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](#_Toc9812292)

[Analysis 2](#_Toc9812293)

[Data Preparation 2](#_Toc9812294)

[Results 2](#_Toc9812295)

[Unigram Models 2](#_Toc9812296)

[Unigram and Bigram Models 2](#_Toc9812297)

[Unigram (100% train) 2](#_Toc9812298)

[Conclusions 2](#_Toc9812299)

# Introduction

# Analysis

The Multinomial Naïve Bayes (MNB), and Support Vector Machines (SVM) were used to classify tweets into a series of sentiment categories. Specifically, tweets related to the HBO series Game of Thrones (GOT) were compiled, then trained against a unigram representation. Kfold validation, and confusion matrices were computed to assess model performance. Furthermore, precision, recall, and the f-ratio measures[[1]](#footnote-1) were also used. 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)[[2]](#footnote-2). Though the MNB provides the log probabilities[[3]](#footnote-3), which can be tailored to determine indicative positive and negative words, a more general approach was used. This allowed the approach to apply in a broader way, rather than devising an equivalent method for each model type.

Two additional cases were conducted in a similar fashion. However, rather than implementing a unigram representation an ngram\_range=(1,2)[[4]](#footnote-4) 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[[5]](#footnote-5), the unigram implementation was most suitable.

# Data Preparation

Data was acquired using the standard twitter search api[[6]](#footnote-6) via the twython python package[[7]](#footnote-7). Specifically, three different hash tags were aggregated using the api:

* GameofThrone
* GameofThronesFinal
* GOTFinale

The dates of the collected data were approximately one week after the conclusion of the 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 due to less activity than the former two hashtags.

Moreover, each hashtag was written to a corresponding csv file[[8]](#footnote-8). However, to improve performance on successive analysis, an intermediate sample.csv[[9]](#footnote-9) 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]]

This reduced the space from a combined 3670 rows, then was passed to Amazon Mechanical Turk (MTurk)[[10]](#footnote-10). 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

)

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.

# Results

## Unigram Models

## Unigram and Bigram Models

## Unigram (100% train)

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