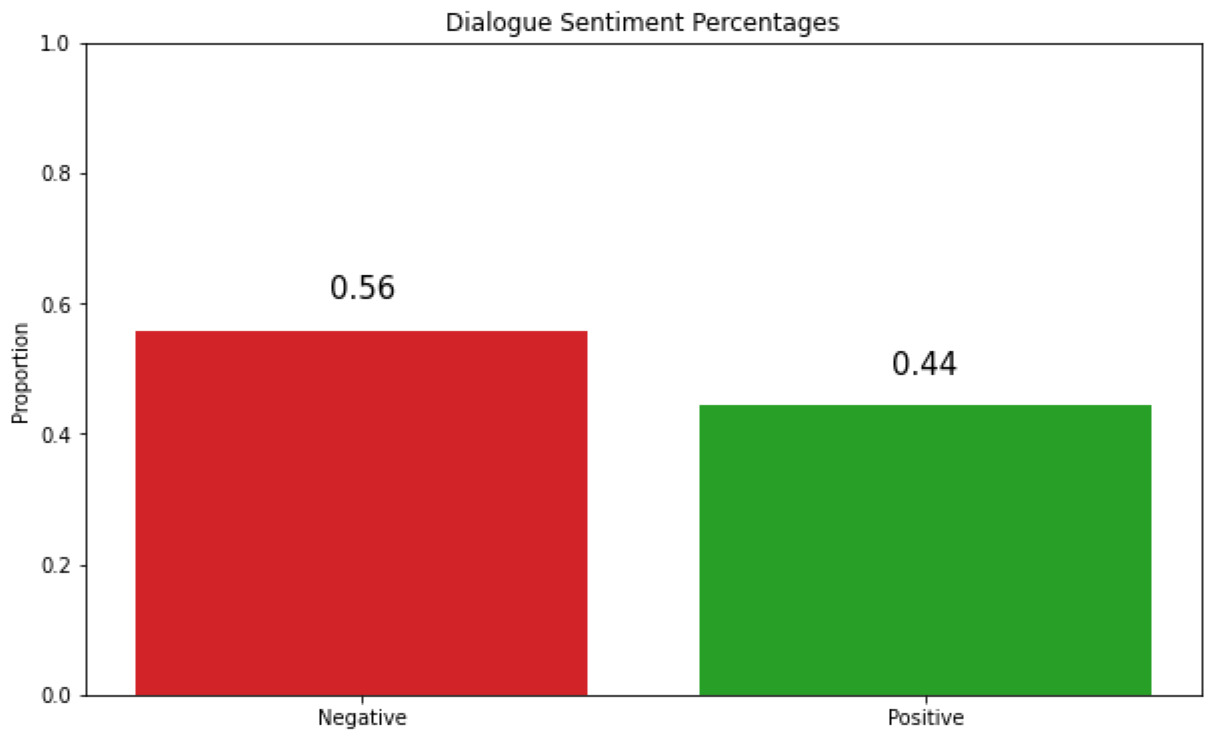
```
In [31]: sentiment_and_clouds(cgi).make_cloud(cgi_adjectives)
         plt.title('CGI', fontsize=20, fontweight='bold')
         plt.show()
```

CGI



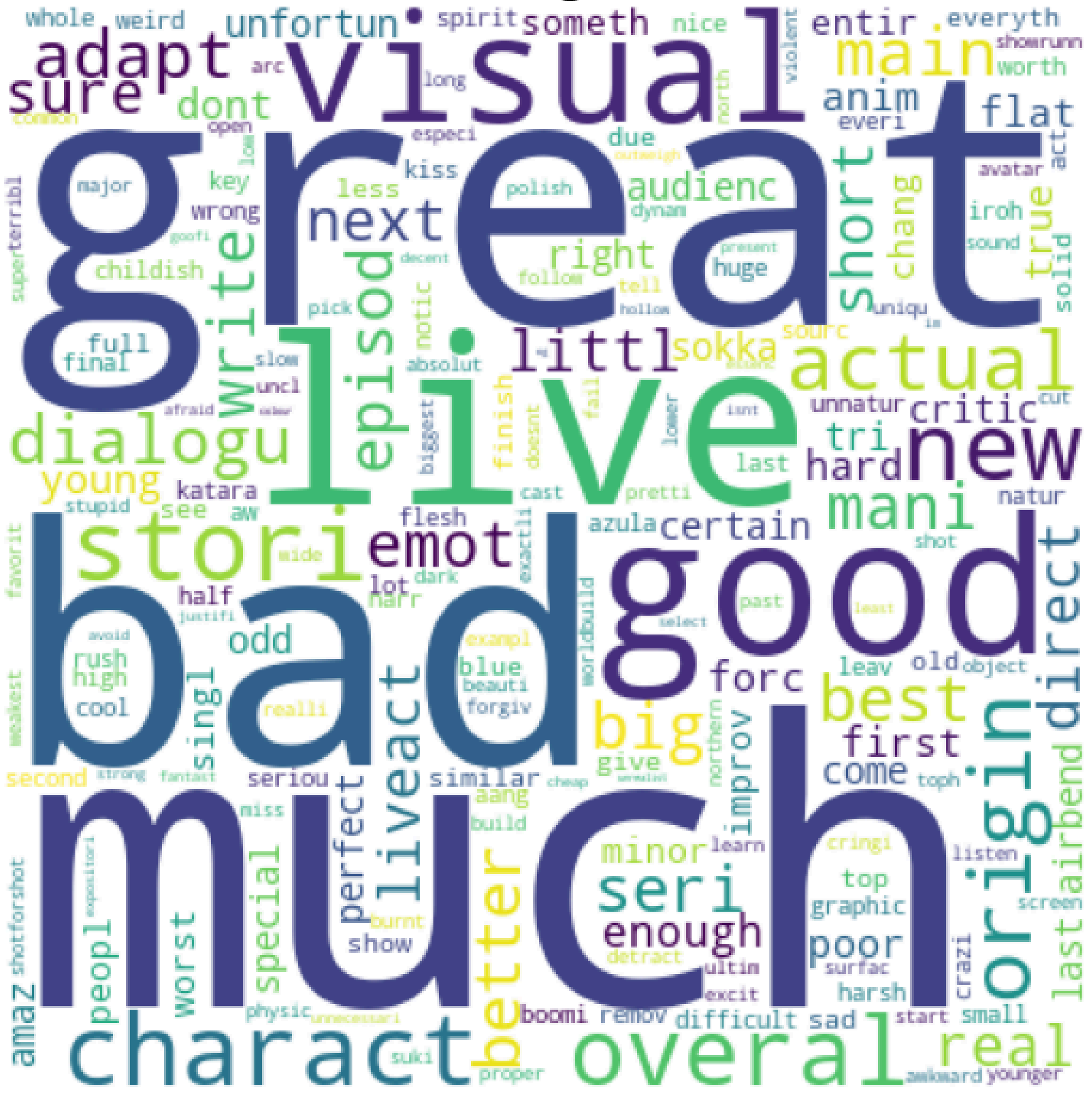
Dialogue

```
In [32]: sentiment_and_clouds(dialogue).plot_scores()
         plt.title('Dialogue Sentiment Percentages')
         plt.show()
```



```
In [33]: sentiment_and_clouds(dialogue).make_cloud(dialogue_adjectives)
         plt.title('Dialogue', fontsize=20, fontweight='bold')
         plt.show()
```

Dialogue



Conclusion

- Uninformative adjectives like 'animated', 'much, 'original' and so on will be ignored in the following conclusion.

Story: The audience seems happy with the story. The sentiment scores are negative:0.3, positive:0.70. The three biggest adjectives are 'great', 'good', 'new'

Characters: The audience seems somewhat happy with the characters. The sentiment scores are negative:0.38, positive:0.62. The three biggest adjectives are 'great', 'good', 'bad'

Acting:The audience seems somewhat happy with the acting. The sentiment scores are negative:0.34, positive:0.66. The three biggest adjectives are 'good', 'great', 'bad'

CGI:The audience seems somewhat unhappy with the CGI. The sentiment scores are negative:0.43, positive:0.57. The three biggest adjectives are 'good', 'great', 'bad'.

Dialogue: The audience seems unhappy with the dialogue. The sentiment scores are negative:0.56, positive:0.44. The three biggest adjectives are 'great', 'bad', 'good'.

> In colclusion, the word clouds of adjectives were only somewhat helpful, but from the sentiment scores, it seems that what could improve the show the most in the opinion of the audience is better dialogue and to a lesser degree, better CGI.