A drawing of a cartoon character

Description automatically generated

Jeffrey Levesque

jlevesqu@syr.edu | Compare MNB and SVMs for Sentiment Analysis

IST-736: HW7

PRofessor gates

Table of Contents

[Introduction 2](#_Toc9895814)

[Analysis 3](#_Toc9895815)

[Data Preparation 5](#_Toc9895816)

[Amazon Mechanical Turk 5](#_Toc9895817)

[Stop Words 6](#_Toc9895818)

[Results 6](#_Toc9895819)

[Unigram Models 6](#_Toc9895820)

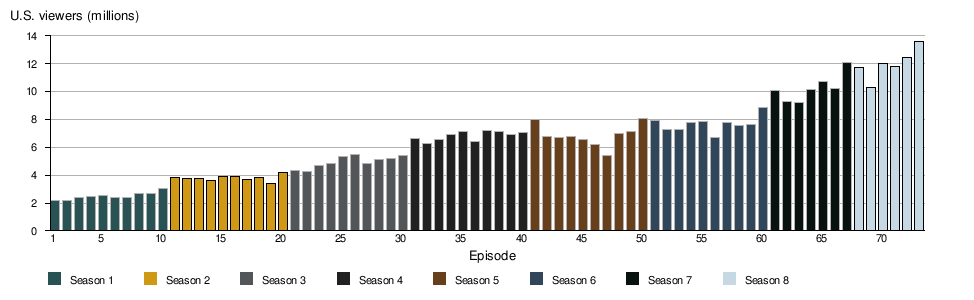
[Unigram and Bigram Models 8](#_Toc9895821)

[Unigram (100% train) 10](#_Toc9895822)

[Conclusions 11](#_Toc9895823)

# Introduction

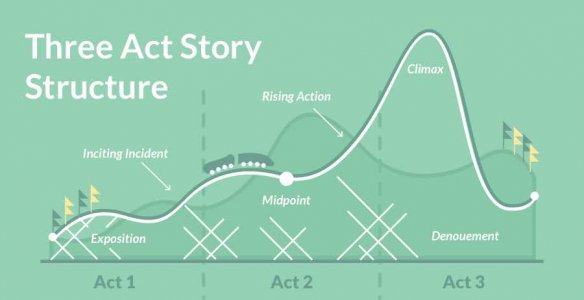
The Game of Thrones (GOT) is the most successful television series produced by HBO. Accounting for various factors, GOT has eclipsed previous celebrated shows including The Sopranos, and The Wire[[1]](#footnote-1). It has been approximated roughly 44.2 million viewers (including delayed viewers) participated watching the season finale[[2]](#footnote-2).



**Figure 1:** US live GOT historical viewership per episode[[3]](#footnote-3).

While viewership, and fanbase has clearly grown over the years, the glue that captivates audience is something well-orchestrated by respective producers. From the very first season, audience members followed multiple storylines premised on the struggles of a fictional continent containing the seven Kingdoms of Westeros. Subplots with mystical creatures including giants, dragons, and “walking dead” along with noble families vying for the “iron throne”, were many factors of the successful series. Each season cleverly interwoven multiple storylines, sometimes hinting to audience members a degree of foreshadowing, while other times requiring some level of imagination. This type of engagement between audience members, has allowed HBO producers David Benioff and D. B. Weiss to be exceedingly successful in the GOT series.

With the recent season 8 finale, audience members developed a great sense of curiosity regarding the many interwoven story layers. From the very first season, producers provided numerous major back stories leading to a perceived and anticipated season finale climax.



**Figure 2:** common three act structure used by screenwriters and novelists[[4]](#footnote-4)

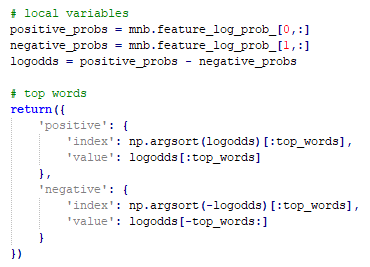
However, a stinted climax occurred during the shortened mid-season, followed by a progressive killing-off major characters through the remaining three episodes. No major back-story was expanded upon, and potentially eliminated by the death of the major characters. While the conclusion of such a large drama series is not expected to please the whole, many fans expected some degree of climax associated with the many subplots. Consequently, fans turned to petition a rewrite of season 8[[5]](#footnote-5). On the Sunday of the season finale, the petition received over 1.1M votes[[6]](#footnote-6). Additionally, fans have expressed themselves in interesting ways including flying a banner over Seattle, WA to rewrite season 8[[7]](#footnote-7). Others have proposed an alternate finale by writing their own suggestive plot[[8]](#footnote-8). Furthermore, some have conducted exploratory sentiment analysis for associated GOT tweets, within 24 hours of the finale[[9]](#footnote-9). Surprisingly, general sentiment was slightly more positive (0.1612) than negative (0.1307).

# Analysis

While others have conducted exploratory analysis on GOT sentiment using vader[[10]](#footnote-10), this study focuses mainly on the ability to classify negative and positive sentiment associated with GOT. However, the train distribution used for each classifier does provide an indication regarding the distribution of twitter sentiment, which is similar in nature to the exploratory exercise.

The Multinomial Naïve Bayes (MNB), and Support Vector Machines (SVM) were used to classify tweets into a series of sentiment categories. This was conducted using the HBO series Game of Thrones (GOT), where tweets were trained against a unigram representation. Kfold validation, and confusion matrices were computed to assess model performance. Furthermore, precision, recall, and the fbeta measures[[11]](#footnote-11) were used. Specifically, precision was used to measure the ratio of correctly predicted positive labels against the entire positive labels (precision = tp/(tp+fp)). Generally, high precision is related to low false positives[[12]](#footnote-12). Similarly, recall was calculated to measure the ratio of correctly positive labels against the entire labels for the given class (recall = tp/(tp+fn)). Finally, the fbeta score combined the former scores to produce a harmonic mean (fbeta = 2(precision\*recall)/(precision+recall))[[13]](#footnote-13). In general, these measures provide a more granular measure of the confusion matrix.

Additionally, the top 25 most indicative words for positive and negative categories were determined by frequency, as well as by values corresponding to term frequency-inverse document frequency (TFIDF)[[14]](#footnote-14). Furthermore, the MNB provides log probabilities[[15]](#footnote-15) to determine positive and negative words associated with corresponding sentiment. Using a custom adaptation, the ‘*value’* calculation provides a somewhat arbitrary measure:



**Figure 0:** preprocessing to list most positive and negative influential MNB words.

Two additional cases were conducted in a similar fashion. However, rather than implementing a unigram representation an ngram\_range=(1,2)[[16]](#footnote-16) was utilized. This allowed models to train using a vectorized combination of both unigrams and bigrams. Moreover, using knowledge of the first two cases, a final model was constructed using the entire train dataset. This eliminated the need for a test condition, with the assumption that the first two cases satisfied initial data exploration. Additionally, since tweets are relatively, with no more than 280 characters[[17]](#footnote-17), the unigram implementation was most suitable.

# Data Preparation

Data was acquired using the standard twitter search api[[18]](#footnote-18) via the twython python package[[19]](#footnote-19). Specifically, three different hash tags were aggregated using the api:

* GameofThrone
* GameofThronesFinal
* GOTFinale

The dates of the collected tweets were approximately one week after season finale of the Game of Thrones (GOT). While most of the data hovered exactly at one week after, tweets from the GOTFinale ranged from 5-7 days after the finale. This was due to less activity than the former two hashtags.

Moreover, each hashtag was written to a corresponding csv file[[20]](#footnote-20). However, to improve performance on successive analysis, an intermediate sample.csv[[21]](#footnote-21) was created. Specifically, for each of chosen hashtags, 500 tweets were sampled for a total of 1500 rows:

df\_sample = [data[x].sample(500) for x in [\*data]]

## Amazon Mechanical Turk

This reduced the space from a combined 3670 rows, then was passed to Amazon Mechanical Turk (MTurk)[[22]](#footnote-22). More generally, using the reduced csv, five workers were selected at $0.01 per task to categorize tweets into the following categories:

* Very Negative
* Negative
* Neutral
* Positive
* Very Positive

Obtaining and approving results was relatively fast, under 2.5 hours after batch submission. Since five workers conducted the same task for each tweet, tweets with the highest category mode was selected. In the case where the highest mode was a tie, the first highest value was selected. Furthermore, the sample dataset was unbalanced. Specifically, Neutral contained significantly higher than all categories. Therefore, this dimension was removed. Furthermore, both Very Negative, and Very Positive were each less than 10% of the counterparts. Therefore, these dimensions were combined using the following pattern:

df.sentiment.replace(

to\_replace='very negative',

value='negative',

inplace=True

)

## Stop Words

A rich set of stop words was created, and used to remove words prior to vectorization:

* http
* https
* nhttps
* rt
* RT
* amp
* co
* gameofthrones
* gameofthronesfinale
* gotfinale
* season
* seasons
* got8
* got
* gt
* think
* twitter
* thefinalepisode
* ye
* el
* la
* lo
* en
* es
* est
* deo
* se
* s8
* que
* para
* guion
* hecho
* trono
* se

# Results

Once all preprocessing was completed, the full\_text (X) containing tweet text, was trained against the sentiment column (y) containing the positive vs. negative sentiment.

## Unigram Models

The BNB showed the greatest accuracy, while having difficulty predicting negative sentiment. On the other hand, the MNB had an accuracy almost similar, while having a slightly more balanced result. Similarly, the SVM performed almost equal to the MNB.

|  |  |  |
| --- | --- | --- |
|  |  |  |
| **Figure 3:** BNB confusion matrix on twitter sentiment. | **Figure 4:** MNB confusion matrix on twitter sentiment. | **Figure 5:** SVM confusion matrix on twitter sentiment. |

The kfold validation replicates results somewhat like the above confusion matrices. Specifically, the MNB produces the least variability nearing 0.7. While BNB and SVM produces similar high score, both contain the higher variability.

|  |  |  |
| --- | --- | --- |
|  |  |  |
| **Figure 6:** BNB kfold validation on twitter sentiment. | **Figure 7:** MNB kfold validation on twitter sentiment. | **Figure 8:** SVM kfold validation on twitter sentiment. |

|  |  |  |
| --- | --- | --- |
|  |  |  |
| **Figure 9:** BNB precision, recall, f-beta on twitter sentiment. | **Figure 10:** MNB precision, recall, f-beta on twitter sentiment. | **Figure 11:** SVM precision, recall, f-beta on twitter sentiment. |

The top 25 positive and negative words follow somewhat expected results. Specifically, negative words include wedeservebetter, nonsense, purenonsense, as well as derogatory words. On the other hand, positive words include best, bold, love, behindthescene.

|  |  |  |
| --- | --- | --- |
|  |  |  |
| **Figure 12:** top 25 tfdif words on twitter sentiment. | **Figure 13:** top 25 MNB positive words on twitter sentiment. | **Figure 14:** top 25 MNB negative words on twitter sentiment. |

Train data used for various models was relatively balanced, with a slight favor towards positive sentiment. Performing a long run sampling technique could indicate the overall distribution for positive versus negative sentiment.

|  |  |  |
| --- | --- | --- |
|  |  |  |
| **Figure 15:** BNB train distribution on twitter sentiment. | **Figure 16:** MNB train distribution on twitter sentiment. | **Figure 17:** SVM train distribution on twitter sentiment. |

## Unigram and Bigram Models

The SVM showed the greatest accuracy, while containing the most balanced results. The MNB followed by BNB contained less accuracy and balanced results, respectively. Like the pure unigram approach, the combined unigram and bigram models still had difficulty predicting negative sentiment.

|  |  |  |
| --- | --- | --- |
|  |  |  |
| **Figure 18:** BNB confusion matrix on twitter sentiment. | **Figure 19:** MNB confusion matrix on twitter sentiment. | **Figure 20:** SVM confusion matrix on twitter sentiment. |

The kfold validation indicates a relatively stable result set, for each model, with a slight preference for MNB and SVM. Specifically, the latter two models contain less variability in scores.

|  |  |  |
| --- | --- | --- |
|  |  |  |
| **Figure 21:** BNB kfold validation on twitter sentiment. | **Figure 22:** MNB kfold validation on twitter sentiment. | **Figure 23:** SVM kfold validation on twitter sentiment. |

|  |  |  |
| --- | --- | --- |
|  |  |  |
| **Figure 24:** BNB precision, recall, f-beta on twitter sentiment. | **Figure 25:** MNB precision, recall, f-beta on twitter sentiment. | **Figure 26:** SVM precision, recall, f-beta on twitter sentiment. |

Top positive ngram words include ‘love tonight’, ‘best, ‘love’, ‘watch tonight’. On the other hand, negative ngram words include, ‘nonsense’, ‘shame’, ‘disappointed’, ‘deserve better’, and profane words.

|  |  |  |
| --- | --- | --- |
|  |  |  |
| **Figure 27:** top 25 tfdif words on twitter sentiment. | **Figure 28:** top 25 MNB positive words on twitter sentiment. | **Figure 29:** top 25 MNB negative words on twitter sentiment. |

|  |  |  |
| --- | --- | --- |
|  |  |  |
| **Figure 30:** BNB train distribution on twitter sentiment. | **Figure 31:** MNB train distribution on twitter sentiment. | **Figure 32:** SVM train distribution on twitter sentiment. |

## Unigram (100% train)

The kfold validation was not provided, since 100% of the data was utilized for train. When attempting to force the use of the kfold validation the following error is provided:

ValueError: k-fold cross-validation requires at least one train/test split by setting n\_splits=2 or more, got n\_splits=1.

As generally expected, using the entire dataset for train would produce very high accuracy. Specifically, each model performed in the mid 90% accuracy.

|  |  |  |
| --- | --- | --- |
|  |  |  |
| **Figure 33:** BNB confusion matrix on twitter sentiment. | **Figure 34:** MNB confusion matrix on twitter sentiment. | **Figure 35:** SVM confusion matrix on twitter sentiment. |

|  |  |  |
| --- | --- | --- |
|  |  |  |
| **Figure 39:** BNB precision, recall, f-beta on twitter sentiment. | **Figure 40:** MNB precision, recall, f-beta on twitter sentiment. | **Figure 41:** SVM precision, recall, f-beta on twitter sentiment. |

|  |  |  |
| --- | --- | --- |
|  |  |  |
| **Figure 42:** top 25 tfdif words on twitter sentiment. | **Figure 43:** top 25 MNB positive words on twitter sentiment. | **Figure 44:** top 25 MNB negative words on twitter sentiment. |

|  |  |  |
| --- | --- | --- |
|  |  |  |
| **Figure 45:** BNB train distribution on twitter sentiment. | **Figure 46:** MNB train distribution on twitter sentiment. | **Figure 47:** SVM train distribution on twitter sentiment. |

# Conclusions

1. <https://www.cheatsheet.com/entertainment/is-game-of-thrones-the-most-watched-show-in-hbos-history.html/> [↑](#footnote-ref-1)
2. <https://www.cnn.com/2019/05/20/media/game-of-thrones-finale-ratings/index.html> [↑](#footnote-ref-2)
3. <https://en.wikipedia.org/wiki/List_of_Game_of_Thrones_episodes> [↑](#footnote-ref-3)
4. <https://blog.reedsy.com/three-act-structure/> [↑](#footnote-ref-4)
5. <https://www.change.org/p/hbo-remake-game-of-thrones-season-8-with-competent-writers> [↑](#footnote-ref-5)
6. <https://gizmodo.com/game-of-thrones-petition-to-remake-season-8-reaches-a-m-1834875436> [↑](#footnote-ref-6)
7. <https://www.youtube.com/watch?v=0IeTz5W0u5c> [↑](#footnote-ref-7)
8. <https://www.youtube.com/watch?v=G0mncEl4nVU&feature=youtu.be> [↑](#footnote-ref-8)
9. <https://towardsdatascience.com/using-natural-language-processing-to-rate-the-sentiment-of-the-game-of-thrones-finale-cf3afaf636a2> [↑](#footnote-ref-9)
10. <https://github.com/cjhutto/vaderSentiment> [↑](#footnote-ref-10)
11. <https://scikit-learn.org/stable/modules/generated/sklearn.metrics.precision_recall_fscore_support.html> [↑](#footnote-ref-11)
12. <https://en.wikipedia.org/wiki/Precision_and_recall> [↑](#footnote-ref-12)
13. <https://www.youtube.com/watch?v=Clo-t9eeEwg> [↑](#footnote-ref-13)
14. <https://en.wikipedia.org/wiki/Tf%E2%80%93idf> [↑](#footnote-ref-14)
15. <https://scikit-learn.org/stable/modules/generated/sklearn.naive_bayes.MultinomialNB.html> [↑](#footnote-ref-15)
16. <https://scikit-learn.org/stable/modules/generated/sklearn.feature_extraction.text.TfidfVectorizer.html> [↑](#footnote-ref-16)
17. <https://developer.twitter.com/en/docs/tweets/tweet-updates.html> [↑](#footnote-ref-17)
18. <https://developer.twitter.com/en/docs/tweets/search/api-reference/get-search-tweets.html> [↑](#footnote-ref-18)
19. <https://twython.readthedocs.io/en/latest/usage/basic_usage.html> [↑](#footnote-ref-19)
20. <https://github.com/jeff1evesque/ist-736-hw/tree/master/data/twitter> [↑](#footnote-ref-20)
21. <https://github.com/jeff1evesque/ist-736-hw/blob/master/hw7/data/twitter/sample.csv> [↑](#footnote-ref-21)
22. <https://www.mturk.com/> [↑](#footnote-ref-22)