Text Mining South Park

Dominik Nerger (i6146759)

June 10, 2017

Contents

1 Introduction

> And tell everyone in the past for us, that no one single answer... is ever the answer.
>
> UAA Leader, Ep. Go God Go XII

South Park is an animated series revolving around the four boys Stan Marsh, Kyle Broflovski, Eric Cartman and Kenny McCormick who live in South Park, Colorado. South Park has the third most episodes in regard to American animated TV series, totaling 277 episodes over 20 seasons.

Because of the huge amount of episodes, it is an interesting target for the practical assignment, which is part of the course *Information Retrieval & Text Mining*, with the possibility of acquiring new insights into a TV series that is used to satirize American culture.

At first, all libraries that were used will be presented. Afterwards, the data set and the used corpora are introduced. Then the pre-processing tasks that are performed on the data set will be explained. Next, the results and visualization will be discussed while explaining why and how everything has been implemented. In the end, a conclusion will be drawn and possible improvements for the future will be discussed.

The repository is available on GitHub[1].

2 Libraries

All scripts have been programmed in *R*. To execute the scripts, R needs to be installed. To view temporary files that are executed during runtime, e.g. the corpus or a TermDocumentMatrix, it is advised to install RStudio. The libraries necessary for each script are imported at the top of each script, if they are not installed they can be installed by executing:

```
install.packages("library-name")
```

In the following, all libraries that are related to Text Mining techniques will be introduced shortly.

The library **tm** is the Text Mining package of R, which enables pre-processing of data sets and allows to build the corpus. **RWeka** is a collection of machine learning algorithms for data mining tasks. **NMF** introduces the Non-negative Matrix Factorization to R. **NLP** and **OpenNLP** are libraries that provide Natural Language Processing techniques and are used for NER-Tagging. **syuzhet** uses tokenized input to extract sentiments and emotions and contains the three dictionaries *bing*, *afinn* and *nrc*. The package **stm** is used for Structural Topic Modeling which is LDA with additional met-data and can be visualized using the package **LDAvis**. Libraries used for visualization include **igraph**,**ggplot2**, **ggraph**, **viridis** and **wordcloud**.

[1] `https://github.com/dnerger/South-Park-Text-Mining`

3 Data set

The data set spans from seasons **1** to **18**, adding up to 257 episodes overall with a file size of 5.41MB. It contains 70896 rows, with each row possessing information about the **season**, **episode**, **character** and **line** and is available as a *.csv* file.

The data set has been crawled by Bob Adams and is available to download on GitHub[2]. It has been assembled by crawling the South Park Archives[3]. The code of this GitHub repository is not available to the public.

I made an attempt at a crawl with the R package **rvest**, which is able to harvest data from websites. However, the *html data* of the episode scripts provided on the South Park Archives differ between episodes. For 16 of 20 seasons, the whole episode needs to be pre-processed manually because the **line** and *character* are extracted individually and can not be merged because of random whitespace lines that are introduced by **rvest**.

In the scripts, the data set is used in three different variations:

1. *dialogue* - each entry holds information about season, episode, character and line
2. *by.season* - each entry holds information about season and the complete script for that season
3. *by.episode* - each entry holds information about that episode, season and script for that episode

Both *by.episode* and *by.season* are created by using the data from *dialogue*.

4 Pre-processing

In general, the standard pre-processing tasks are executed on the corpus. Depending on the later use, different packages perform these. The package **tm** is used to convert the corpus to lower case, remove punctuation and numbers as well as strip whitespaces. Furthermore, stopwords are removed using the stopword list from the Snowball stemmer project. The corpus is stemmed using Porter's stemming algorithm.[4] It is used together with several DocumentTerm- and TermDocument-Matrices. From these DTMs/TDMs, sparse terms are removed. The maximum allowed sparsity depends on the data set and the intended use, with the allowed percentage being higher for the *by.episod* data set in comparison to *by.season* because there are more documents in the *by.episode* data set.

The pre-processing for the Structural Topic Model is done with the same tasks, however those are performed by the package **stm** itself.

[2] `https://github.com/BobAdamsEE/SouthParkData`
[3] `http://southpark.wikia.com/wiki/South_Park_Archives`
[4] `http://snowball.tartarus.org/algorithms/english/stop.txt`

For the Named Entity Recognition with package **openNLP**, no pre-processing tasks are performed. They can be executed and might improve the overall runtime, but it does not change the overall result. Especially stemming and the lower case convertion could decrease the results because stemmed or lower case entities, as well as the sentiments that are mined, will not be recognized.

5 Results and visualization

To sufficiently present the results and the corresponding visualization, each relevant script will present them in their own subsection.

5.1 Wordclouds

To start to gain an insight into South Park, I started with building wordclouds that display the most frequent words to see what South Park is about. However, even with stopwords removed, the words that are most frequent mostly include words such as *get, just* and *now*. From these words, nothing about the TV show can be derived.

To get meaningful answers to what is important in and characteristic for South Park, the corpus is built with a term frequency-inverse document frequency (tf-idf) weight, which reflects how important each term is to the corresponding document in the corpus.

Figure 1: Terms with highest TF-IDF score - *by.episode*

Figure 2: Terms with highest TF-IDF score - *by.season*

With this weighting, the terms that are most important over all 18 seasons can be derived, as can be seen in Figures 1 and 2.

The difference between the wordclouds descends from the different *idf* weights for the data sets *by.season* and *by.episode*. The words in Figure 1 show the

names of secondary characters such as Butters, Kenny and Wendy. These characters are not featured in Figure 2 since they do appear in every season, but not in every episode.

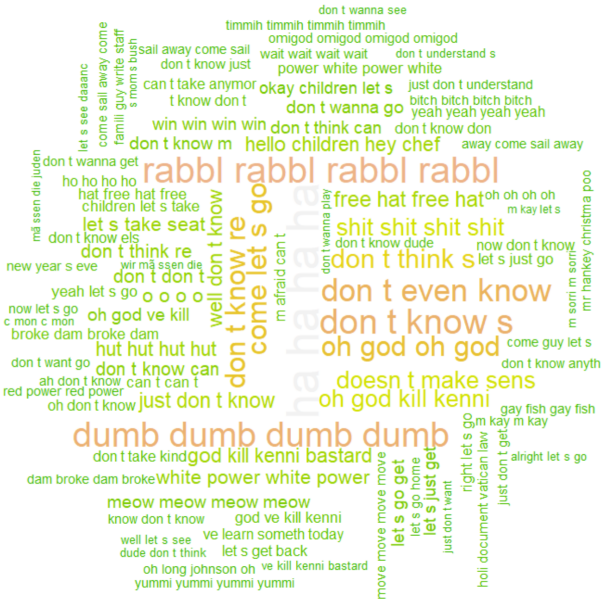

Figure 3: Wordcloud containing ngrams (n=4) by frequency

Figure 3 shows the most frequent quadgram phrases without stopwords included. This includes sentences like the trademark phrase "Oh my god, they killed Kenny", which is shown as "oh god kill kenni".

5.2 Co-occurences and dendrogram

To gain insight into the relations between characters and to confirm the major characters of South Park, which are Stan, Kyle and Cartman, I decided to visualize the 20 characters with the most lines.

As can be seen in Figure 4, Cartman is the character with most lines overall, with Stan and Kyle, as well as Butters and Randy being other major characters. This has been implemented by aggregating all lines for a character and then calculating the percentages in respect to the overall lines.

However, this does not show the relations between the characters. To visualize this, I decided to create a co-occurence network by creating a DocumentTermMatrix that contains the characters as terms and the unique episode number as document. The value in each entry is the amount of lines for the character in the specific episode. By using the formula $C = A^t A$ with A as the DocumentTermMatrix, I created a TermTermMatrix that contains the co-occurences between lines of characters, resulting in the co-occurences of the 20 most relevant characters in South Park.

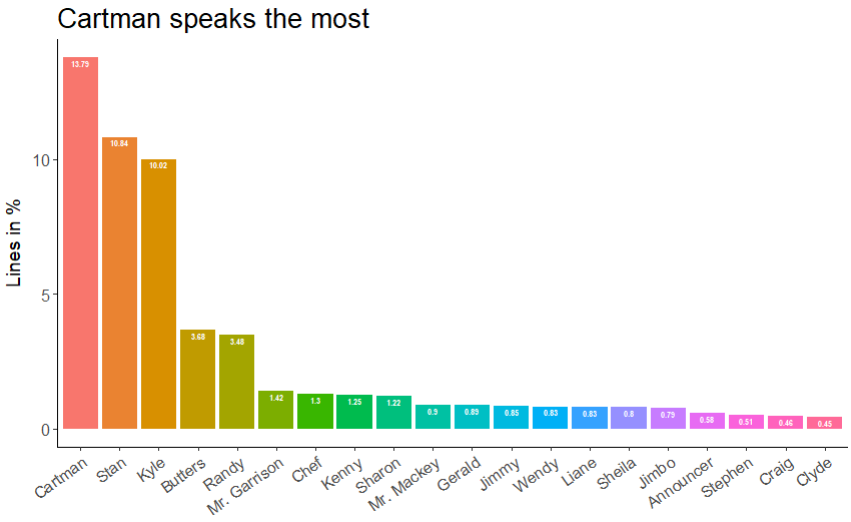

Figure 4: Speaking percentage for 20 major characters

As can be seen in Figure 5, all 20 major characters interact in some way throughout all 257 episodes, with Cartman, Kyle and Stan being characters that are closely tied together.

Since it is hard to derive the relations between the characters from Figure 5, I decided to use a cluster dendrogram to visualize which characters have the most episodes in common.

Figure 6 shows the dendogram clustering the characters. The further down the node of two characters splits, the more episodes these characters have in common. This shows once again that the major characters Cartman, Stan and Kyle are closely tied together. Furthermore, interesting connections between the students Clyde and Craig as well as

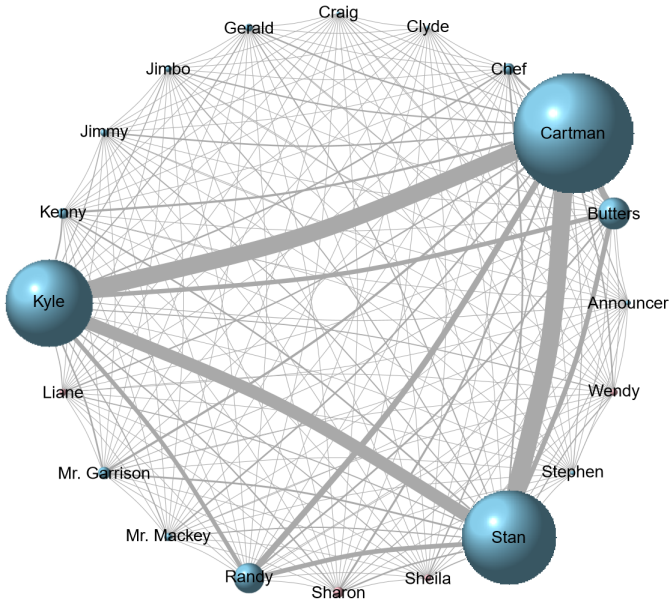

Figure 5: Co-occurence network

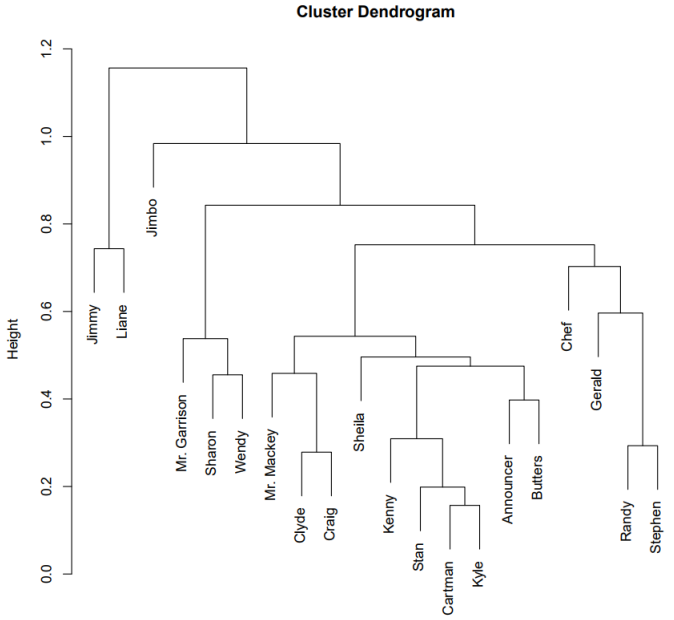

Figure 6: Cluster dendrogram of 20 major South Park characters

5.3 Named Entity Recognition

To extract information about characteristic information such as organizations, persons and locations that are mentioned in South Park, I decided to use the Named Entity Recognition provided by the Apache **openNLP** library.

5.4 Sentiment and Emotion Mining

Another approach to learn something about South Park is to extract the sentiments and emotions that are expressed by the characters. The sentiments are annotated onto the seasons, using ...

After the complete corpus has been annotated, which results in little chunks of words being tied together, the sentiments are processed with the library **syuzhet** and the sentiment dictionary *nrc*. During the processing the word chunks are grouped with 100 chunks being tied together. Under the assumption that each character line contains at least 1 chunk, this results in a minimum of 700 grouped chunks for 70000 character lines, which can be seen as roughly three different sentiment values for each episode. This can be visualized with a sentiment timeline, as can be seen in Figures 9 and 10. One interesting observation is that the sentiments in Figure 10 for season 18 are mostly positive with only a few segments having a negative sentiment score.

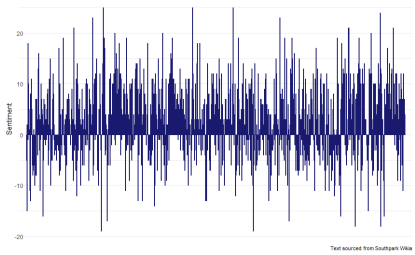

Figure 7: Sentiment timeline over 18 seasons

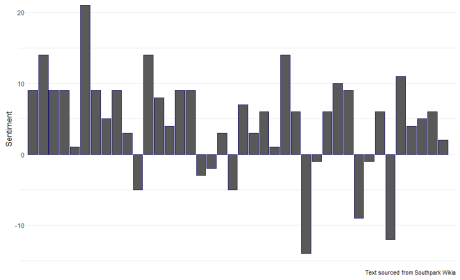

Figure 8: Sentiment timeline for season 18

However, it is hard to derive a meaning from the sentiments that are shown in Figure 9. Therefore, I decided to filter and transform the sentiment timelines with a low-pass Fourier transform using a low-pass size of 3. The Fourier Transform puts the data into a periodic shape.

5.5 Non-negative Matrix Factorization

5.6 Structural Topic Model

To gain further insights into the topics that define South Park, I decided to use the **stm** package, which provides the Structural Topic Model. Basically,

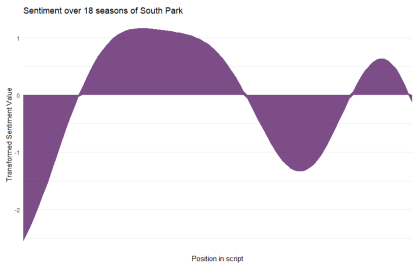

Figure 9: Fourier transformed sentiment timeline over 18 seasons

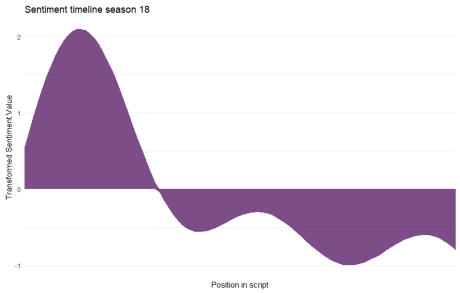

Figure 10: Fourier transformed sentiment timeline for season 18

stm is similar to other probabilistic topic models like the Latent Dirichlet allocation(LDA). However, it allows to incorporate metadata into the topic model and enables the user to discover the main topics in a text corpus with a connection to the metadata.

6 Conclusion

In conclusion,

7 Future work

One possible future implementation is to visualize the sentiment and emotion timelines for different characters, which would then combine the three W-questions *who, why* and *when*.