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](#_Toc9886260)

[Analysis 2](#_Toc9886261)

[Data Preparation 3](#_Toc9886262)

[Amazon Mechanical Turk 3](#_Toc9886263)

[Stop Words 4](#_Toc9886264)

[Results 4](#_Toc9886265)

[Unigram Models 4](#_Toc9886266)

[Unigram and Bigram Models 6](#_Toc9886267)

[Unigram (100% train) 8](#_Toc9886268)

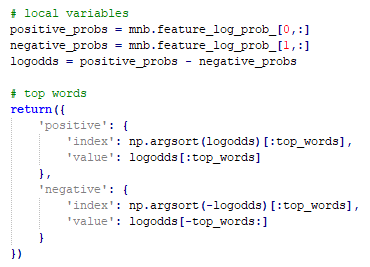
[Conclusions 9](#_Toc9886269)

# Introduction

# Analysis

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[[1]](#footnote-1) 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[[2]](#footnote-2). 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))[[3]](#footnote-3). 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)[[4]](#footnote-4). Furthermore, the MNB provides log probabilities[[5]](#footnote-5), to determine positive and negative words associated with corresponding sentiment. The associated ‘*value’* calculation provides a somewhat arbitrary measure:



Two additional cases were conducted in a similar fashion. However, rather than implementing a unigram representation an ngram\_range=(1,2)[[6]](#footnote-6) 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[[7]](#footnote-7), the unigram implementation was most suitable.

# Data Preparation

Data was acquired using the standard twitter search api[[8]](#footnote-8) via the twython python package[[9]](#footnote-9). 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[[10]](#footnote-10). However, to improve performance on successive analysis, an intermediate sample.csv[[11]](#footnote-11) 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)[[12]](#footnote-12). 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 1:** BNB confusion matrix on twitter sentiment. | **Figure 2:** MNB confusion matrix on twitter sentiment. | **Figure 3:** 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 4:** BNB kfold validation on twitter sentiment. | **Figure 5:** MNB kfold validation on twitter sentiment. | **Figure 6:** SVM kfold validation on twitter sentiment. |

|  |  |  |
| --- | --- | --- |
|  |  |  |
| **Figure 7:** BNB precision, recall, f-beta on twitter sentiment. | **Figure 8:** MNB precision, recall, f-beta on twitter sentiment. | **Figure 9:** 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 otherhand, positive words include best, bold, love, behindthescene.

|  |  |  |
| --- | --- | --- |
|  |  |  |
| **Figure 10:** top 25 tfdif words on twitter sentiment. | **Figure 11:** top 25 MNB positive words on twitter sentiment. | **Figure 12:** 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 of positive versus negative sentiment.

|  |  |  |
| --- | --- | --- |
|  |  |  |
| **Figure 13:** BNB train distribution on twitter sentiment. | **Figure 14:** MNB train distribution on twitter sentiment. | **Figure 15:** 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 16:** BNB confusion matrix on twitter sentiment. | **Figure 17:** MNB confusion matrix on twitter sentiment. | **Figure 18:** 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 19:** BNB kfold validation on twitter sentiment. | **Figure 20:** MNB kfold validation on twitter sentiment. | **Figure 21:** SVM kfold validation on twitter sentiment. |

|  |  |  |
| --- | --- | --- |
|  |  |  |
| **Figure 22:** BNB precision, recall, f-beta on twitter sentiment. | **Figure 23:** MNB precision, recall, f-beta on twitter sentiment. | **Figure 24:** 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 25:** top 25 tfdif words on twitter sentiment. | **Figure 26:** top 25 MNB positive words on twitter sentiment. | **Figure 27:** top 25 MNB negative words on twitter sentiment. |

|  |  |  |
| --- | --- | --- |
|  |  |  |
| **Figure 28:** BNB train distribution on twitter sentiment. | **Figure 29:** MNB train distribution on twitter sentiment. | **Figure 30:** 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 31:** BNB confusion matrix on twitter sentiment. | **Figure 32:** MNB confusion matrix on twitter sentiment. | **Figure 33:** SVM confusion matrix on twitter sentiment. |

|  |  |  |
| --- | --- | --- |
|  |  |  |
| **Figure 37:** BNB precision, recall, f-beta on twitter sentiment. | **Figure 38:** MNB precision, recall, f-beta on twitter sentiment. | **Figure 39:** SVM precision, recall, f-beta on twitter sentiment. |

|  |  |  |
| --- | --- | --- |
|  |  |  |
| **Figure 40:** top 25 tfdif words on twitter sentiment. | **Figure 41:** top 25 MNB positive words on twitter sentiment. | **Figure 42:** top 25 MNB negative words on twitter sentiment. |

|  |  |  |
| --- | --- | --- |
|  |  |  |
| **Figure 43:** BNB train distribution on twitter sentiment. | **Figure 44:** MNB train distribution on twitter sentiment. | **Figure 45:** SVM train distribution on twitter sentiment. |

# Conclusions

1. <https://scikit-learn.org/stable/modules/generated/sklearn.metrics.precision_recall_fscore_support.html> [↑](#footnote-ref-1)
2. <https://en.wikipedia.org/wiki/Precision_and_recall> [↑](#footnote-ref-2)
3. <https://www.youtube.com/watch?v=Clo-t9eeEwg> [↑](#footnote-ref-3)
4. <https://en.wikipedia.org/wiki/Tf%E2%80%93idf> [↑](#footnote-ref-4)
5. <https://scikit-learn.org/stable/modules/generated/sklearn.naive_bayes.MultinomialNB.html> [↑](#footnote-ref-5)
6. <https://scikit-learn.org/stable/modules/generated/sklearn.feature_extraction.text.TfidfVectorizer.html> [↑](#footnote-ref-6)
7. <https://developer.twitter.com/en/docs/tweets/tweet-updates.html> [↑](#footnote-ref-7)
8. <https://developer.twitter.com/en/docs/tweets/search/api-reference/get-search-tweets.html> [↑](#footnote-ref-8)
9. <https://twython.readthedocs.io/en/latest/usage/basic_usage.html> [↑](#footnote-ref-9)
10. <https://github.com/jeff1evesque/ist-736-hw/tree/master/data/twitter> [↑](#footnote-ref-10)
11. <https://github.com/jeff1evesque/ist-736-hw/blob/master/hw7/data/twitter/sample.csv> [↑](#footnote-ref-11)
12. <https://www.mturk.com/> [↑](#footnote-ref-12)